

Deep Learning per Computer Vision (CV)

Francesco Pugliese

DIRM-DCME-MEC
frpuglie@istat.it

Programma del Corso del 4 Dicembre 2019

18/11 – Deep Learning per Computer Vision (CV):

Mattina - Teoria:

- Reti Neurali Convulsive (CNN): Convoluzione, Campo Recettivo, Stride, 0-Padding, Feature Map, Funzioni d'attivazione, Strato di Pooling
- Architetture di Reti Neurali Convulsive: LeNet, AlexNet, VggNet, Inception, ResNet, CapsNet, InceptionResNet, NasNet, PixelCNN, GAN
- Reti Convulsive per la segmentazione: FastRCNN, SegNet, Unet
- Satellite Imagery Classification for Land Cover Estimation – Intervento Diego Zardetto come caso di studio in Istat

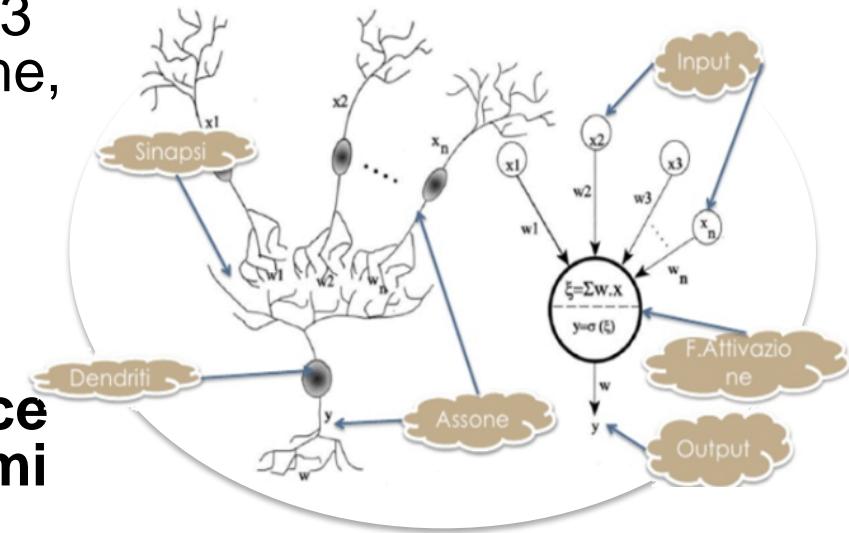
Pomeriggio - Pratica:

- Esercizi di Implementazione in Keras di Classificatori di Immagini
- Use Case 1: Deep Learning per Computer Vision

RETI NEURAL ARTIFICIALI (ANN)

Le Reti Neurali Artificiali furono introdotte, per la prima volta, nel 1943
ANNs were introduced, for the first time, in un lavoro sulla formalizzazione dell'attività neurale in forma logica proposizionale (McCulloch & Pitts, 1943).

Possiamo definire le Reti Neurali Artificiali come un modello semplice del Sistema nervosa degli organismi biologici !!



Neuron Activation

$$A_j = \sum_{i=1}^N w_{ij} X_i - \theta_i$$

Activation function

$$y_j = \Phi(A_j) = \Phi(\sum_{i=1}^N w_{ij} X_i - \theta_i)$$

In ambito **data mining**: I metodi sono stati sviluppati per produrre modelli comprensibili e ridurre le ore di training.

1) Estrazione delle regole: estrazione di modelli simbolici da reti neurali pre-addestrate..

2) Apprendimento semplice: costruzione di reti neurali semplici da comprendere.

DEEP LEARNING: Neural Networks Profonde

Negli anni recenti le **Reti Neurali Profonde** hanno raggiunto notevoli progressi in ambiti di ricerca (*Bengio, 2009*). Questa nuova metodologia che si occupa delle reti neurali profonde e dei loro algoritmi di addestramento fu chiamata da Hinton con il termine **“Deep Learning”**, per la prima volta nel 2006. Da allora Hinton divenne il padre del **Deep Learning** e della moderna Intelligenza Artificiale in generale. Finora, **in tutti gli esperimenti fatti**, le prestazioni risultanti del deep learning superano di gran lunga le alter tecniche di **Machine Learning** tradizionale.



GOOGLE DATACENTER

1,000 CPU Servers
2,000 CPUs • 16,000 cores

600 kWatts
\$5,000,000



STANFORD AI LAB

3 GPU-Accelerated Servers
12 GPUs • 18,432 cores

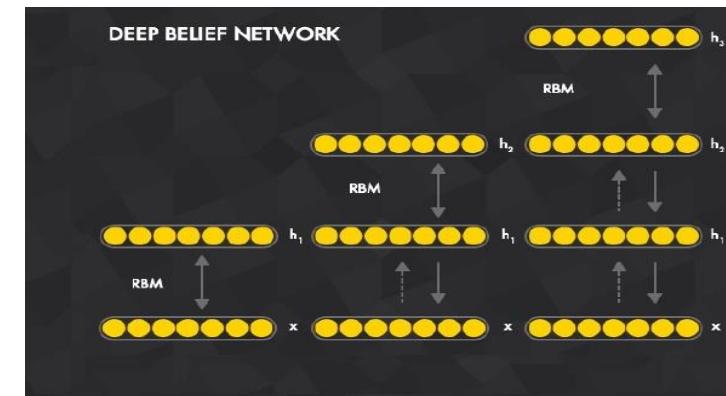
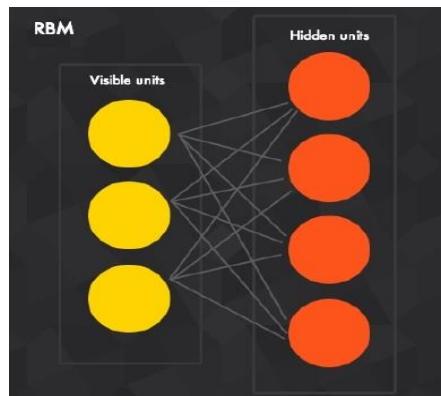
4 kWatts
\$33,000

DEEP LEARNING: un approccio rivoluzionario all'analisi dei Big Data

- Architetture **poco profonde** producono feature grezze e poco rappresentative.
- Si ha la necessità di aumentare la profondità delle reti neurali per avere una maggiore performance

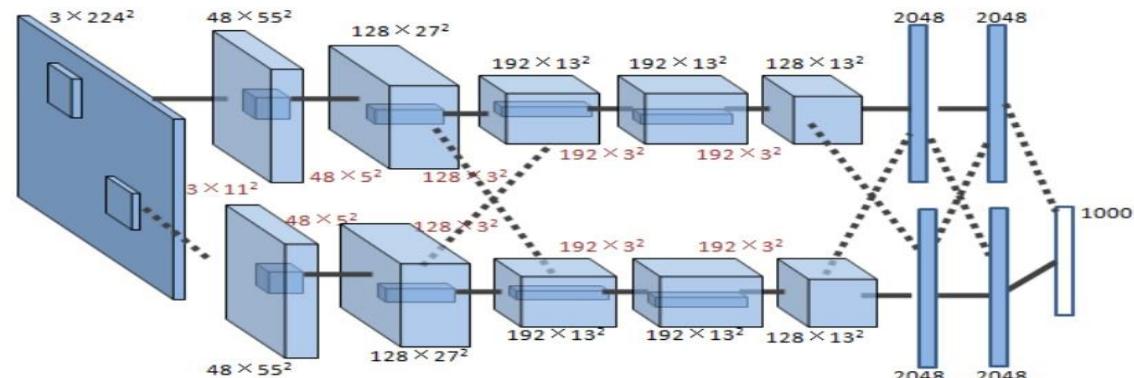
**Apprendimento non Supervisionato
Unsupervised Learning
Modelli Generativi:**

- Restricted Boltzmann Machines (RBM)
- Deep Belief Networks (Hinton et al., 2006).
- Generative Adversarial Networks (GAN)

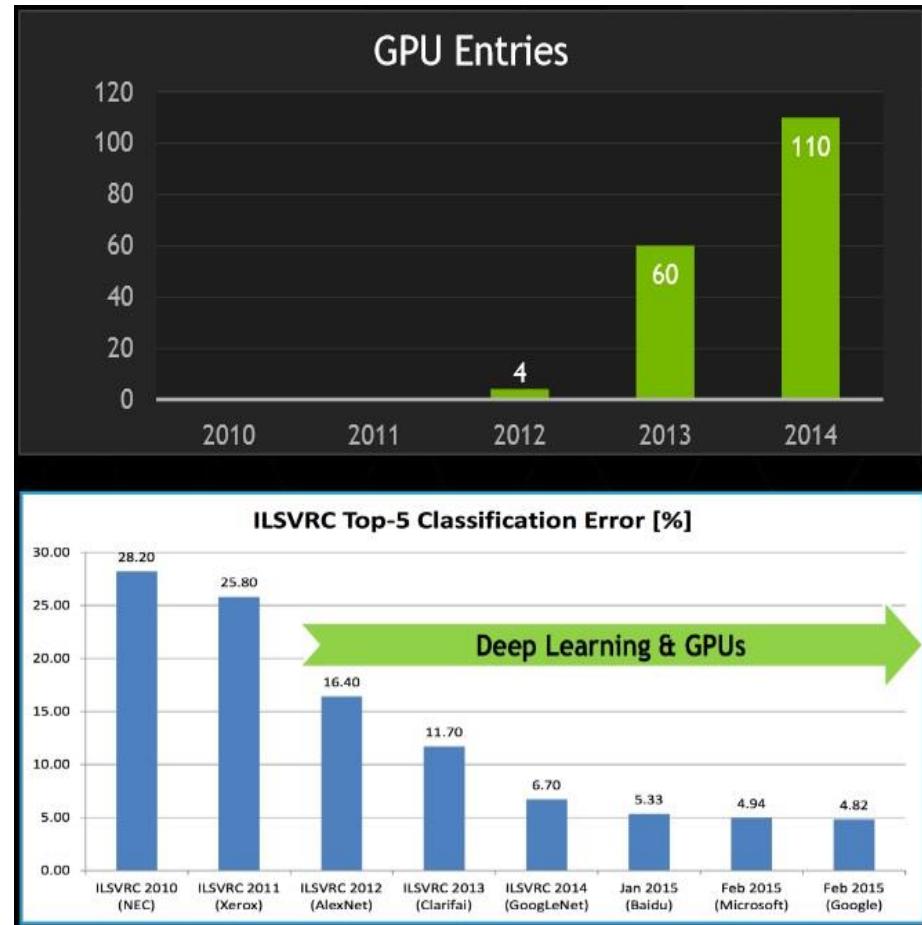
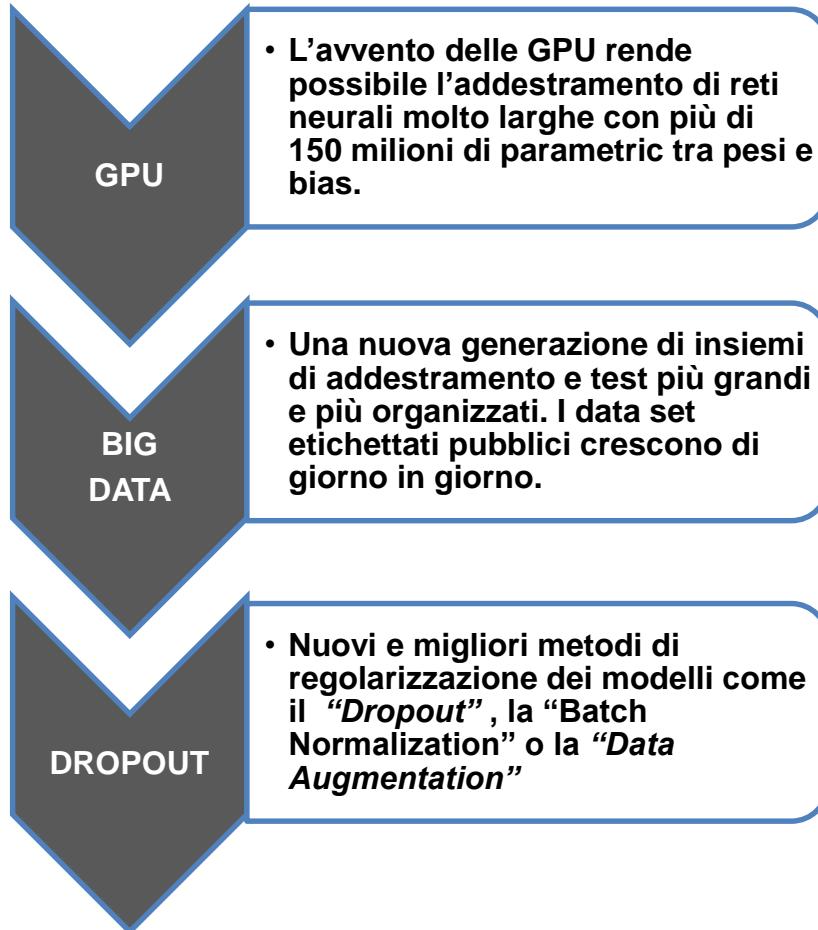


**Apprendimento Supervisionato
Supervised Learning
(Modelli Discriminativi):**

- Deep Convolutional Neural Networks (LeCun et al., 1989).
- Back Propagation

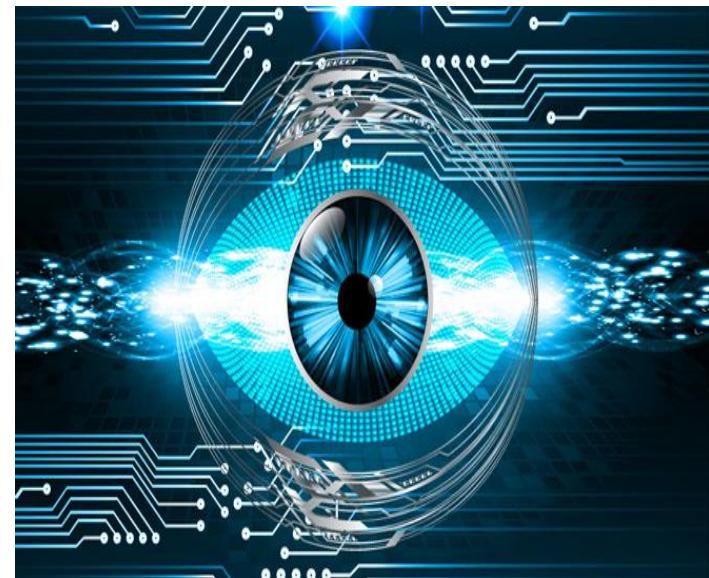


DEEP LEARNING: un approccio rivoluzionario all'analisi dei Big Data



Computer Vision: dove la Statistica tradizionale fallisce.

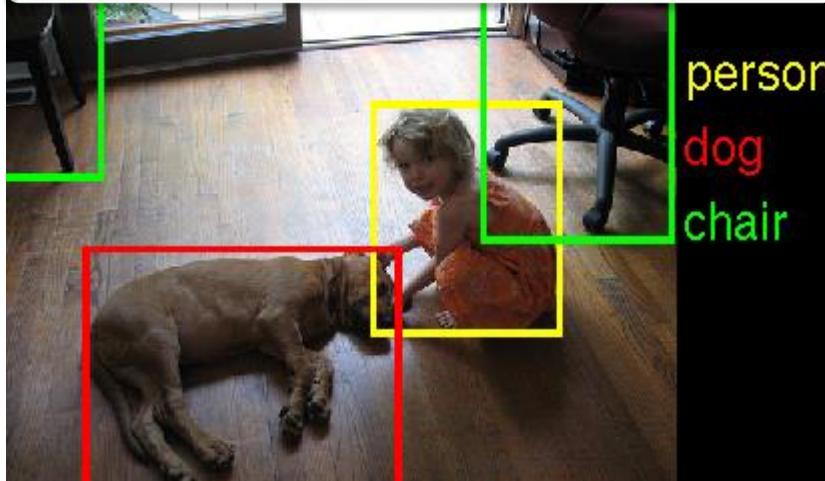
- La **Computer Vision** è un campo interdisciplinare che si occupa del modo in cui una nuova generazione di algoritmi è in grado di elaborare immagini o video per estrarre informazione significativa (da una prospettiva umana) di alto livello.
- I metodi statistici tradizionali hanno spesso enormi problemi ad elaborare big data visuali costituiti da immagini e video.
- L'inadeguatezza dei metodi matematico/statistici tradizionali nel settore della **Computer Vision** è principalmente dovuta al fatto che spesso non sono in grado di essere scalati agevolmente su big data ad alto volume e ad alta dimensionalità (numero di attributi o features) come nel caso appunto delle immagini usate nella **Computer Vision** (*Chellappa, R., 2012*). Anche una piccola immagine 28x28 produce 784 attributi da immettere nel modello. Quindi un'esplosione combinatoria di input.



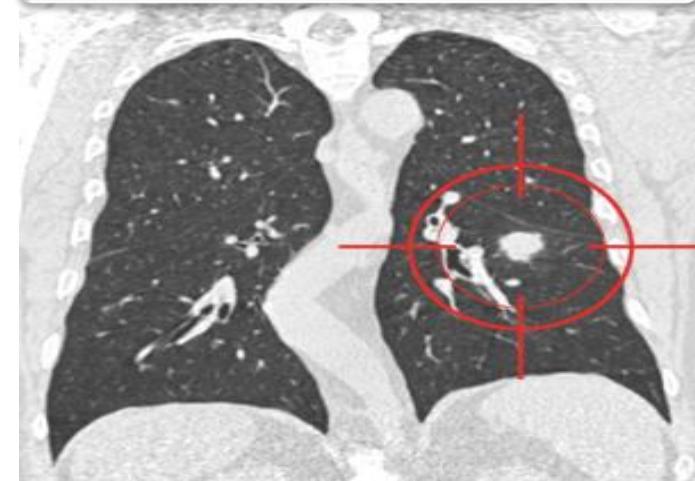
Perchè la Computer Vision è così importante?

- Un nuovo studio dimostra la relazione tra le capacità di Visione e l'Intelligenza. A quanto pare gli esseri umani hanno uno degli apparati di visione più sofisticati nel mondo animale e questo correlerebbe con l'evoluzione dell'intelligenza umana. (*Tsukahara et al., 2016*).
- In altri termini, la **Computer Vision** ha bisogno di capacità d'intelletto di tipo umano o superiori per avere l'efficienza ricercata.

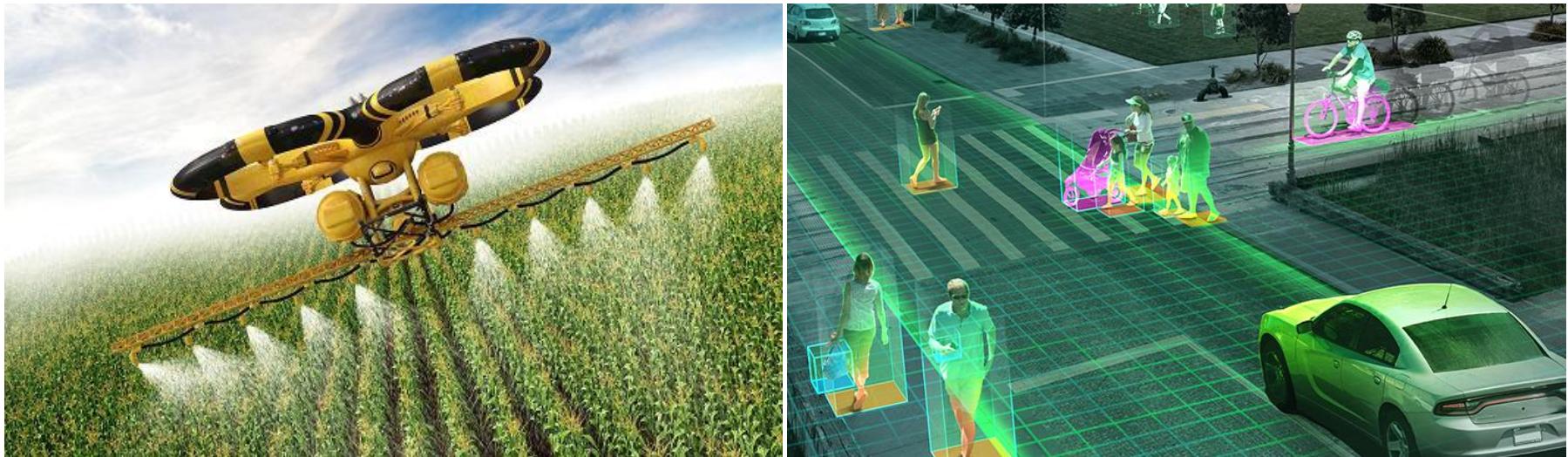
EVERYDAY LIFE



BIOMEDICAL IMAGES



Perchè la Computer Vision è così importante?



- Grazie alla **Computer Vision** si potrebbe costruire una nuova generazione di macchine in grado di compiere task tipicamente umani come riconoscere e spostare oggetti, guidare automobile, coltivare ed irrigare campi autonomamente, pulire strade, raccogliere spazzatura nelle città, e altro.

Computer Vision al servizio del riciclo

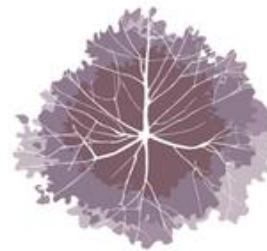
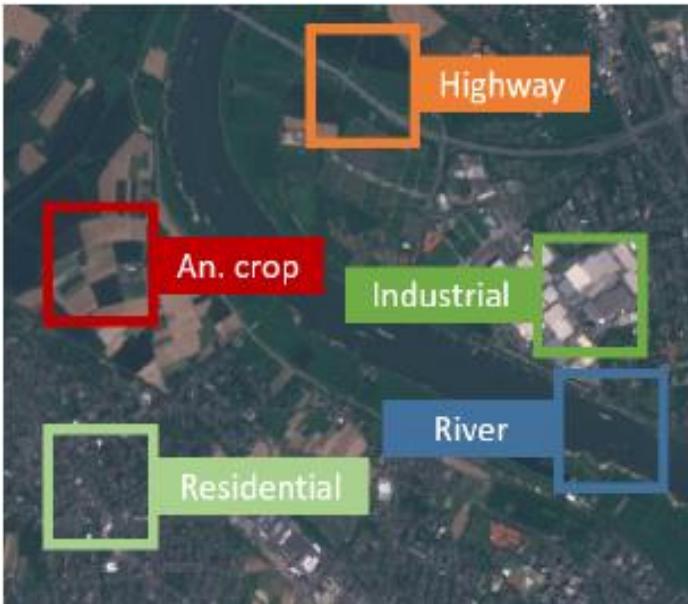
- **Cassonetti intelligenti** potrebbero classificare i rifiuti e **differenziarli** automaticamente riducendo enormemente l'errore umano.
- Per quanto concerne la **sostenibilità** del cibo, un primo classificatore intelligente potrebbe consentire a un ristorante o a un negozio di riconoscere, se vi è cibo ancora utile risultante dal processo produttivo. Poi utilizzando altri sotto-classificatori è possibile comprendere se questo cibo può essere rigenerato.
- Applicazioni di AI che ottimizzano tutto il flusso di interazione con un ristorante, riducendo i **costi del processo** e aumentando la sostenibilità. In un altro lavoro, i ricercatori hanno generato, grazie all'intelligenza artificiale, i menu di alcuni ristoranti analizzando le informazioni non strutturate provenienti dai commenti online dei clienti (Deep Learning al servizio del riciclo, Pugliese).



Computer Vision per la Statistica Ufficiale

Estrazione automatica di Statistiche da Immagini Satellitari

Oggiorno, sono disponibili sempre più dati da immagini satellitari aggiornati per l'osservazione della Terra.



Tuttavia, per comprendere bene l'utilizzo di questi dati, per estrarre automaticamente statistiche, le immagini satellitari devono essere processate e trasformate in strutture semantiche.

Bag of Words e Term Document Matrix (TDM)

Document

In the beginning God created
the heaven and the earth.
And the earth was without form,
and void; and darkness was
upon the face of the deep.
And the Spirit of God moved
upon the face of the waters.
And God said, Let there be
light: and there was light.

Representation

beginning	1
earth	2
God	3

Type	relax	carrello	prenotare	shop	wishlist
0	0	0	0	0	0
0	0	0	0	0	0
1	0	0	0	0	0
1	0	0	1	0	0
0	0	0	0	0	0
0	0	1	0	1	1
1	1	1	1	0	0

Per ogni sito web:
Bag-Of-Words

**Bag-of-words di
tutti i siti web: T
DM**



Per ciascuna riga
della TDM:
**Codifica di immag
ini 32x32**

Classificazione di Siti Web mediante approccio basato sul riconoscimento di Immagini

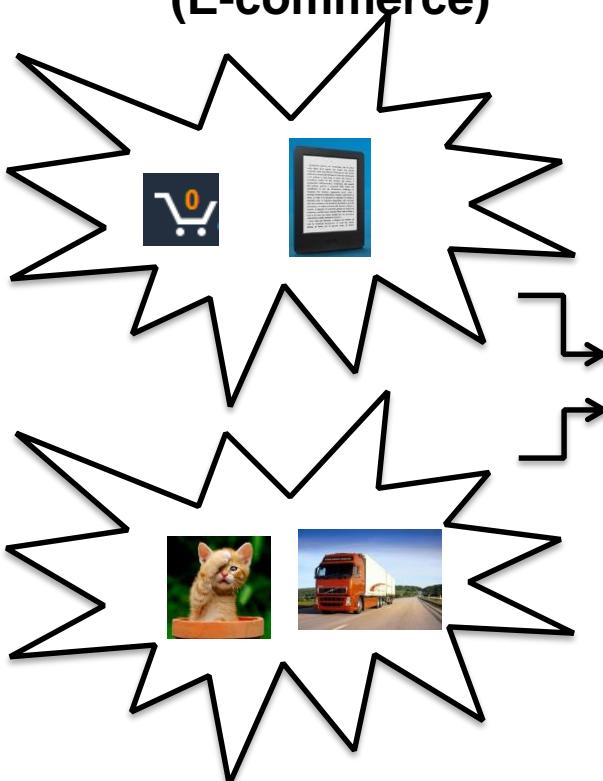
- Grazie alla tecnica della **Riduzione dei Falsi Positivi (False Positive Reduction)** sfruttiamo la segmentazione interna delle immagini di un Sito Web al fine di addestrare una rete convolutiva evoluta (ResNet) su singoli segmenti delle immagini campionate dai siti web.
- **La Rete Convolutiva** viene addestrata in modalità “**Transfer Learning**”, che significa sfruttare un modello pre-addestrato su un dataset ben-posto come Imagenet (1000 image classes, 1.2 mln images). Questo modello pre-addestrato viene addestrato poi sul problema di Computer Vision desiderato.



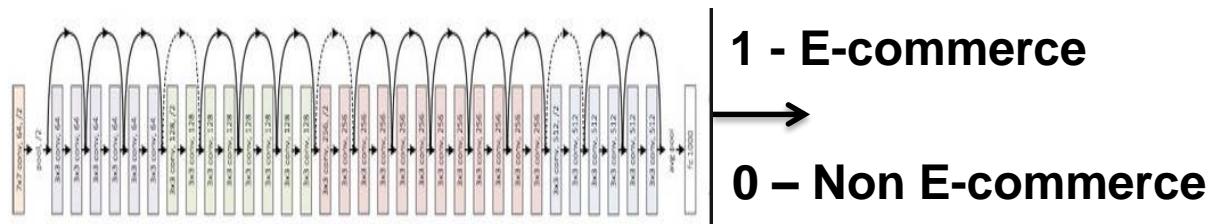
Classificazione di Siti Web mediante approccio basato sul riconoscimento di Immagini

Insieme dei Positivi

(E-commerce)



Rete Neurale Residuale (Microsoft)

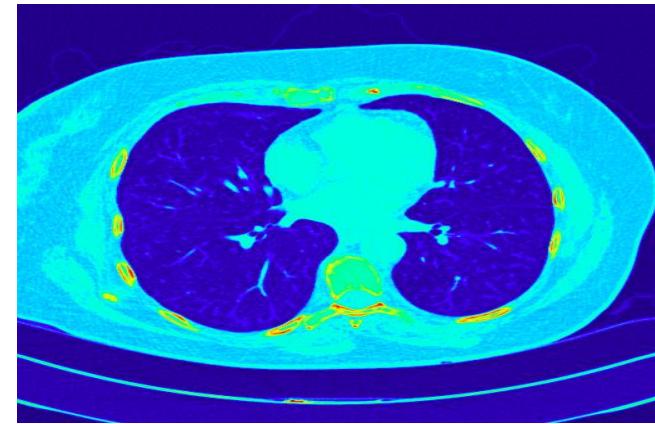


Insieme dei Negativi
(Non E-commerce)

- Nella fase di inferenza (a Run-time), applicata su un **Test Set**, le immagini del Sito Web sono ancora segmentate e viene assegnata al Sito Web l'etichetta dell'immagine con più alta probabilità.

Applicazioni Biomediche

- Negli Stati Uniti, il tumore ai polmoni uccide circa 225.000 persone ogni anno e richiede al Sistema sanitario federale circa 12 miliardi di dollari di costi. Una diagnosi precoce è importante per dare ai malati di cancro ai polmoni le migliori possibilità di guarigione e sopravvivenza.
- Nel 2016, l'ufficio del Vice Presidente degli Stati Uniti è stato a capo di un'iniziativa coraggiosa, denominata come il **“Lancio sulla luna del cancro” (Cancer Moonshot)**, nel tentativo di avviare un futuro decennio di progressi nella prevenzione del cancro, nella diagnosi e del trattamento.
- Nel 2017, il **Data Science Bowl** apparso su Kaggle (www.kaggle.com) ha rappresentato la pietra miliare nel supporto al Cancer Moonshot unendo insieme le comunità di data scientist e medici per tentare di sviluppare insieme algoritmi per l' individuazione di tumori ai polmoni da tac.



Competizioni di Computer Vision in ambito Biomedico sul sito Kaggle

Kaggle: Nel 2010, Kaggle fu fondata come piattaforma per la **modellazione predittiva** e le **competizioni analitiche** nell'ambito delle quali varie aziende e ricercatori mettevano a disposizione i loro dati.

- Data Scientist da tutto il mondo competono tra loro su Kaggle per produrre I migliori modelli.
- Il **Data Science Bowl 2017** fu la più grande competizione focalizzata sul “Lung Cancer Detection”, ovvero l’individuazione di tumori ai polmoni. La competizione fu fondata dalla **Fondazione Arnold** e il montepremi era di 1 milione di dollari. Il primo classificato vinse **500.000** dollari.

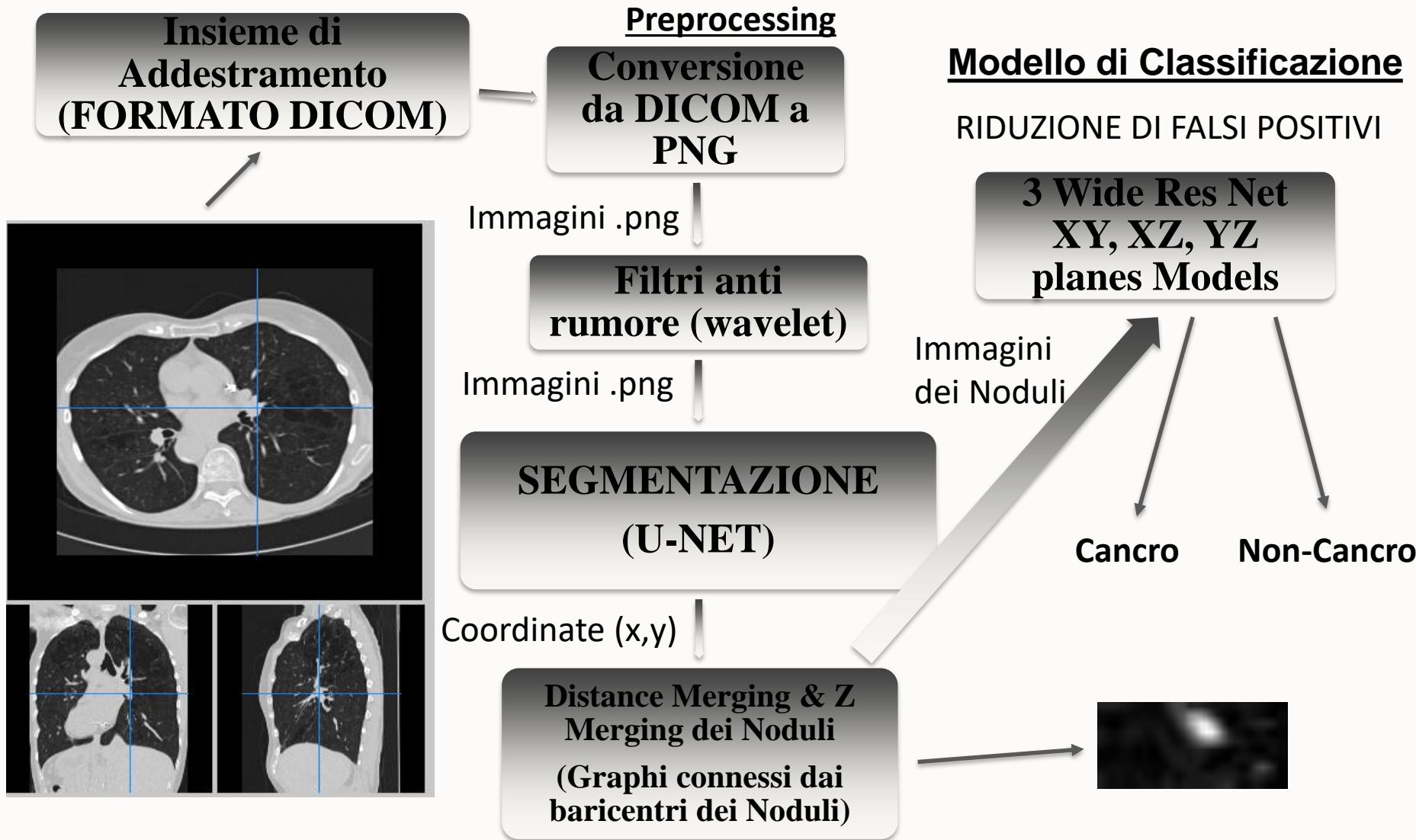
Train Set: around 150 CT labelled scans images per patient from 1200 patients encoded in **DICOM** format.

Stage 1 test set: 190 patients un-labelled CT scans.

Stage 2 test : 500 patients un-labelled CT scans.

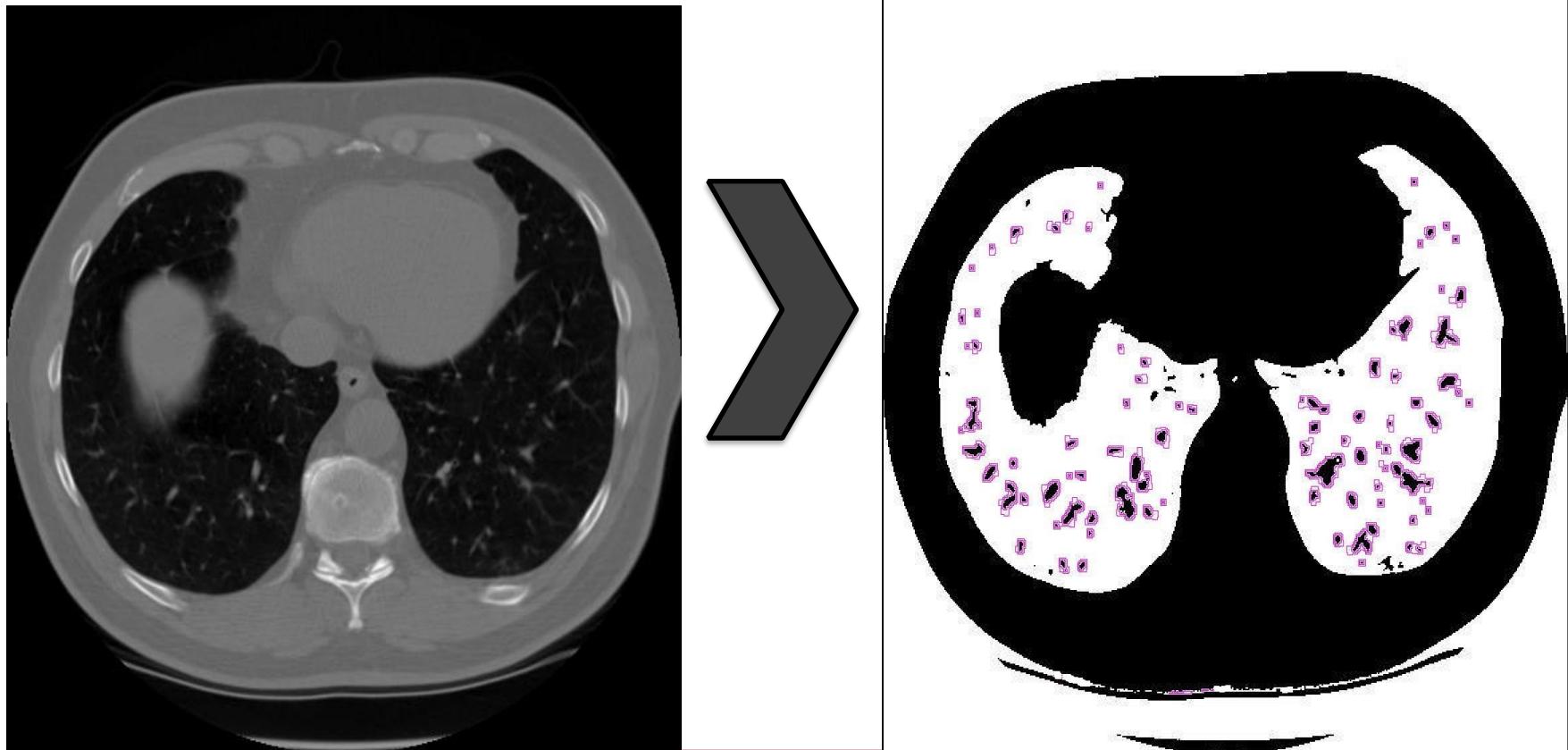


Sistema di Individuazione dei Tumori ai Polmoni



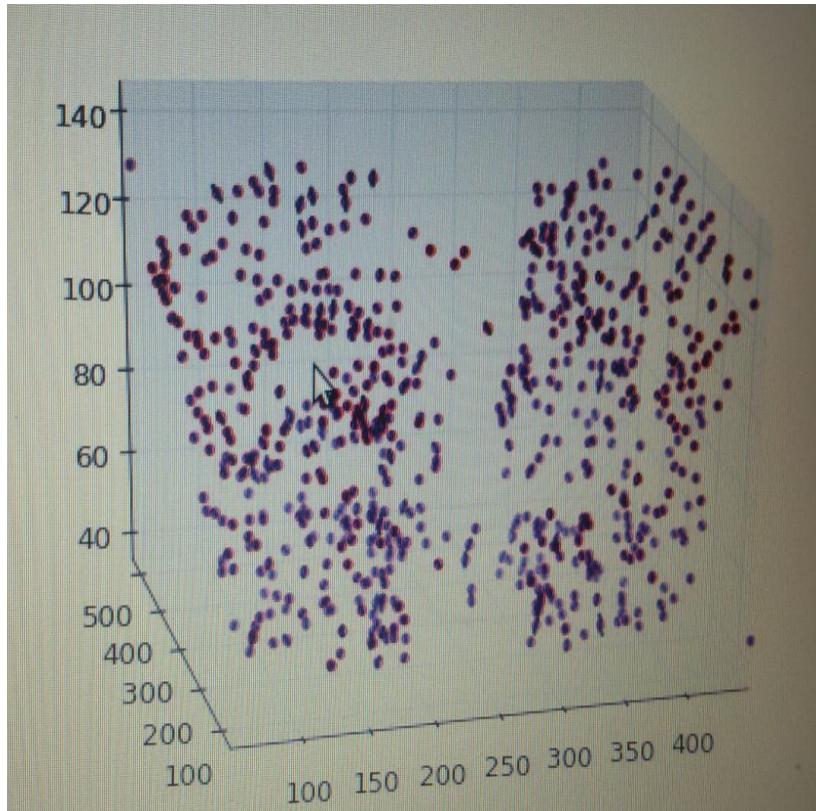
Segmentazione

- Algoritmo di Segmentazione che produce le coordinate (X,Y) del centro dei noduli che permette in seguito all'algoritmo di distance merging di estrarre I noduli direttamente dalle scansioni delle TAC (CT-Scans).

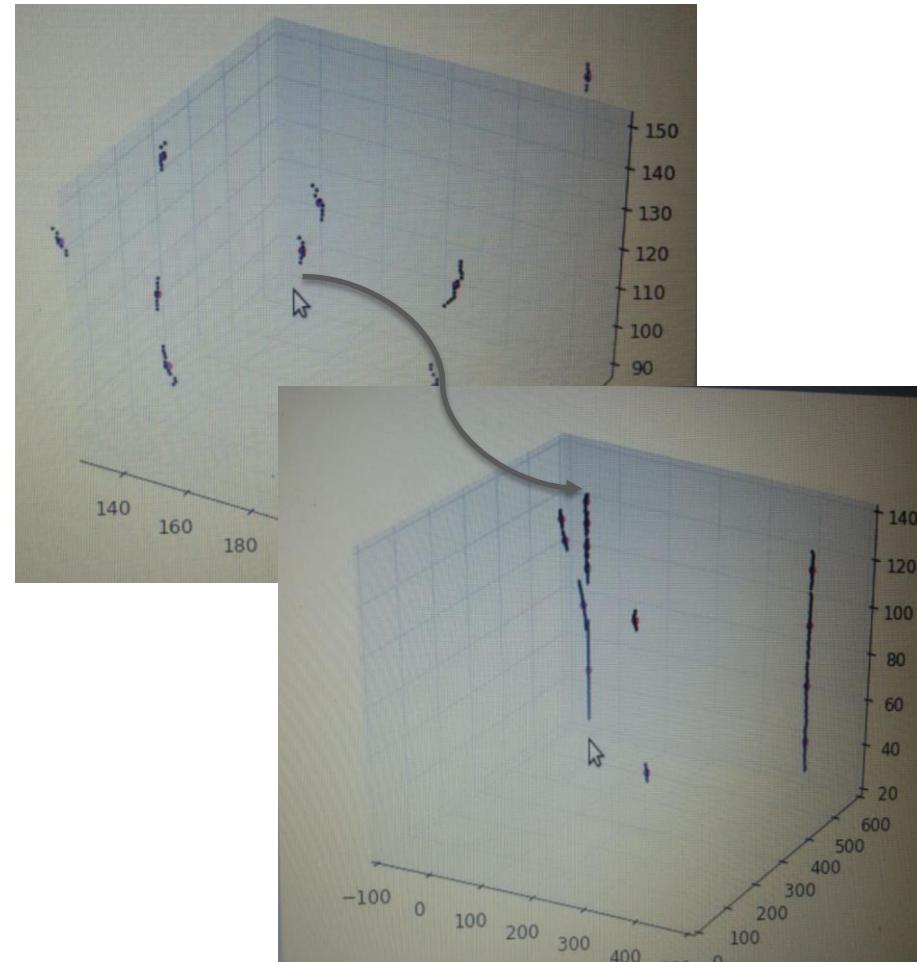


Distance Merging and Z-Merging

Distance Merging



Z-Merging



Risultati e Direzioni Future del progetto Lung Cancer

- Il software risultante dal progetto richiede enormi risorse di calcolo che sono state addestrate su cluster GPU avanzati specificatamente progettati e implementati da noi, e nonostante ciò ha richiesto giorni di addestramento.
- Tuttavia i risultati sembrano incoraggianti, e dimostrano che le accuratezze raggiunte da questi modelli di Deep Learning sono comparabili a quelle degli **oncologi** ed in alcuni casi le superano.
- Fra non molto tempo, queste sistemi di classificazione potrebbero divenire parte integrante di sistemi di TAC, applicazioni di ausilio ai medici, app per smartphones, ecc.
- In futuro si potrebbe sfruttare il **Transfer Learning** per estrarre feature astratte addestrate da altri modelli pre-addestrati su altri compiti e dati biomedici.
- Si potrebbe “wrappare” il modello in un applicazione per dispositivi mobile come smartphone Android o IOS.

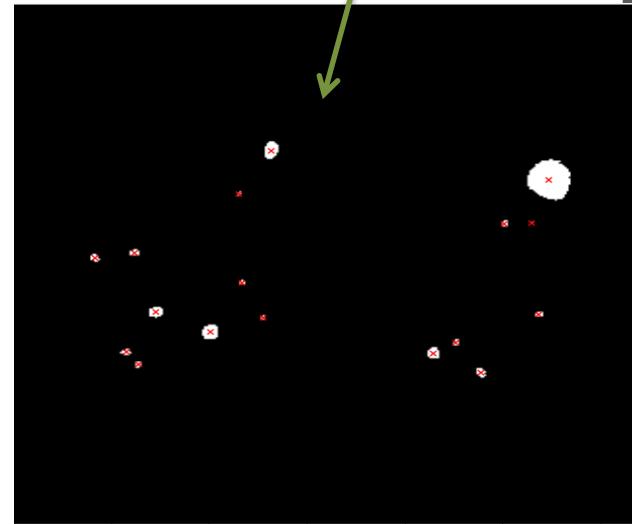
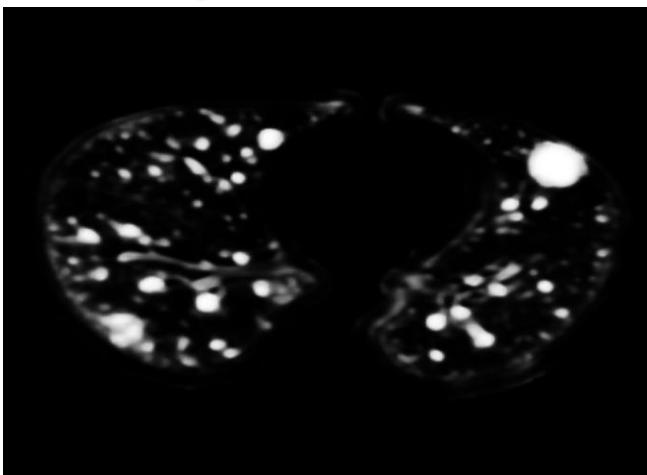
Classificazione di Tumori ai Polmoni

Selezione del
Nodulo Candidato
via UNET

Dilatazione,
Erosione, Distance
Merging dei Noduli

Riduzione di Falsi
Positivi via
WideResNet

Cancro /
Non cancro



Dataset per la Computer Vision e Competizioni

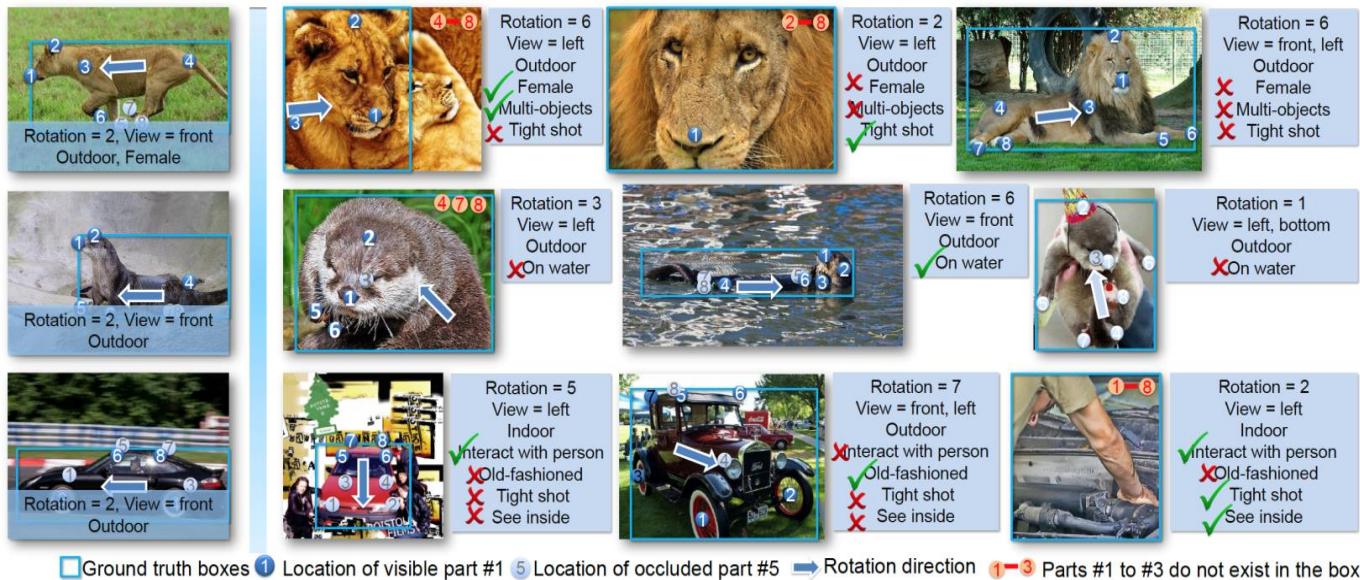
ImageNet: ImageNet è un dataset di oltre **15 milioni** labeled high-resolution images belonging to roughly **22,000** categories.

- Dal **2010** è stata istituita una competizione chiamata «**ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)**» la quale usa un sottoinsieme di ImageNet con all'incirca **1000** immagini posizionata in ognuna delle **1000** categorie disponibili.

Train Set: 1.2 million

Validation Set: 50,000

Test set: 150,000



Computer Vision Datasets and Competitions

Kaggle: In 2010, Kaggle was founded as a platform for predictive **modeling** and **analytics competitions** on which companies and researchers post their data.

- Statisticians and data scientists from all over the world compete to produce the best models.
- **Data Science Bowl 2017** was the biggest competition focused on “Lung Cancer Detection”. The competition was founded by **Arnold Foundation** and awarded **\$1 million** in prizes (**1st** ranked **\$500,000**).

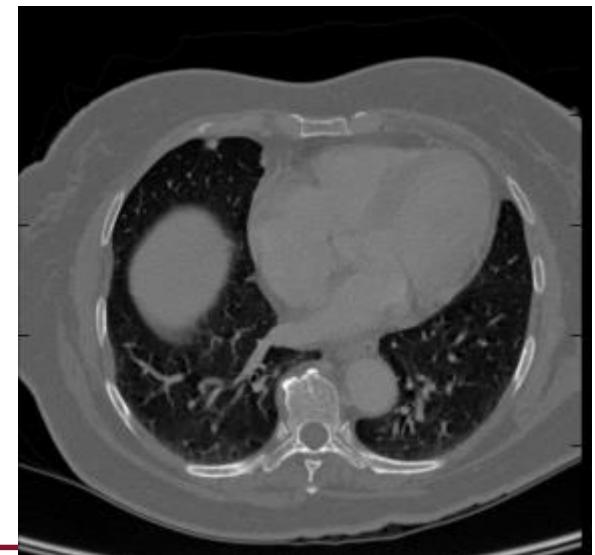
Train Set: around 150 CT labelled scans images per patient from 1200 patients encoded in **DICOM** format.

Stage 1 test set: 190 patients CT scans.

Stage 2 test : 500 patients CT scans.

Grand Challenges in Biomedical Image Analysis: This is a web-site hosting new competitions in the Biomedicine field. Specifically, **LUNA (LUng Nodule Analysis)** focuses on a large-scale evaluation of automatic nodule detection algorithms.

Train Set: LIDC/IDRI database consisting of 888 CT Scans labelled by 4 expert radiologists.

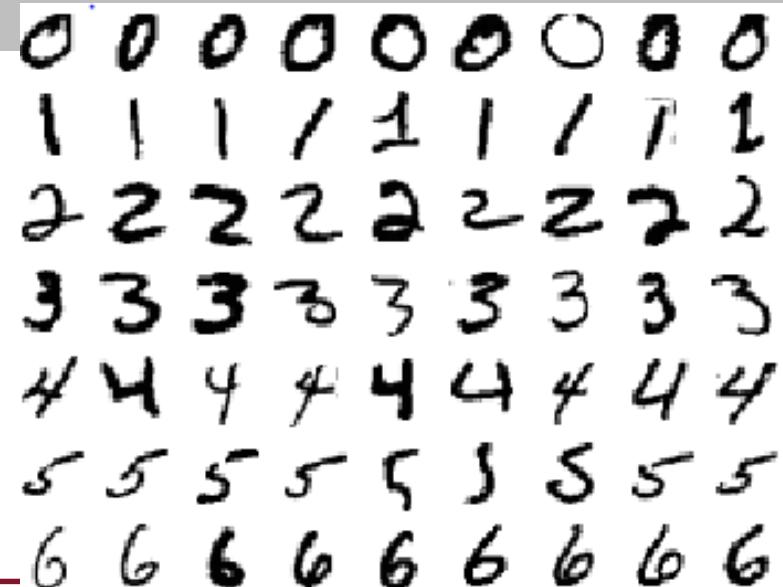
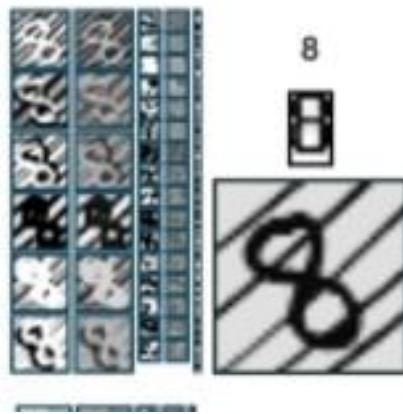


MNIST DATASET

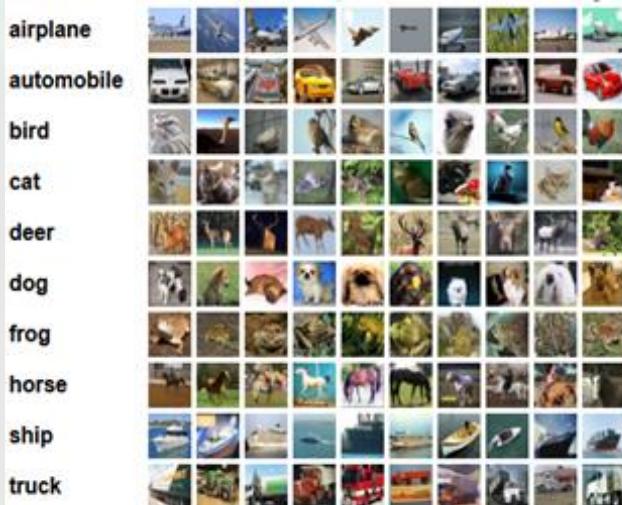
MNIST database is handwritten digits composed of 60.000 pattern.

- 60.000 training set
- 10,000 test set

LeNet has been applied to this dataset with accuracy of 0.95%.

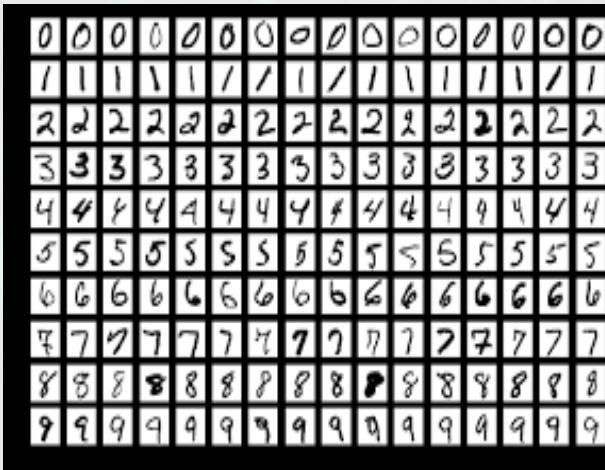


Computer Vision Datasets and Competitions



- The **CIFAR-10** dataset consists of **60,000 32x32** coloured images divided into **10 classes**, with **6000 images per class**. There are **50,000 training images** and **10,000 test images**.
- The dataset is divided into **five training batches** and one test batch, each with **10,000** images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the ²⁵ training batches contain exactly 5000 images from each class.
- The classes are completely **mutually exclusive**. There is **no overlap** between **automobiles** and **trucks**. "Automobile" includes sedans, SUVs, things of that sort. "Truck" includes only big trucks. Neither includes pickup trucks.

Computer Vision Datasets and Competitions



- The **MNIST** database of handwritten digits, available from this page, has a training set of **60,000** examples, and a test set of **10,000** examples.
- It is a subset of a larger set available from **NIST**. The digits have been size-normalized and centered in a fixed-size image.
- It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on **preprocessing** and **formatting**.
- The images contain **grey levels** as a result of the **anti-aliasing technique** used by the normalization algorithm. The images were centered in a **28x28** image by computing the center of mass of the pixels, and translating the image so as to position this point at the center of the **28x28** field.

26

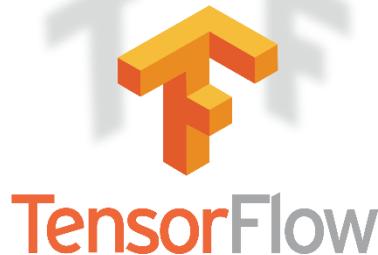
Computer Vision Datasets and Competitions



- **Fashion MNIST** is a dataset of **Zalando's** article images consisting of a training set of **60,000** images and a test set of **10,000** examples.
- Each example is a **28 x 28** grayscale image associated with a label from **10 classes**.
- Reasons for replacing MNIST according to **Zalando** opinions:
 - **MNIST is too easy.** Convolutional nets can achieve ²⁷ **99.7%** on **MNIST**. Classic machine learning algorithms can also achieve 97% easily.
 - **MNIST is overused and cannot represent modern CV tasks**, as noted by deep learning **expert/Keras author François Chollet**.

Framework disponibili per il Deep Learning

Keras è un'interfaccia di alto livello per Theano, Tensorflow e Cntk che lavorano in back-end. Keras dispone di un più intuitiva set di istruzioni e astrazioni che rendono semplice il compito di configurare una rete neurale senza dover andare troppo nel dettagli delle librerie di calcolo **scientifico e parallelo**.



TensorFlow è una libreria open-source per la programmazione di flussi di dati e grafi computazionali attraverso un insieme di compiti. E' una libreria matematica simbolica ed anche usata per le applicazioni di machine learning come le reti neurali, le svm, ecc. Viene utilizzata in Google stessa sia per scopi di ricerca che di produzione..

PyTorch è una libreria di deep learning open-source per Python, derivate da Torch, usata per applicazioni come Computer Vision e Natural Language Processing. E' stata primariamente sviluppata dal Gruppo di ricerca di Intelligenza Artificiale di Facebook, e dal Software Pytro di Uber per **programmazione probabilistica**.



Il problema della classificazione di un'Immagine

- La classificazione di un'immagine rappresenta il compito di trasformare un'immagine di input in una classe (un gatto, un cane, ecc) o una probabilità di quella classe che descrive meglio l'immagine. Per noi esseri umani, questo compito di riconoscimento è una delle prime capacità che impariamo.
- Tutte le volte che vediamo un'immagine o quando guardiamo il mondo intorno a noi, la maggior parte del tempo noi caratterizziamo la scena e diamo a ciascun oggetto un'etichetta, tutto questo senza neanche saperlo consciamente.



Il problema della classificazione di un'Immagine



Ciò che noi vediamo

06 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 48 04 56 62 00
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80
24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50
32 98 81 28 64 23 67 10 26 38 40 67 59 54 70 66 18 38 64 70
67 26 20 68 02 62 12 20 95 63 94 39 63 08 40 91 66 49 94 21
24 55 58 05 66 73 99 26 97 17 78 78 96 83 14 88 34 89 63 72
21 36 23 09 75 00 76 44 20 45 35 14 00 61 33 97 34 31 33 95
78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92
16 39 05 42 96 35 31 47 55 55 88 24 00 17 54 24 36 29 85 57
86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58
19 80 81 68 05 94 47 49 28 73 92 13 56 52 17 77 04 89 55 40
04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66
88 36 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69
04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 36
20 69 36 41 72 30 23 88 34 62 99 69 82 67 59 85 74 04 36 16
20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 57 05 54
01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 89 19 67 48

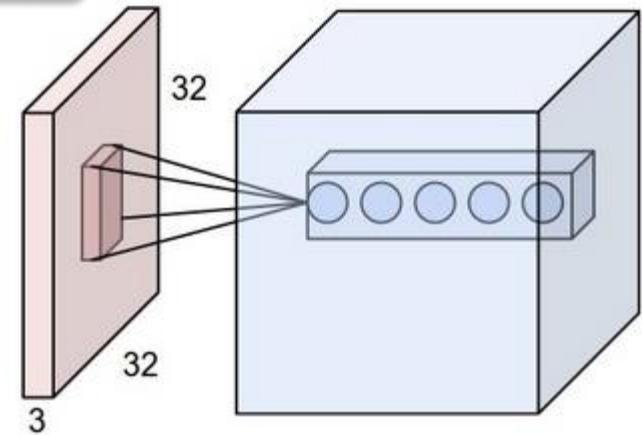
Ciò che i computer vedono

Reti Neurali Convulsive (ConvNet or CNN)

Le Reti Neurali Convulsive o Convoluzionali (CNN) sono varianti biologicamente ispirate dei Percettroni multi-strato (**Multi Layer Perceptrons - MLP**). Sappiamo che la corteccia visiva contiene una disposizione complessa di cellule (**Hubel, D. and Wiesel, T., 1968**). Questi neuroni sono sensibili a piccolo sotto-regioni del campo visivo, chiamato “**Campo Recettivo**”. Oltre allo strato convoluzionale altri strati delle **CNN** possono essere: lo strato **ReLU**, lo strato di **Pool**. Tipici parametrici delle **CNN**: a) **Numero di Filtri (Kernel)**, b) **Dimensione del Campo Recettivo**, b) **0-Padding**, c) **Stride**. Questi parametrici sono legati dalla seguente formula:

$$(W - F + 2P)/(S + 1)$$

Ciascun neurone nello strato convoluzionale è connesso spazialmente solo ad una regione locale dell'input. In questo caso ci sono 5 neuroni attraverso la profondità tutti connessi alla stessa regione.



Tipici settaggi degli strati Convolutivi

a) Numero di Filtri (Kernel)

b) Dimensione del Campo Recettivo

c) 0-Padding

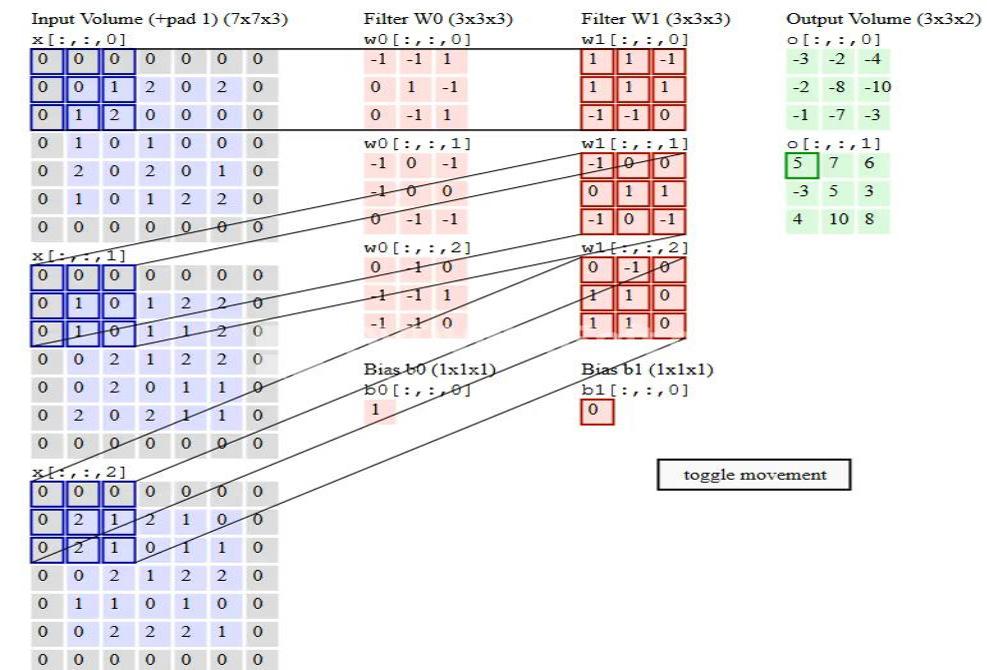
d) Stride

Altri strati:

- Strato di Pool
- Strato di Attivazione (ReLU, TanH, Sigmoid)
- Strato Completamente Connesso

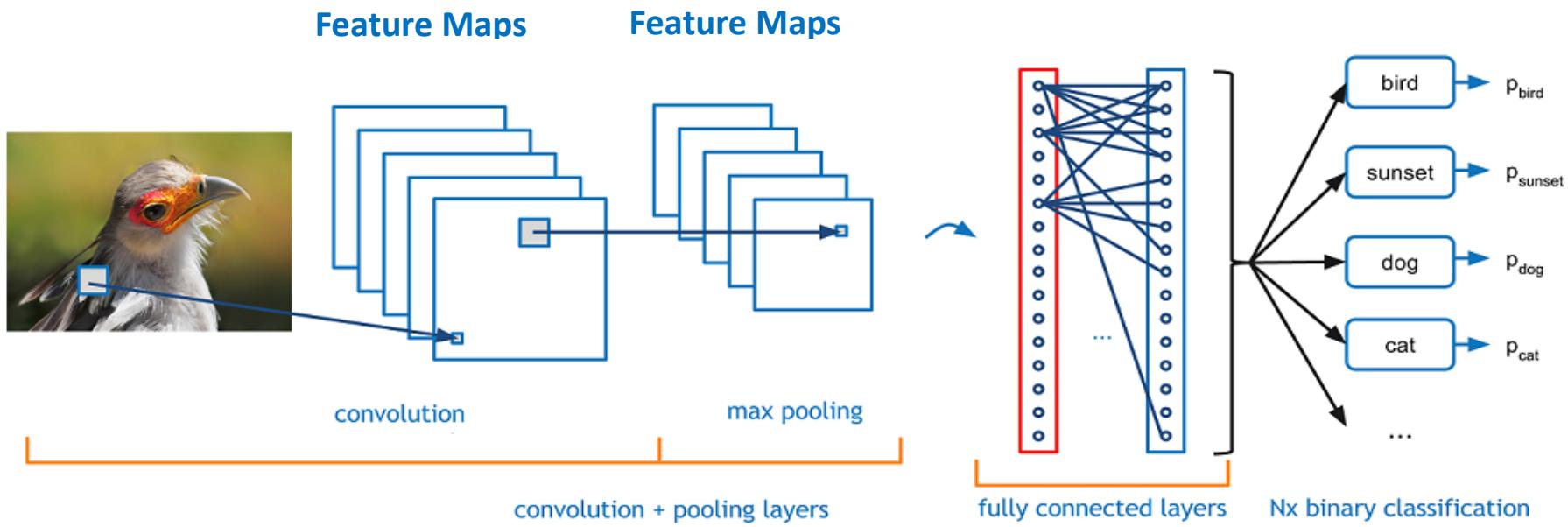
Questi parametri sono legati insieme dalla seguente formula:

$$(W - F + 2P)/(S + 1)$$

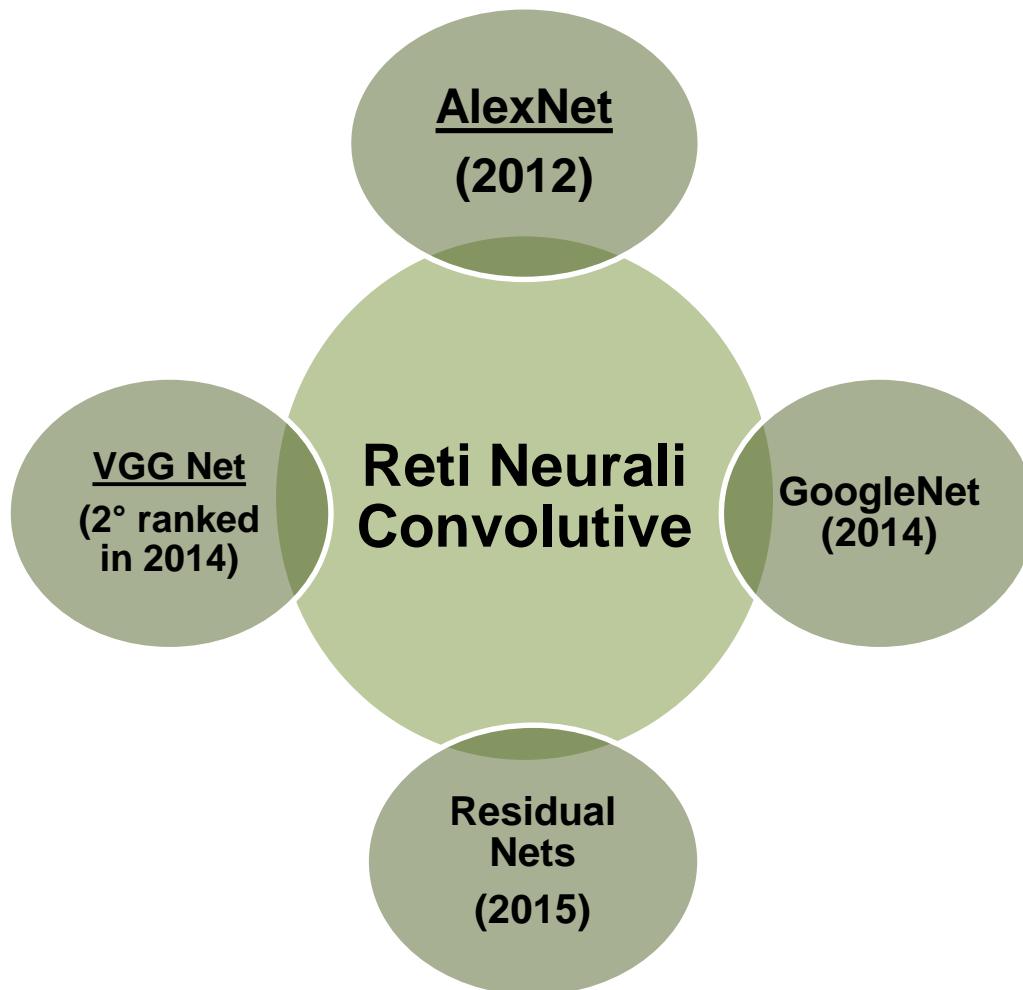


Reti Neurali Convolute (ConvNet or CNN)

- **CNNs** were initially devised for Image Recognition, nowadays very often reach better-than-human accuracy
- **CNNs** need to be fed with images, but since for a machine images are just numeric matrices...
- ...they are increasingly being used in Natural Language Processing, e.g. text classification, with excellent results



Reti Neurali «Deep» di successo per Computer Vision (fino al 2015)



Problemi tradizionali legati agli strati Convolutivi:

- Strati convoluzionali molto grandi possono determinare il problema dell'**overfitting** e **vanishing gradient** a causa dei limiti del Solver (SGD, Adam, etc).

Model Regularization :

- **Dropout** – Co-adattamento ed Ensemble di modelli. (**Srivastava, N. et al., 2014**).
- **Weight penalty L1/L2**
- **Data Augmentation** (crop, flip, rotation, ZCA whitening, etc.)

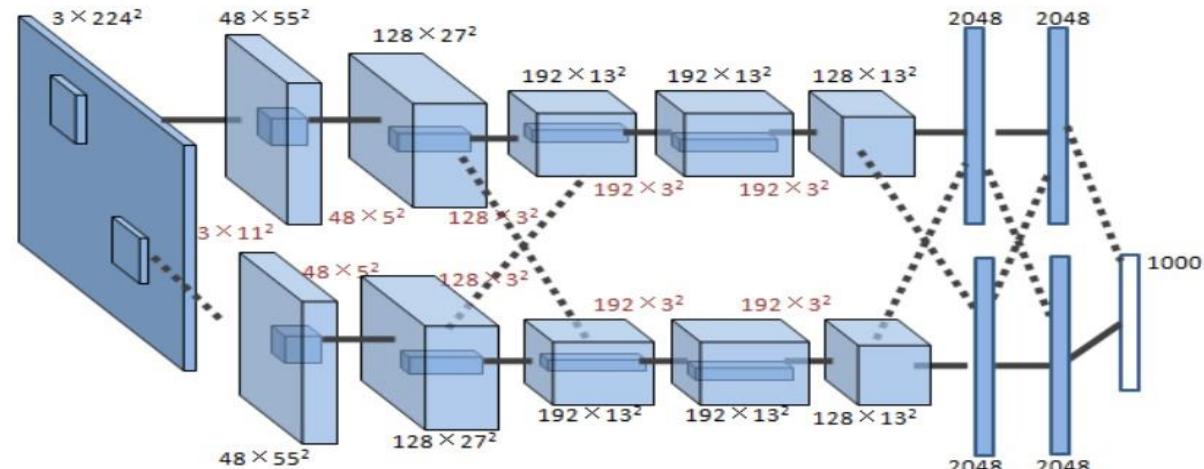
AlexNet – Università di Toronto (Krizhevsky, A. et al, 2012)

Caratteristiche Principali

- **8 strati addestrabili:** 5 strati convolutivi e 3 strati completamente connessi.
- **Max pooling layers** dopo primo, secondo e quinto strato.
- **Rectified Linear Units (ReLUs)** (Nair, V., & Hinton, G. E. 2010).
- **Local Response Normalization.**
- **60 milioni di parametri liberi (pesi sinaptici e bias), 650 migliaia di neuroni.**
- **Metodi di regolarizzazione del modello:** Dropout (probabilità 50% dopo i primi due strati completamente connessi, Data Augmentation (traslazioni, riflessioni orizzontali, PCA su RGB)).
- **Addestrato su 2 GTX 580 3 GB GPUs.**

Risultati:

- **1 CNN: 40.7% Top-1 Error, 18.2% Top-5 Error**
- **5 CNNs: 38.1% Top-1 Error, 16.4% Top-5 Error**
- **SIFT+FVs: 26.2% Top-5 Error (Sánchez, J., et al., 2013).**



AlexNet in Keras

```
def get_alexnet(input_shape,nb_classes,mean_flag):
    # code adapted from https://github.com/heuritech/convnets-keras

    inputs = Input(shape=input_shape)

    if mean_flag:
        mean_subtraction = Lambda(mean_subtract, name='mean_subtraction')(inputs)
        conv_1 = Convolution2D(96, 11, 11,subsample=(4,4),activation='relu',
                              name='conv_1', init='he_normal')(mean_subtraction)
    else:
        conv_1 = Convolution2D(96, 11, 11,subsample=(4,4),activation='relu',
                              name='conv_1', init='he_normal')(inputs)

    conv_2 = MaxPooling2D((3, 3), strides=(2,2))(conv_1)
    conv_2 = crosschannelnormalization(name="convpool_1")(conv_2)
    conv_2 = ZeroPadding2D((2,2))(conv_2)
    conv_2 = merge([
        Convolution2D(128,5,5,activation="relu",init='he_normal', name='conv_2_'+str(i+1))(
            splitensor(ratio_split=2,id_split=i)(conv_2)
        ) for i in range(2)], mode='concat',concat_axis=1,name="conv_2")

    conv_3 = MaxPooling2D((3, 3), strides=(2, 2))(conv_2)
    conv_3 = crosschannelnormalization()(conv_3)
    conv_3 = ZeroPadding2D((1,1))(conv_3)
    conv_3 = Convolution2D(384,3,3,activation='relu',name='conv_3',init='he_normal')(conv_3)
```

AlexNet in Keras

```
conv_4 = ZeroPadding2D((1,1))(conv_3)
conv_4 = merge([
    Convolution2D(192,3,3,activation="relu", init='he_normal', name='conv_4_'+str(i+1))(
        splitensor(ratio_split=2,id_split=i)(conv_4)
    ) for i in range(2)], mode='concat',concat_axis=1,name="conv_4")

conv_5 = ZeroPadding2D((1,1))(conv_4)
conv_5 = merge([
    Convolution2D(128,3,3,activation="relu",init='he_normal', name='conv_5_'+str(i+1))(
        splitensor(ratio_split=2,id_split=i)(conv_5)
    ) for i in range(2)], mode='concat',concat_axis=1,name="conv_5")

dense_1 = MaxPooling2D((3, 3), strides=(2,2),name="convpool_5")(conv_5)

dense_1 = Flatten(name="flatten")(dense_1)
dense_1 = Dense(4096, activation='relu',name='dense_1',init='he_normal')(dense_1)
dense_2 = Dropout(0.5)(dense_1)
dense_2 = Dense(4096, activation='relu',name='dense_2',init='he_normal')(dense_2)
dense_3 = Dropout(0.5)(dense_2)
dense_3 = Dense(nb_classes,name='dense_3_new',init='he_normal')(dense_3)

prediction = Activation("softmax",name="softmax")(dense_3)

alexnet = Model(input=inputs, output=prediction)

return alexnet
```

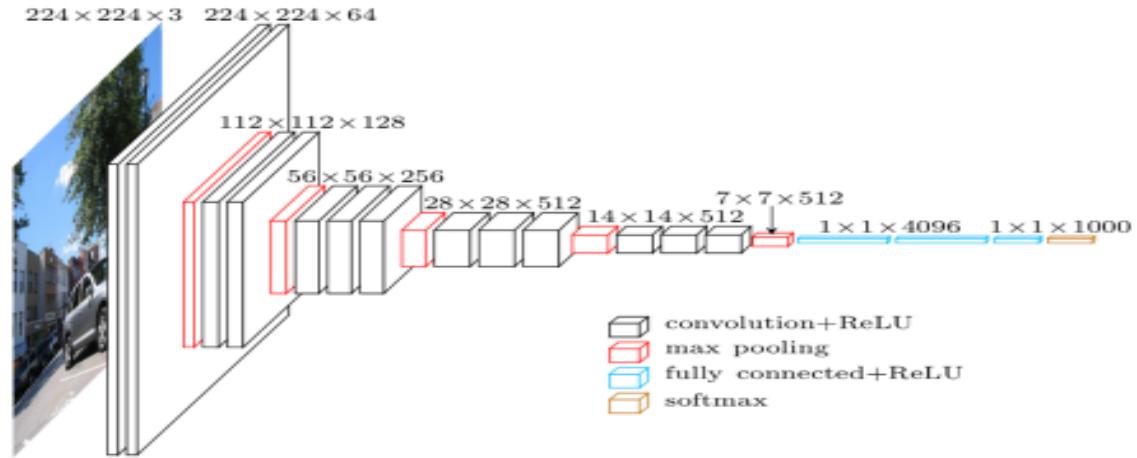
VGG-Net – Università di Oxford (Simonyan, K., & Zisserman, A., 2014)

Caratteristiche Principali:

- **Filtri (Kernel) con piccolo campi recettivi:** 3x3 che è la dimensione minima di un campo recettivo per catturare il concetto di sinistra/destra, su/giù e centro. E' facile vedere che una pila di due strati convolutivi con campo recettivo 3x3 (senza pooling spaziale tra i due) corrispondono ad un solo strato convolitivo con campo recettivo effettivo di 5x5, e via dicendo.
- Avere **Campi Recettivi piccoli** è un modo per incrementare la non-linearietà della **funzione di decisione**.
- **Architetture di profondità crescente:** VGG-16 (2xConv3-64, 2xConv3-128, 3xConv3-256, 6xConv3-512, 3xFC), VGG-19 (identico a VGG-16 ma con 8xConv3-512).
- **Vantaggio:** topologia meno complessa, supera GoogleNet nell'accuratezza della classificazione a singola rete
- **Svantaggio: 138 milioni di parametri solo per la VGG-16!!**

Risultati:

- **Modello Multi-Rete Convolutiva:** (D/[256;512]/256,384,512), (E/[256;512]/256,384,512), multi-crop & dense eval: **23.7% Top-1 Error, 6.8% Top-5 Error.**



VggNet in Keras

```
class VGG_16:  
  
    @staticmethod  
    def build(width, height, depth, classes, mul_factor, summary, weightsPath=None):  
  
        model = Sequential()  
        model.add(ZeroPadding2D((1,1),input_shape=(depth, height, width)))  
        model.add(Convolution2D(64, 3, 3, activation='relu'))  
        model.add(ZeroPadding2D((1,1)))  
        model.add(Convolution2D(64, 3, 3, activation='relu'))  
        model.add(MaxPooling2D((2,2), strides=(2,2)))  
  
        model.add(ZeroPadding2D((1,1)))  
        model.add(Convolution2D(128, 3, 3, activation='relu'))  
        model.add(ZeroPadding2D((1,1)))  
        model.add(Convolution2D(128, 3, 3, activation='relu'))  
        model.add(MaxPooling2D((2,2), strides=(2,2)))  
  
        model.add(ZeroPadding2D((1,1)))  
        model.add(Convolution2D(256, 3, 3, activation='relu'))  
        model.add(ZeroPadding2D((1,1)))  
        model.add(Convolution2D(256, 3, 3, activation='relu'))  
        model.add(ZeroPadding2D((1,1)))  
        model.add(Convolution2D(256, 3, 3, activation='relu'))  
        model.add(MaxPooling2D((2,2), strides=(2,2)))
```

VggNet in Keras

```
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))

model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))

model.add(Flatten())
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(classes, activation='softmax'))

if summary==True:
    model.summary()

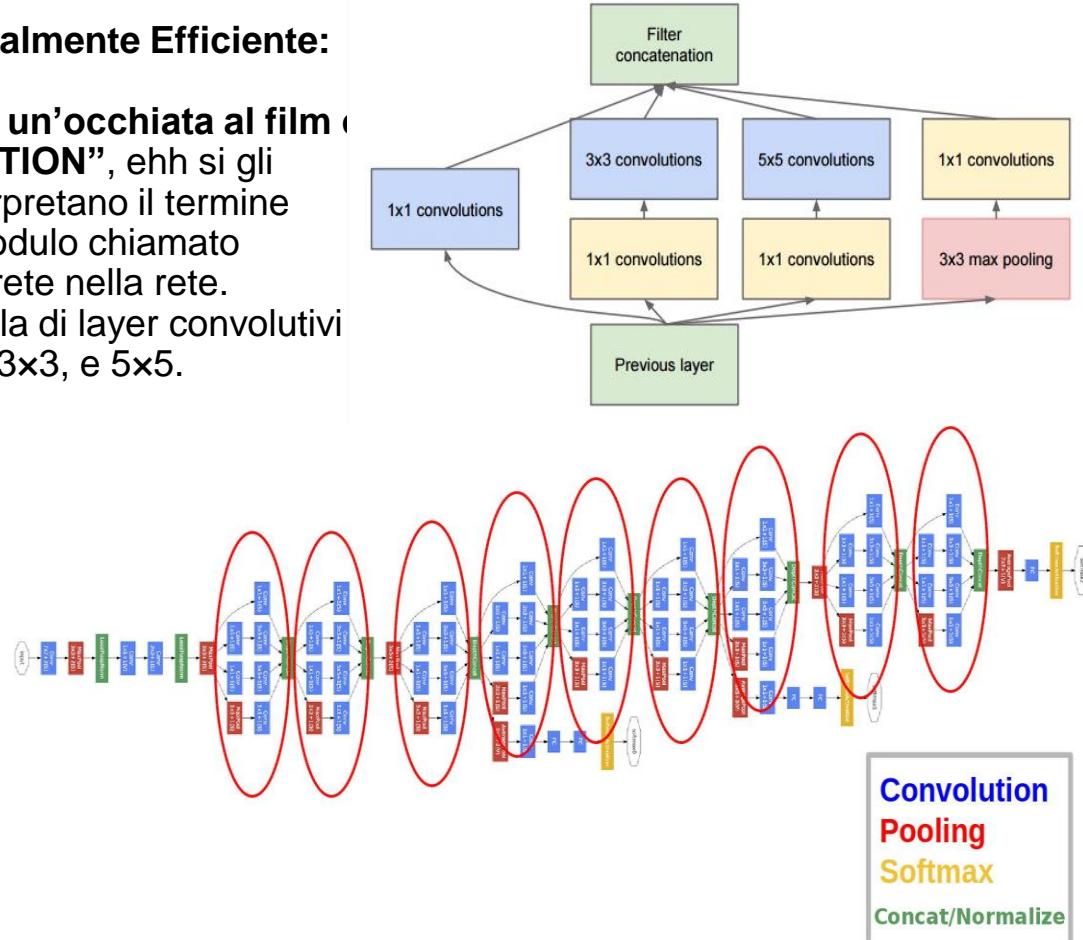
if weightsPath:
    model.load_weights(weightsPath)

return model
```

GoogleNet (Inception V1) – Google (Szegedy, C., et al., 2015)

Caratteristiche Principali:

- **Architettura profonda Compazionalmente Efficiente:**
22 strati
- **Perchè il nome “Inception”?** Date un’occhiata al film Christopher Nolan intitolato “INCEPTION”, ehh si gli informatici sono strani!! Alcuni interpretano il termine inception come ad indicare che il modulo chiamato appunto inception rappresenta una rete nella rete.
- Inception: è la combinazione parallela di layer convolutivi con filtri di diverse dimensioni: 1×1 , 3×3 , e 5×5 .
- **Strato Bottleneck:** La grande intuizione del modulo inception è l’uso di blocchi convolutivi 1×1 per ridurre il numero di feature prima dei blocchi espansivi paralleli.
- **Vantaggio:** 4 milioni di parametri!
- **Svantaggio:** Non scalabile!



Risultati:

- **Ensemble di 7 Modelli:**
6.67% Top-5 Error.

GoogleNet in Keras

```
class GoogleNet:

    @staticmethod
    def build(width, height, depth, classes, mul_factor, weightsPath=None):
        input = Input(shape=(depth, height, width)) # Set Input Shape
        conv1_7x7_s2 = Convolution2D(64, 7, 7, subsample=(2, 2), border_mode='same', activation='relu', name='conv1/7x7_s2', W_regularizer=l2(0.0002))(input)
        conv1_zero_pad = ZeroPadding2D(padding=(1, 1))(conv1_7x7_s2)
        pool1_helper = PoolHelper()(conv1_zero_pad)
        pool1_3x3_s2 = MaxPooling2D(pool_size=(3, 3), strides=(2, 2), border_mode='valid', name='pool1/3x3_s2')(pool1_helper)
        pool1_norml = LRN(name='pool1/norml')(pool1_3x3_s2)
        conv2_3x3_reduce = Convolution2D(64, 1, 1, border_mode='same', activation='relu', name='conv2/3x3_reduce', W_regularizer=l2(0.0002))(pool1_norml)
        conv2_3x3 = Convolution2D(192, 3, 3, border_mode='same', activation='relu', name='conv2/3x3', W_regularizer=l2(0.0002))(conv2_3x3_reduce)
        conv2_norm2 = LRN(name='conv2/norm2')(conv2_3x3)
        conv2_zero_pad = ZeroPadding2D(padding=(1, 1))(conv2_norm2)
        pool2_helper = PoolHelper()(conv2_zero_pad)
        pool2_3x3_s2 = MaxPooling2D(pool_size=(3, 3), strides=(2, 2), border_mode='valid', name='pool2/3x3_s2')(pool2_helper)

        # First Inception Module
        inception_3a_lx1 = Convolution2D(64, 1, 1, border_mode='same', activation='relu', name='inception_3a/lx1', W_regularizer=l2(0.0002))(pool2_3x3_s2)
        inception_3a_3x3_reduce = Convolution2D(96, 1, 1, border_mode='same', activation='relu', name='inception_3a/3x3_reduce', W_regularizer=l2(0.0002))(pool2_3x3_s2)
        inception_3a_3x3 = Convolution2D(128, 3, 3, border_mode='same', activation='relu', name='inception_3a/3x3', W_regularizer=l2(0.0002))(inception_3a_3x3_reduce)
        inception_3a_5x5_reduce = Convolution2D(16, 1, 1, border_mode='same', activation='relu', name='inception_3a/5x5_reduce', W_regularizer=l2(0.0002))(pool2_3x3_s2)
        inception_3a_5x5 = Convolution2D(32, 5, 5, border_mode='same', activation='relu', name='inception_3a/5x5', W_regularizer=l2(0.0002))(inception_3a_5x5_reduce)
        inception_3a_pool = MaxPooling2D(pool_size=(3, 3), strides=(1, 1), border_mode='same', name='inception_3a/pool')(pool2_3x3_s2)
        inception_3a_pool_proj = Convolution2D(32, 1, 1, border_mode='same', activation='relu', name='inception_3a/pool_proj', W_regularizer=l2(0.0002))(inception_3a_pool)
        inception_3a_output = merge([inception_3a_lx1, inception_3a_3x3, inception_3a_5x5, inception_3a_pool_proj], mode='concat', concat_axis=1, name='inception_3a/output')
```

GoogleNet in Keras

```
# Second Inception Module
inception_3b_1x1 = Convolution2D(128,1,1,border_mode='same',activation='relu',name='inception_3b/1x1',W_regularizer=l2(0.0002))(inception_3a_output)
inception_3b_3x3_reduce = Convolution2D(128,1,1,border_mode='same',activation='relu',name='inception_3b/3x3_reduce',W_regularizer=l2(0.0002))(inception_3a_output)
inception_3b_3x3 = Convolution2D(192,3,3,border_mode='same',activation='relu',name='inception_3b/3x3',W_regularizer=l2(0.0002))(inception_3b_3x3_reduce)
inception_3b_5x5_reduce = Convolution2D(32,1,1,border_mode='same',activation='relu',name='inception_3b/5x5_reduce',W_regularizer=l2(0.0002))(inception_3a_output)
inception_3b_5x5 = Convolution2D(96,5,5,border_mode='same',activation='relu',name='inception_3b/5x5',W_regularizer=l2(0.0002))(inception_3b_5x5_reduce)
inception_3b_pool = MaxPooling2D(pool_size=(3,3),strides=(1,1),border_mode='same',name='inception_3b/pool')(inception_3a_output)
inception_3b_pool_proj = Convolution2D(64,1,1,border_mode='same',activation='relu',name='inception_3b/pool_proj',W_regularizer=l2(0.0002))(inception_3b_pool)
inception_3b_output = merge([inception_3b_1x1,inception_3b_3x3,inception_3b_5x5,inception_3b_pool_proj],mode='concat',concat_axis=1,name='inception_3b/output')
inception_3b_output_zero_pad = ZeroPadding2D(padding=(1, 1))(inception_3b_output)
pool3_helper = PoolHelper()(inception_3b_output_zero_pad)
pool3_3x3_s2 = MaxPooling2D(pool_size=(3,3),strides=(2,2),border_mode='valid',name='pool3/3x3_s2')(pool3_helper)

# Third Inception Module
inception_4a_1x1 = Convolution2D(192,1,1,border_mode='same',activation='relu',name='inception_4a/1x1',W_regularizer=l2(0.0002))(pool3_3x3_s2)
inception_4a_3x3_reduce = Convolution2D(96,1,1,border_mode='same',activation='relu',name='inception_4a/3x3_reduce',W_regularizer=l2(0.0002))(pool3_3x3_s2)
inception_4a_3x3 = Convolution2D(208,3,3,border_mode='same',activation='relu',name='inception_4a/3x3',W_regularizer=l2(0.0002))(inception_4a_3x3_reduce)
inception_4a_5x5_reduce = Convolution2D(16,1,1,border_mode='same',activation='relu',name='inception_4a/5x5_reduce',W_regularizer=l2(0.0002))(pool3_3x3_s2)
inception_4a_5x5 = Convolution2D(48,5,5,border_mode='same',activation='relu',name='inception_4a/5x5',W_regularizer=l2(0.0002))(inception_4a_5x5_reduce)
inception_4a_pool = MaxPooling2D(pool_size=(3,3),strides=(1,1),border_mode='same',name='inception_4a/pool')(pool3_3x3_s2)
inception_4a_pool_proj = Convolution2D(64,1,1,border_mode='same',activation='relu',name='inception_4a/pool_proj',W_regularizer=l2(0.0002))(inception_4a_pool)
inception_4a_output = merge([inception_4a_1x1,inception_4a_3x3,inception_4a_5x5,inception_4a_pool_proj],mode='concat',concat_axis=1,name='inception_4a/output')
lossl_ave_pool = AveragePooling2D(pool_size=(5,5),strides=(3,3),name='lossl/ave_pool')(inception_4a_output)
lossl_conv = Convolution2D(128,1,1,border_mode='same',activation='relu',name='lossl/conv',W_regularizer=l2(0.0002))(lossl_ave_pool)
lossl_flat = Flatten()(lossl_conv)
lossl_fc = Dense(1024,activation='relu',name='lossl/fc',W_regularizer=l2(0.0002))(lossl_flat)
lossl_drop_fc = Dropout(0.7)(lossl_fc)
lossl_classifier = Dense(1000,name='lossl/classifier',W_regularizer=l2(0.0002))(lossl_drop_fc)
lossl_classifier_act = Activation('softmax')(lossl_classifier)
```

GoogleNet in Keras

```
# Fourth Inception Module
inception_4b_1x1 = Convolution2D(160,1,1,border_mode='same',activation='relu',name='inception_4b/1x1',W_regularizer=l2(0.0002))(inception_4a_output)
inception_4b_3x3_reduce = Convolution2D(112,1,1,border_mode='same',activation='relu',name='inception_4b/3x3_reduce',W_regularizer=l2(0.0002))(inception_4a_output)
inception_4b_3x3 = Convolution2D(224,3,3,border_mode='same',activation='relu',name='inception_4b/3x3',W_regularizer=l2(0.0002))(inception_4b_3x3_reduce)
inception_4b_5x5_reduce = Convolution2D(24,1,1,border_mode='same',activation='relu',name='inception_4b/5x5_reduce',W_regularizer=l2(0.0002))(inception_4a_output)
inception_4b_5x5 = Convolution2D(64,5,5,border_mode='same',activation='relu',name='inception_4b/5x5',W_regularizer=l2(0.0002))(inception_4b_5x5_reduce)
inception_4b_pool = MaxPooling2D(pool_size=(3,3),strides=(1,1),border_mode='same',name='inception_4b/pool')(inception_4a_output)
inception_4b_pool_proj = Convolution2D(64,1,1,border_mode='same',activation='relu',name='inception_4b/pool_proj',W_regularizer=l2(0.0002))(inception_4b_pool)
inception_4b_output = merge([inception_4b_1x1,inception_4b_3x3,inception_4b_5x5,inception_4b_pool_proj],mode='concat',concat_axis=1,name='inception_4b_output')

model = Model(input=input, output=[lossl_classifier])

model.summary()

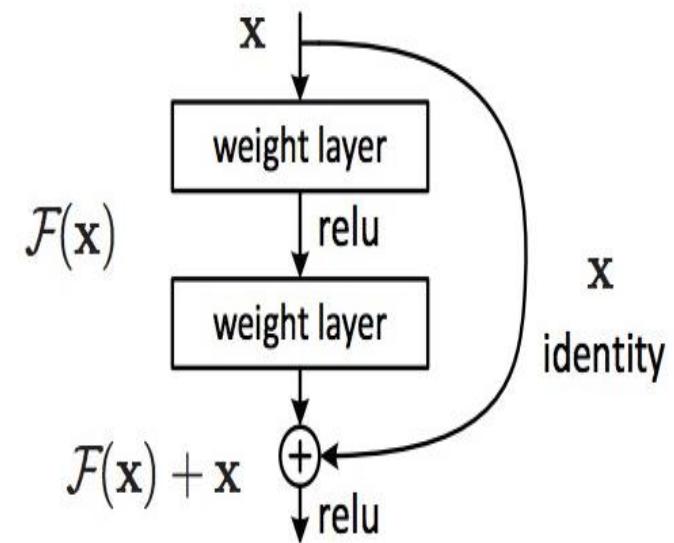
# if a weights path is supplied (indicating that the model was pre-trained), then load the weights
if weightsPath is not None:
    model.load_wights(weightsPath)

return model
```

Reti Neurali Residuali – Microsoft (He, K., et al., 2016)

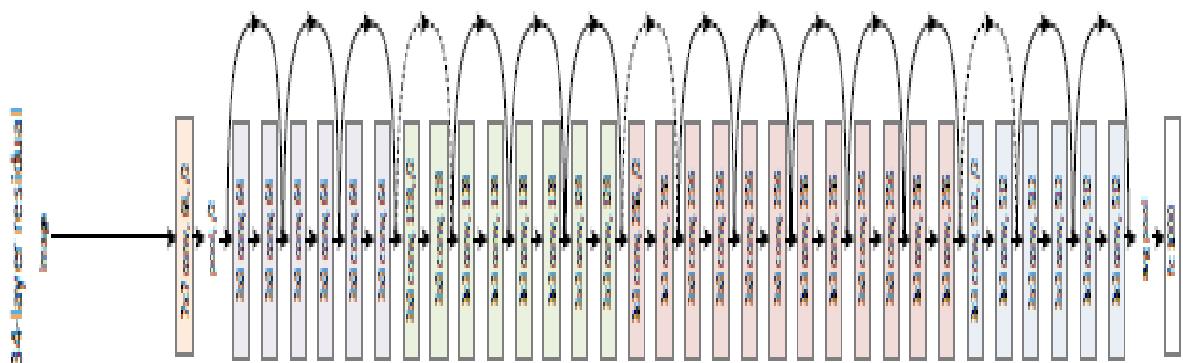
Caratteristiche Principali:

- **Problema della Degradazione:** Impilando sempre più strati **NON E'** meglio. Con la profondità della rete crescente, l'accuratezza diviene saturata e degrada rapidamente! E' un problema di tutti i solver conosciuti.
- **La ResNet risolve il “Problema della Degradazione”:** fissando un mapping residuale che è più facile da ottimizzare.
- **Shortcut connections:** connessioni residuali
- **Very deep architecture:** up to 1202 layers with WideResnet with only **19.4 million parameters!**
- **Vantaggio: Increasing accuracy with more depth**
- **Downside: They don't consider other architectures breakthroughs.**



Results:

- **ResNet : 3.57% Top-5 Error.**
- **CNNs** show superhuman abilities at Image Recognition!
5% Human estimated Top-5 error (Johnson, R. C., 2015).



ResNet in Keras

```
class WideResNet:

    @staticmethod
    def build(width, height, depth, classes, summary, weightsPath=None):
        n = 8 # depth = 6*n + 4
        k = 4 # widen factor

        img_input = Input(shape=(depth, height, width))

        # one conv at the beginning (spatial size: 32x32)
        x = ZeroPadding2D((1, 1))(img_input)
        x = Conv2D(16, (3, 3))(x)

        # Stage 1 (spatial size: 32x32)
        x = bottleneck(x, n, 16, 16 * k, dropout=0.3, subsample=(1, 1))
        # Stage 2 (spatial size: 16x16)
        x = bottleneck(x, n, 16 * k, 32 * k, dropout=0.3, subsample=(2, 2))
        # Stage 3 (spatial size: 8x8)
        x = bottleneck(x, n, 32 * k, 64 * k, dropout=0.3, subsample=(2, 2))

        x = BatchNormalization(axis=1)(x)
        x = Activation('relu')(x)
        x = AveragePooling2D((8, 8), strides=(1, 1))(x)
        x = Flatten()(x)
        preds = Dense(classes, activation='softmax')(x)

        model = Model(inputs=img_input, outputs=preds)
        .....
        if summary==True:
            model.summary()

        #model = to_multi_gpu(model, 2)

        #if a weights path is supplied (indicating that the model was pre-trained), then load the weights
        if weightsPath is not None:
            model.load_weights(weightsPath)
        .....
        return model
```

ResNet in Keras

```
def bottleneck(incoming, count, nb_in_filters, nb_out_filters, dropout=None, subsample=(2, 2)):
    outgoing = wide_basic(incoming, nb_in_filters, nb_out_filters, dropout, subsample)
    for i in range(1, count):
        outgoing = wide_basic(outgoing, nb_out_filters, nb_out_filters, dropout, subsample=(1, 1))

    return outgoing

def wide_basic(incoming, nb_in_filters, nb_out_filters, dropout=None, subsample=(2, 2)):
    nb_bottleneck_filter = nb_out_filters

    if nb_in_filters == nb_out_filters:
        # conv3x3
        y = BatchNormalization(axis=1)(incoming)
        y = Activation('relu')(y)
        y = ZeroPadding2D((1, 1))(y)
        y = Conv2D(nb_bottleneck_filter, (3, 3), strides=subsample, kernel_initializer='he_normal', padding='valid')(y)

        # conv3x3
        y = BatchNormalization(axis=1)(y)
        y = Activation('relu')(y)
        if dropout is not None:
            y = Dropout(dropout)(y)
        y = ZeroPadding2D((1, 1))(y)
        y = Conv2D(nb_bottleneck_filter, (3, 3), strides=(1, 1), kernel_initializer='he_normal', padding='valid')(y)

        return add([incoming, y])

    else: # Residual Units for increasing dimensions
        # common BN, ReLU
        shortcut = BatchNormalization(axis=1)(incoming)
        shortcut = Activation('relu')(shortcut)

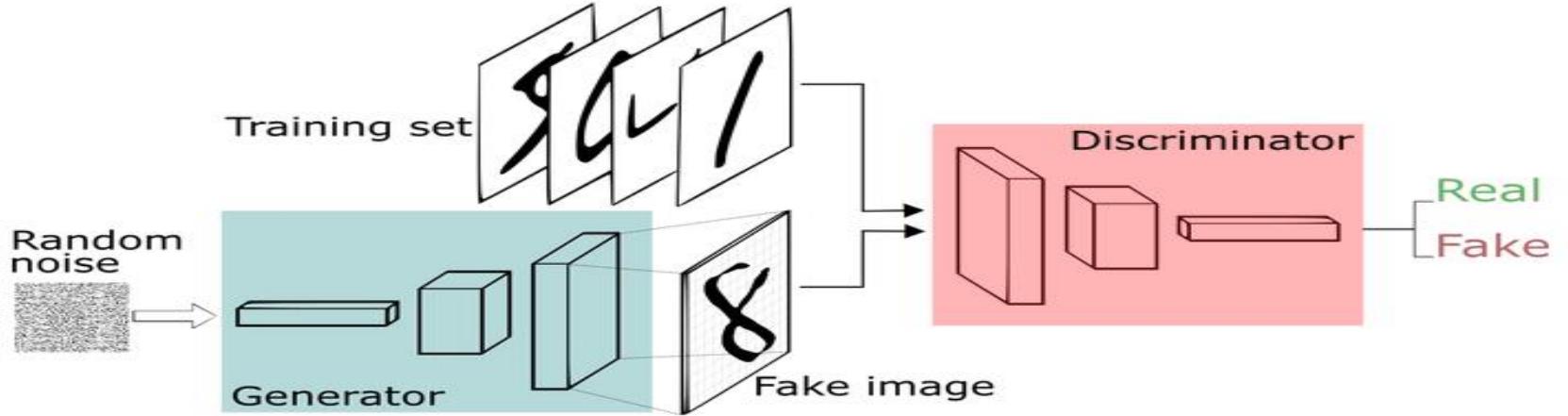
        # conv3x3
        y = ZeroPadding2D((1, 1))(shortcut)
        y = Conv2D(nb_bottleneck_filter, (3, 3), strides=subsample, kernel_initializer='he_normal', padding='valid')(y)

        # conv3x3
        y = BatchNormalization(axis=1)(y)
        y = Activation('relu')(y)
        if dropout is not None:
            y = Dropout(dropout)(y)
        y = ZeroPadding2D((1, 1))(y)
        y = Conv2D(nb_out_filters, (3, 3), strides=(1, 1), kernel_initializer='he_normal', padding='valid')(y)

        # shortcut
        shortcut = Conv2D(nb_out_filters, (1, 1), strides=subsample, kernel_initializer='he_normal', padding='same')(shortcut)

    return add([shortcut, y])
```

Generative Adversarial Networks (GAN) (Goodfellow, et al., 2014)



$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))].$$



References

Chellappa, R. (2012). Mathematical statistics and computer vision. *Image and Vision Computing*, 30(8), 467-468.

Tsukahara, J. S., Harrison, T. L., & Engle, R. W. (2016). The relationship between baseline pupil size and intelligence. *Cognitive Psychology*, 91, 109-123.

Intervista a Francesco Pugliese su Deep Learning al servizio del riciclo:
<https://www.wired.it/attualita/tech/2018/09/20/party-cloud-ibm-deep-learning/>

Acknowledgements

THANK YOU FOR YOUR ATTENTION

FRANCESCO PUGLIESE