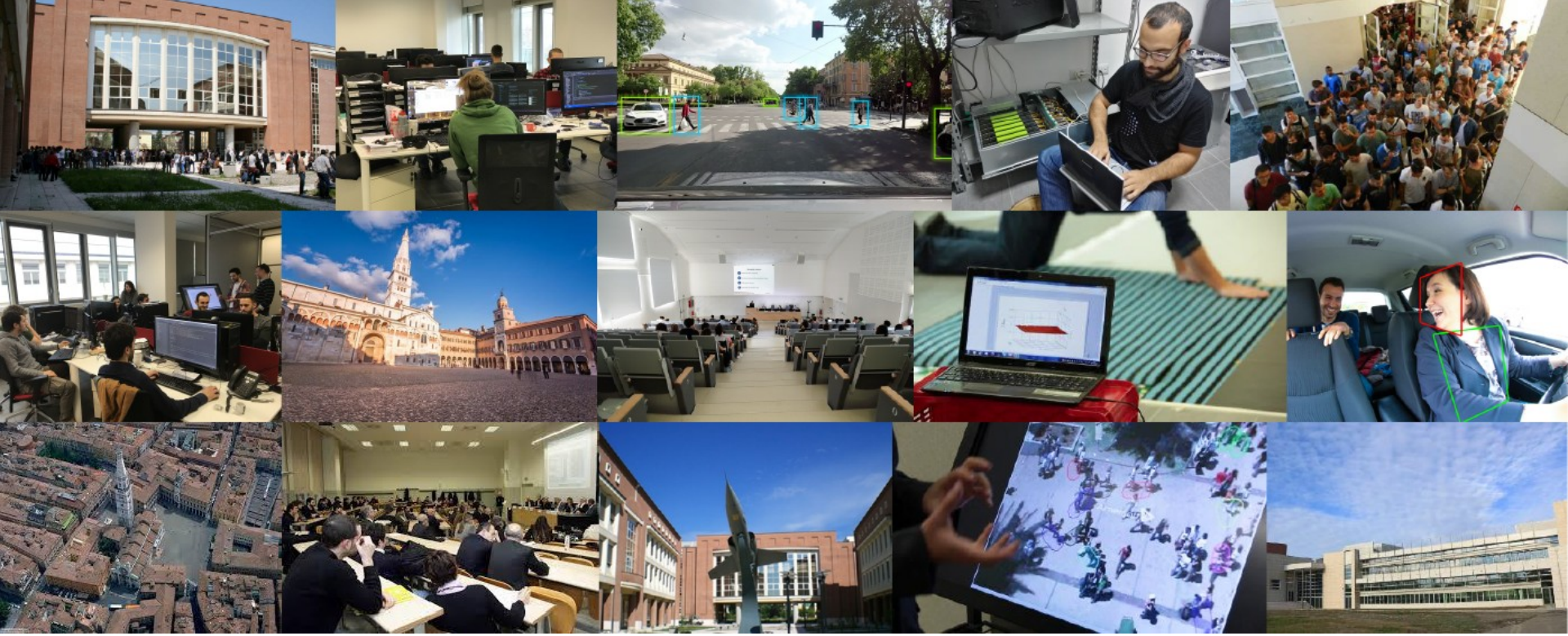


MACHINE LEARNING
- (Un’introduzione al…) Deep Learning -

AImage Lab

UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Il problema delle feature

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Durante tutto il corso abbiamo avuto a che fare con entità descritte da *features* (numeriche o categoriche)

Tipicamente, queste features sono state pensate da chi, presumibilmente *esperto del dominio*, ha creato il dataset. Esempio:

customerI	gender	SeniorCitiz	Partner	Dependen	tenure	PhoneServ	MultipleLi	InternetSe	OnlineSecu	OnlineBac	DevicePro	TechSuppc	Streaming	StreamingI	Contract	PaperlessB	PaymentM	MonthlyCh	TotalCharg	Churn
7590-VHVI	Female	0	Yes	No	1	No	No phone	DSL	No	Yes	No	No	No	No	Month-to-	Yes	Electronic	29.85	29.85	No
5575-GNV	Male	0	No	No	34	Yes	No	DSL	Yes	No	Yes	No	No	No	One year	No	Mailed che	56.95	1889.5	No
3668-QPYI	Male	0	No	No	2	Yes	No	DSL	Yes	Yes	No	No	No	No	Month-to-	Yes	Mailed che	53.85	108.15	Yes
7795-CFO(	Male	0	No	No	45	No	No phone	DSL	Yes	No	Yes	Yes	No	No	One year	No	Bank trans	42.3	1840.75	No
9237-HQIT	Female	0	No	No	2	Yes	No	Fiber optic	No	No	No	No	No	No	Month-to-	Yes	Electronic	70.7	151.65	Yes
9305-CDSI	Female	0	No	No	8	Yes	Yes	Fiber optic	No	No	Yes	No	Yes	Yes	Month-to-	Yes	Electronic	99.65	820.5	Yes
1452-KIOV	Male	0	No	Yes	22	Yes	Yes	Fiber optic	No	Yes	No	No	Yes	No	Month-to-	Yes	Credit carc	89.1	1949.4	No
6713-OKO	Female	0	No	No	10	No	No phone	DSL	Yes	No	No	No	No	No	Month-to-	No	Mailed che	29.75	301.9	No
7892-POO	Female	0	Yes	No	28	Yes	Yes	Fiber optic	No	No	Yes	Yes	Yes	Yes	Month-to-	Yes	Electronic	104.8	3046.05	Yes
6388-TABC	Male	0	No	Yes	62	Yes	No	DSL	Yes	Yes	No	No	No	No	One year	No	Bank trans	56.15	3487.95	No
9763-GRSI	Male	0	Yes	Yes	13	Yes	No	DSL	Yes	No	No	No	No	No	Month-to-	Yes	Mailed che	49.95	587.45	No

Il problema delle feature

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Si stima che circa il 65% dei dati digitali aziendali è rappresentato in forma tabulare

La specificità e la scarsità di questi dati (i dataset di solito sono composti da poche centinaia o migliaia di sample) rende, per ora, difficile l'applicazione di tecniche di Deep Learning, che hanno bisogno di dataset di training molto grandi

Viceversa, il Deep Learning è alla base dell'Intelligenza Artificiale (AI) moderna, e viene tipicamente usato in domini in cui esistono grandi quantità di dati di training e in cui le entità del dominio non sono facilmente descrivibili con delle feature "ingegnerizzate", ovvero in cui la conoscenza del dominio non è sufficiente per creare delle feature che descrivano in maniera soddisfacente le entità da rappresentare

Il problema delle feature

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Cosa succede, infatti, se, per un dato dominio, è difficile capire quali sono le feature adeguate a rappresentare le entità di quel dominio?

Esempio: quali feature uso per rappresentare un'immagine?

Il problema delle feature

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

E per rappresentare un testo scritto in “linguaggio naturale” (i.e., linguaggio umano)?

Ad sales boost Time Warner profit

Quarterly profits at US media giant TimeWarner jumped 76% to $1.13bn (£600m) for the three months to December, from $639m year-earlier.

The firm, which is now one of the biggest investors in Google, benefited from sales of high-speed internet connections and higher advert sa

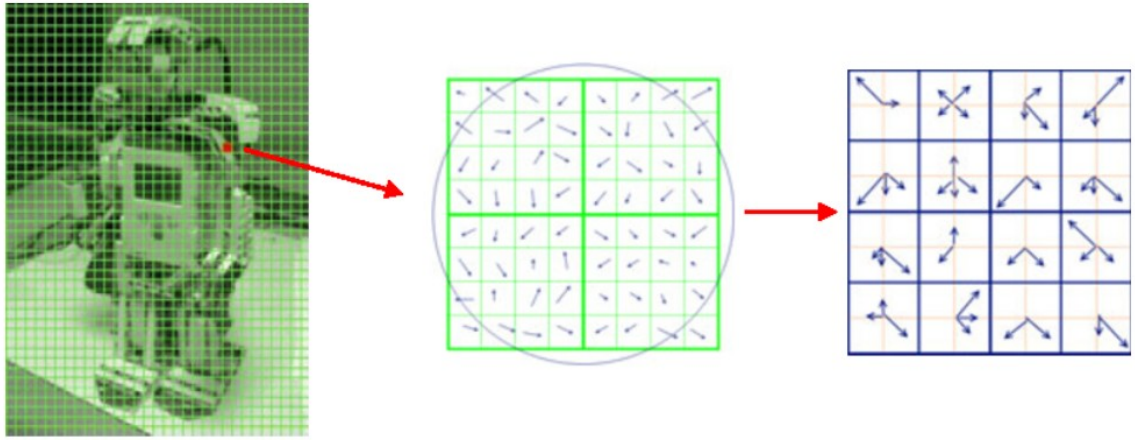
Time Warner said on Friday that it now owns 8% of search-engine Google. But its own internet business, AOL, had has mixed fortunes. It lost

Time Warner's fourth quarter profits were slightly better than analysts' expectations. But its film division saw profits slump 27% to $284m

TimeWarner is to restate its accounts as part of efforts to resolve an inquiry into AOL by US market regulators. It has already offered to

«Hand crafted» features

Per decenni la ricerca in Computer Vision, Natural Language Processing e altre aree dell'AI, si è basata sullo studio di "feature" sufficientemente significative per rappresentare entità visive, testuali, ecc.



(a) Cell Segmentation (b) Cell Image Gradients (c) Cell Descriptor

Deep Learning come Representation Learning

Con l'avvento del Deep Learning queste feature "fatte a mano" sono state accantonate, perchè si è scoperto che è molto più efficace trovare le feature più adeguate per un certo domino/task *utlizzando un processo di ottimizzazione*

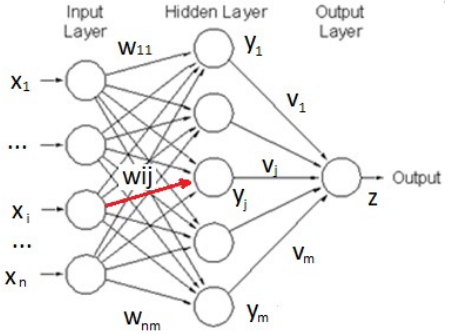
In sostanza il Deep Learning può essere visto così: usiamo il ML non solo per addestrare il modello di predizione (e.g., un classificatore o un regressore) ma anche per *ottimizzare le feature*, ovvero la *rappresentazione* delle entità del dominio

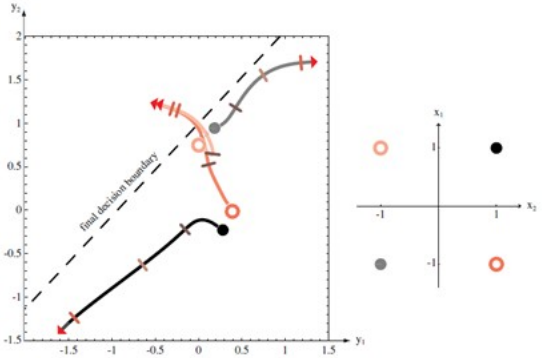
Deep Learning, infatti, è spesso considerato un sinonimo di Representation Learning, ovvero dell'uso del ML (anche) per scoprire automaticamente una rappresentazione efficace delle entità del dominio d'interesse

Deep Learning come Representation Learning

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Il principio base è quello che abbiamo visto con lo XOR problem, dove l'hidden layer di un MLP riesce, tramite Gradient Descent, a creare un nuovo feature space e un mapping tra le feature in input e la nuova rappresentazione:
y = ***ϕ(x)***





Deep Learning come Representation Learning

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

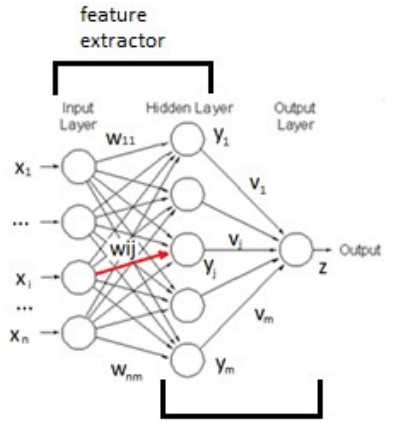
Ad esempio, quando ottimizziamo la seguente loss function:

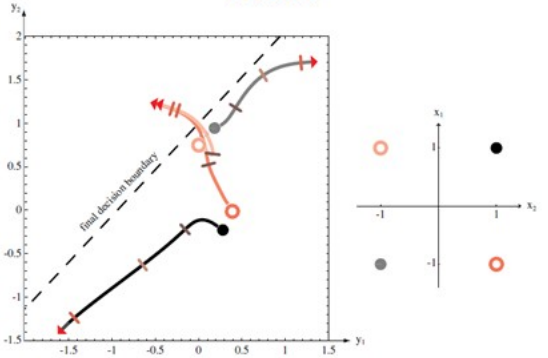
$$J(\theta) = -\sum_{i=1}^{N} t_i \log(h_\theta(\boldsymbol{x}_i)) + (1 - t_i) \log(1 - h_\theta(\boldsymbol{x}_i))$$

il modello $h_\theta(\boldsymbol{x}) = g(\boldsymbol{v}^t f(W\boldsymbol{x}))$ può essere interpretato schematicamente come composto da due parti:

- $\boldsymbol{y} = f(W\boldsymbol{x})$ è il "feature extractor", e rappresenta $\boldsymbol{x}$ nello spazio dell'hidden layer
- $g(\boldsymbol{v}^t \boldsymbol{y})$ è il "classificatore", in questo caso una logistic regression *definita sullo spazio dell'hidden layer*

Minimizzando la loss, ottimizziamo *congiuntamente* sia W che $\boldsymbol{v}$, ovvero stiamo sia addestrando il "classificatore" che cercando (*automaticamente*) la migliore rappresentazione $\boldsymbol{y} = \phi(\boldsymbol{x})$ per quel dominio e quel task specifico





Deep Learning basato sulle ANN

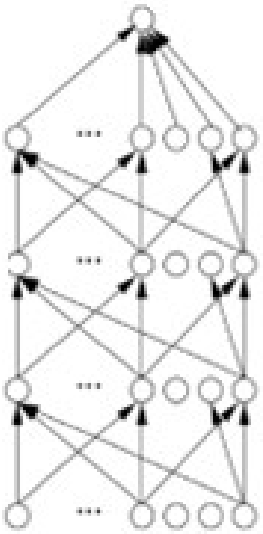
AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Il deep learning estende questo concetto già presente nel MLP e lo rafforza

Sebbene in teoria paradigmi diversi di ML potrebbero essere usati per ottimizzare la scelta delle feature, il Deep Learning è, in pratica, (quasi?) sempre basato su ANN, anche se, come vedremo, normalmente le architetture usate sono più sofisticate di un MLP

La scelta delle ANN è dovuta alla loro flessibilità, che consente di usare layer intermedi di una rete neurale per ottenere (automaticamente) la rappresentazione delle feature

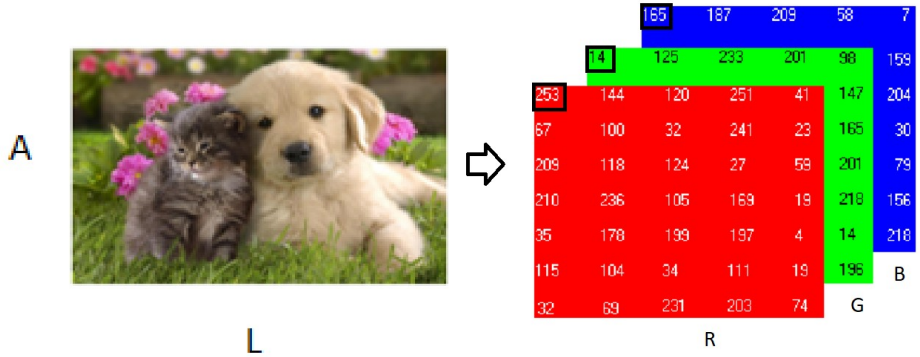
Storicamente si ritiene “profonda” (“deep”) una ANN con più di 3 layer (inclusi il layer di input e di output)



Deep Learning basato sulle ANN: Esempio

Esploriamo questo concetto seguendo un esempio basato sulle immagini

La prima osservazione è che un'immagine digitale è rappresentata con un tensore di *A X L X C* elementi, dove *A, L* sono l'altezza e la larghezza e *C* il numero di canali (colori)

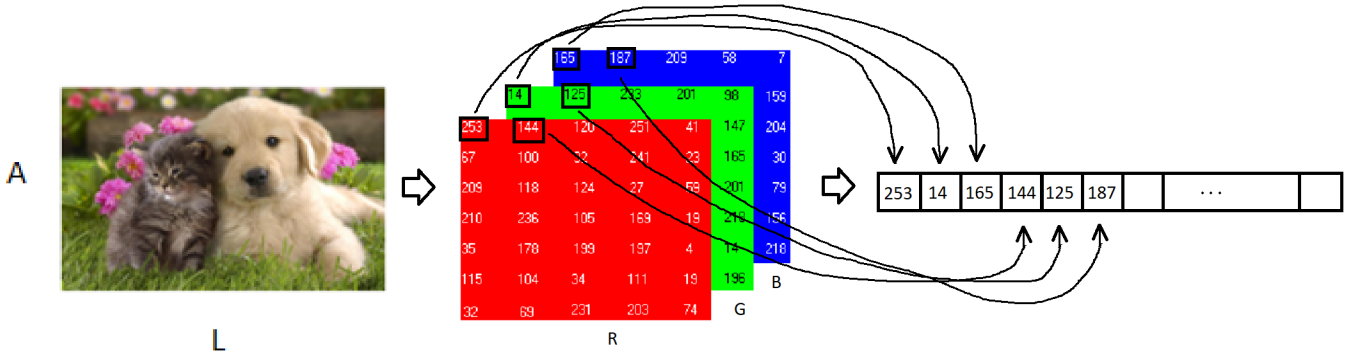


Ad ogni pixel corrispondono 3 valori reali. Ad ese., il primo pixel nella figura sopra corrisponde ai valori (incorniciati in nero): 253, 14, 165

Deep Learning basato sulle ANN: Esempio

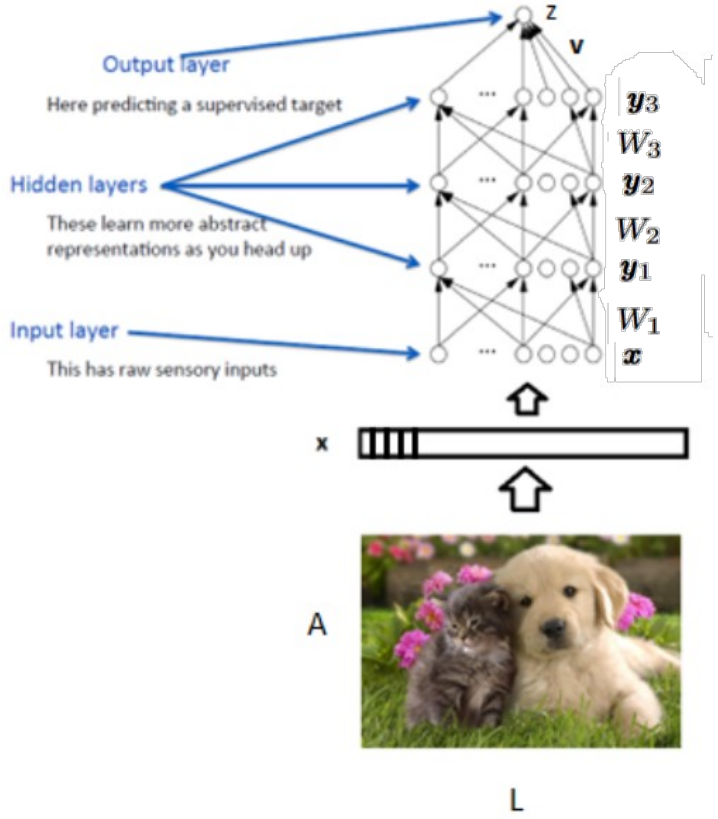
AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Posso “linearizzare” il valore di questi pixel in un vettore $\boldsymbol{x} \in R^{A \times L \times C}$



Deep Learning basato sulle ANN

- Rappresentare un'immagine usando direttamente il valore dei suoi pixel è quel che si dice usare "raw data"
- $\boldsymbol{x} \in R^{A \times L \times C}$ è l'input di un MLP con *n* hidden layer
- Il primo hidden layer calcola $\boldsymbol{y}_1 = f(W_1\boldsymbol{x})$
- Il secondo hidden layer calcola $\boldsymbol{y}_2 = f(W_2\,\boldsymbol{y}_1) = f(W_2\,(f(W_1\boldsymbol{x}))$
- …
- L'ultimo hidden layer calcola $\boldsymbol{y}_n = f(W_n\,\boldsymbol{y}_{n-1})$
- Il layer di output calcola $z = g(\boldsymbol{v}^T\,\boldsymbol{y}_n)$
- L'MLP è addestrato, ad esempio, usando la Binary Cross Entropy per distinguere immagini con e senza animali
- L'MLP imparerà sia a predire che a rappresentare (insieme!)



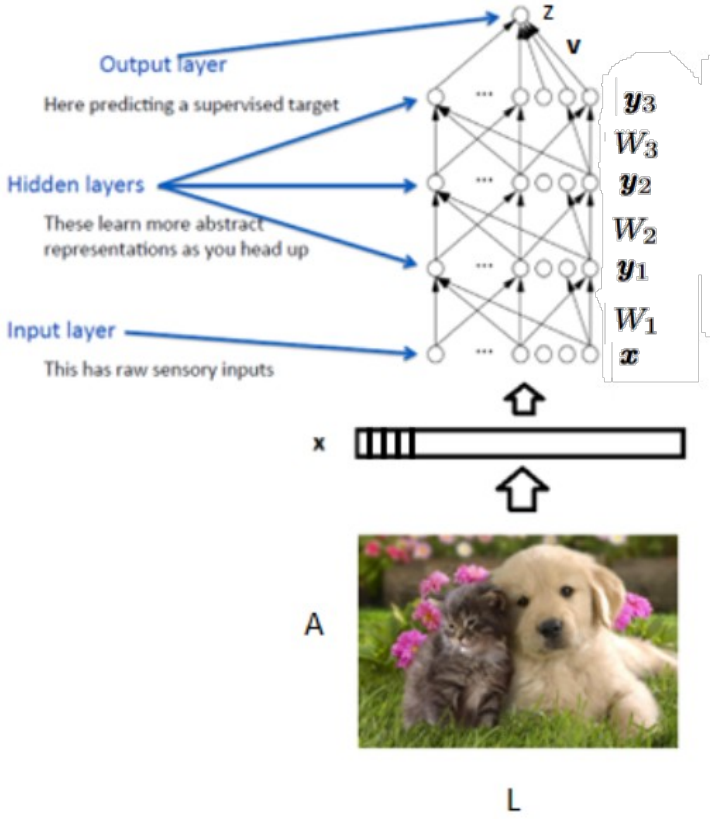
Deep Learning basato sulle ANN

I pixel rappresentano l'entità del mio domino visivo (i.e., l'immagine) in maniera "grezza", senza che siano state estratte feature pre-ingegnerizzate. Quindi rappresenta l'entità in maniera completa, senza nessuna perdita d'informazione

Quello che succede, durante il processo di ottimizzazione dell'MLP, è che le "feature grezze" (i pixel) vengono trasformate, layer dopo layer, in maniera analoga a quanto abbiamo visto per il problema dello XOR

Aspetto importante: non è l'uomo che, ingegnerizzando le feature, deve decidere che tipo di informazione estrarre e come rappresentarla, ma è il processo di ottimizzazione che guiderà questa trasformazione, scegliendo una rappresentazione ottimale per quel task

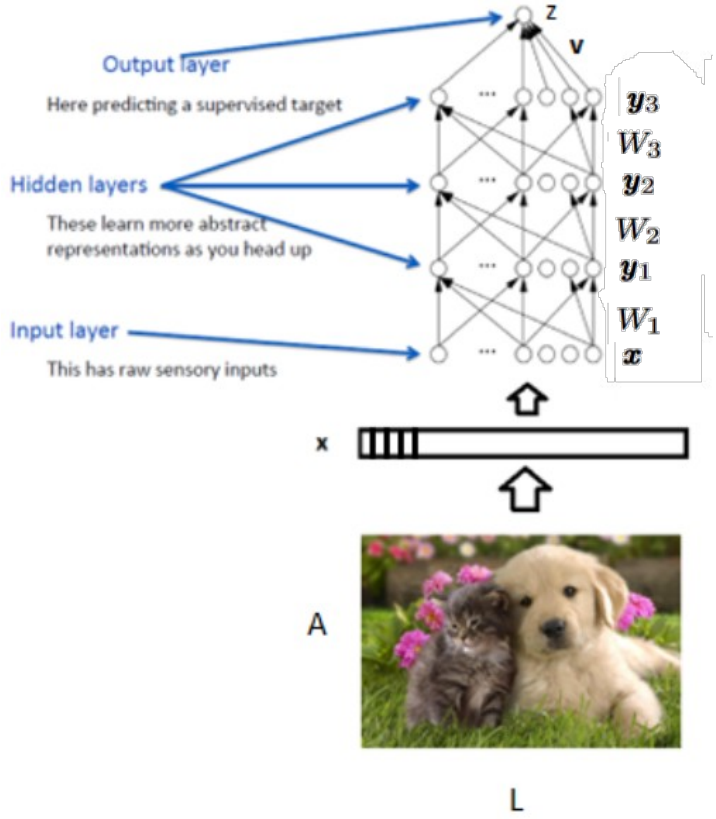
In altri termini, è il Gradient Descent che, per diminuire la loss, sceglierà le rappresentazioni intermedie più efficaci



Deep Learning basato sulle ANN

In un certo senso, questo processo ricorda un pò (vagamente) la filosofia del Wrapper, cioè il simultaneo addestramento di un classificatore insieme alla scelta delle feature ottimali *per quel classificatore*

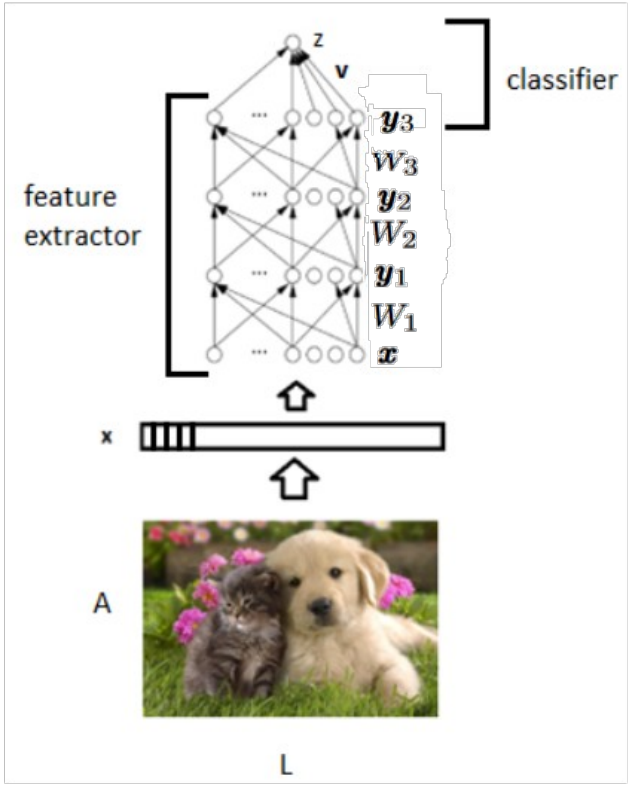
Però il Wrapper può solo *selezionare* le feature *in un insieme precostituito di feature candidate*, mentre nel Deep Learning le feature grezze, attraverso una serie di layer, possono potenzialmente essere *trasformate* in qualsiasi altro feature space



Deep Learning basato sulle ANN

Una volta addestrata, la ANN “profonda” fungerà sia da “feature extractor” che da classificatore, senza reale soluzione di continuità tra i due aspetti

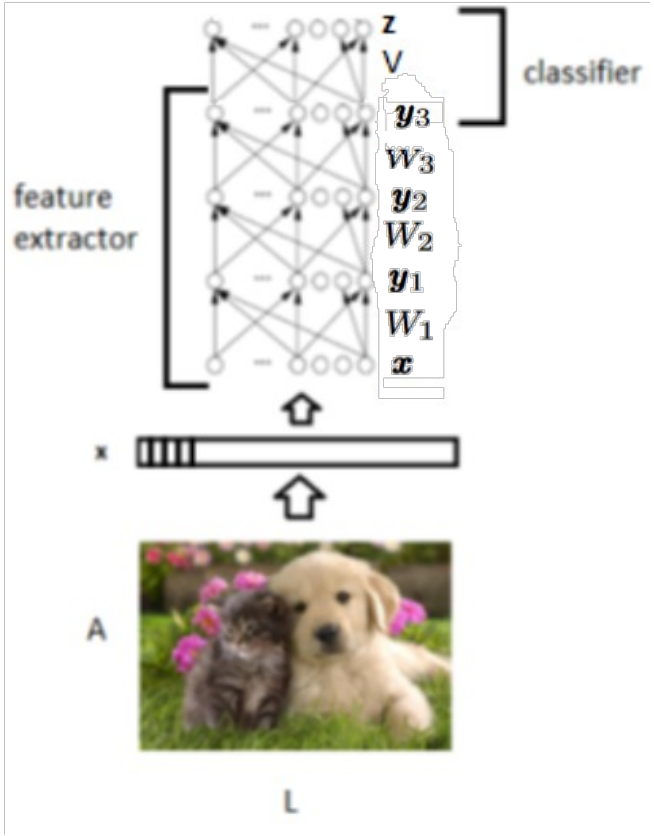
Tuttavia, di solito si ritiene che gli *n* hidden layer siano il “feature extractor” mentre il predittore (classificatore/regressore) sia un modello ($g(\boldsymbol{v}^T \boldsymbol{y}_n)$) con una score function lineare ($\boldsymbol{v}^T \boldsymbol{y}_n$) definita sull’ultimo feature space ($\boldsymbol{y}_n$)



Deep Learning basato sulle ANN

Ovviamente il layer di output (*n+1*) può essere adattato a seconda del task da risolvere (regressione/classificazione) scegliendo la nonlinearità più adeguata

Ad esempio, nel caso di classificazione multi-classe con *k* classi, l'ultimo layer avrà *k* neuroni di output $\mathbf{z} = g(V \mathbf{y}_n)$) ($\boldsymbol{z}$ vettore di *k* elementi) e *g()* sarà la funzione Softmax



Perché «Deep»?

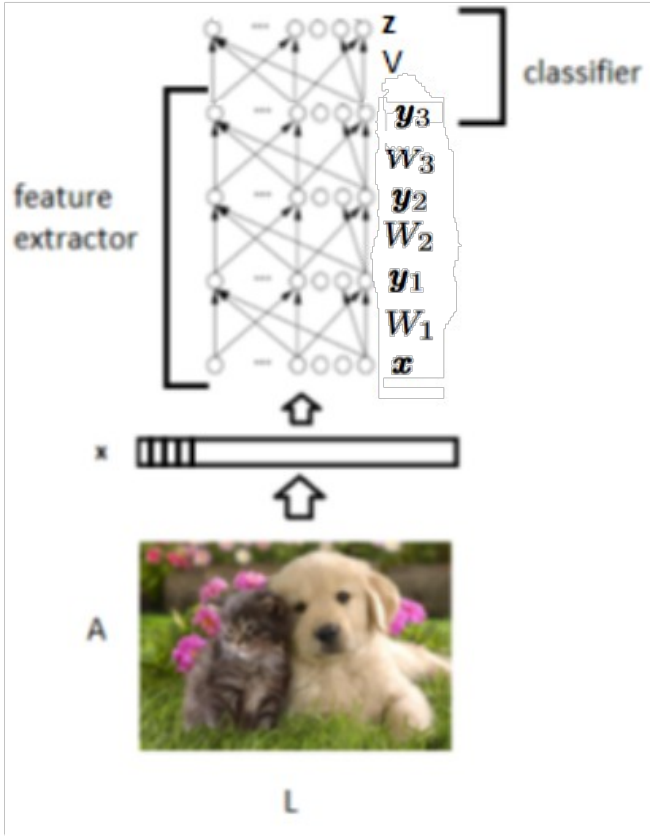
Come già accennato, storicamente, si ritiene “deep” una ANN con più di due hidden layer

Ma… perchè deve essere “deep”? Il teorema di approssimazione universale ci dice che *un solo hidden layer* è sufficiente per risolvere qualsiasi problema di ML…

Tuttavia, quel teorema non ci dice nulla riguardo al problema dell’overfitting

La profondità serve ad alleviare implicitamente il rischio di overfitting, perchè rende la rappresentazione interna della rete più *efficiente*, facendo risparmiare parametri e, quindi, training sample

Con “efficienza” qui non si intende efficienza computazionale ma il rapporto tra il numero di parametri e i training sample



Rappresentazione gerarchica della conoscenza

Analizziamo da vicino questo concetto partendo da una metafora con i sistemi di scrittura del linguaggio umano

Il nostro sistema di scrittura è *gerarchico* e *combinatorio*. Infatti, gli elementi atomici sono le lettere dell'alfabeto (A, B, C, …)

Combinando tali lettere abbiamo un primo livello di astrazione, formato dalle parole del vocabolario (CASA, STUDIARE, …)

Combinando le parole, abbiamo un secondo livello di astrazione in cui costruiamo frasi ("VADO A CASA A STUDIARE", "VOGLIO STUDIARE IL MACHINE LEARNING", …)

Se dovessimo usare un simbolo per ogni parola (come, e.g., per gli ideogrammi egizi), o addirittura un simbolo per ogni frase, avremmo bisogno di una quantità enorme di simboli per poter scrivere

In questo caso l'efficienza di rappresentazione, quindi, è data dalla possibilità di poter combinare elementi base in livelli gerarchici successivi

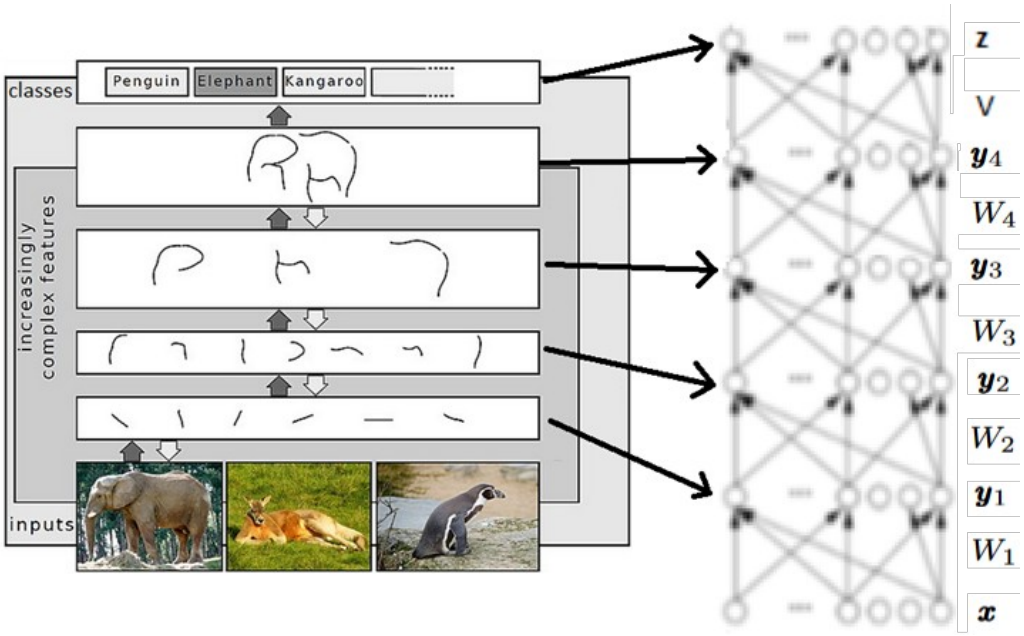
Rappresentazione gerarchica della conoscenza

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Analogamente, una rete «profonda» permette un'organizzazione *gerarchica* e *combinatoria* della rappresentazione dei dati

Più alto nella gerarchia si trova il layer, maggiore è il livello di *astrazione semantica* dell'informazione che rappresenta

Questa struttura gerarchica e le sue rappresentazioni intermedie emergono durante il processo di training (poi vedremo come e perchè…), come testimoniato da innumerevoli esperimenti negli anni passati



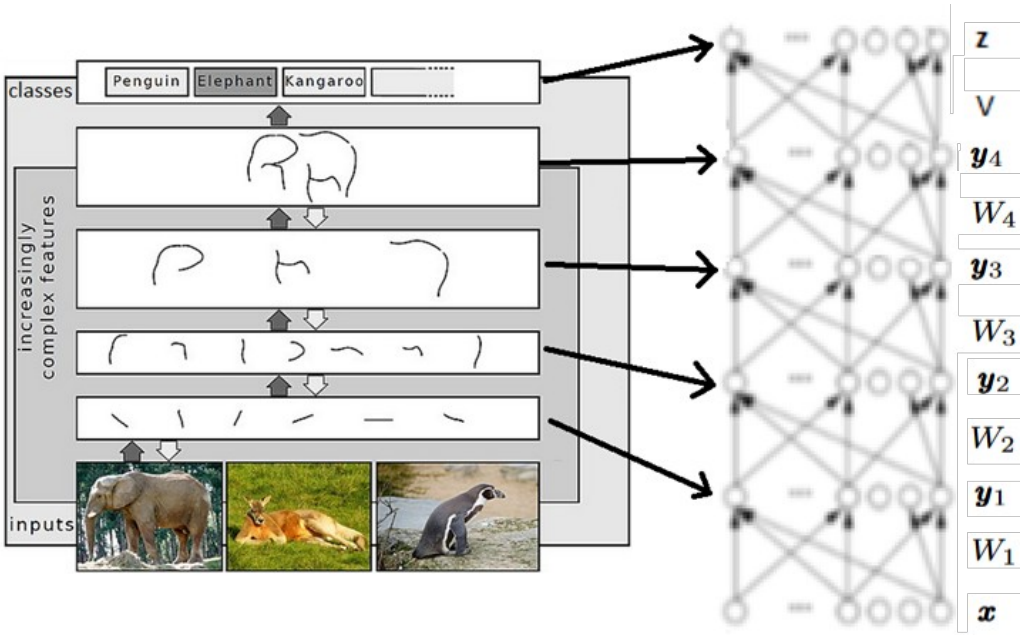
Rappresentazione gerarchica della conoscenza

I neuroni del primo hidden layer tipicamente si specializzano per riconoscere strutture semplici, come piccoli segmenti con una determinata angolazione

I neuroni del secondo layer “combinano” gli output dei neuroni del primo layer per riconoscere strutture più complesse

Nei livelli successivi i neuroni si specializzano per riconoscere parti di oggetti o addirittura oggetti interi

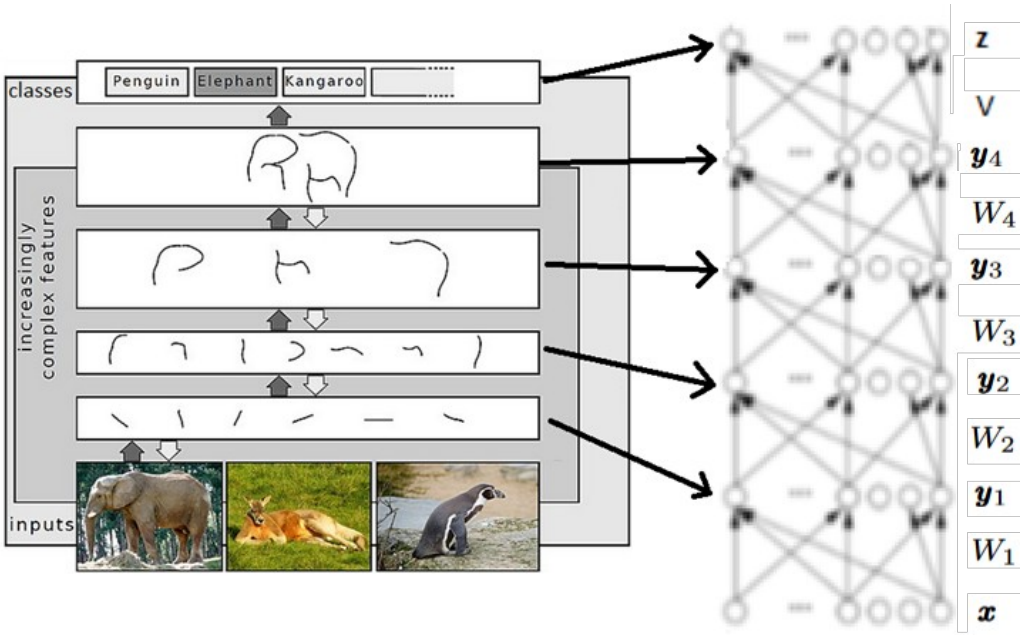
Quindi una generica scena può essere rappresentata nell’ultimo layer con una composizione gerarchica che progressivamente aggrega informazione sempre più complessa ed astratta



Rappresentazione gerarchica della conoscenza

La cosa importante da notare è che una stessa struttura o sotto-struttura visiva deve essere imparata (ovvero codificata nei parametri di qualche neurone) una volta sola e poi può essere *condivisa e riutilizzata*

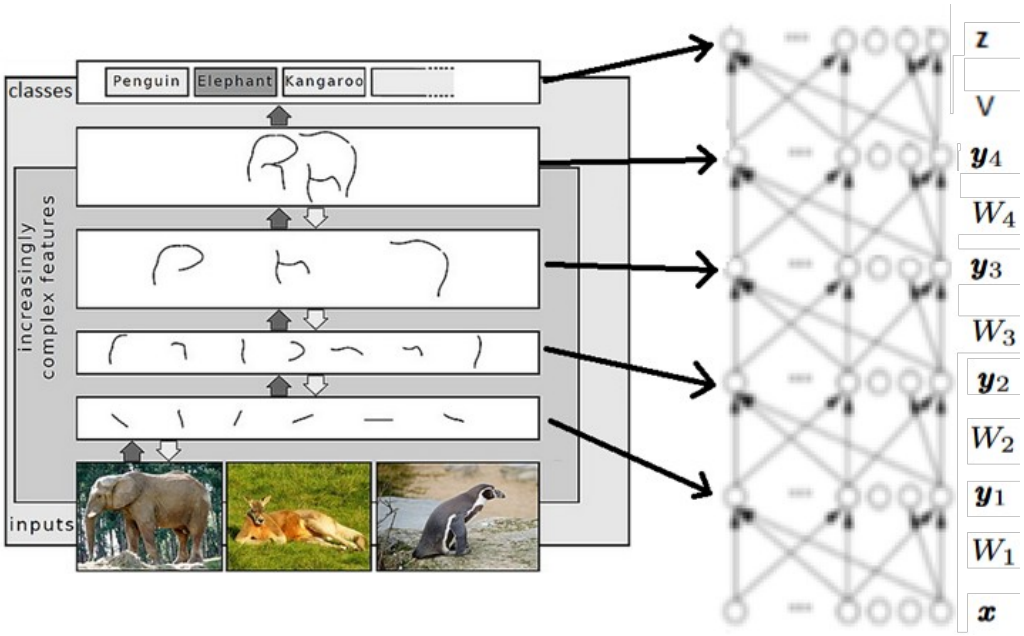
Ad esempio una linea retta verticale, riconosciuta da un neurone nel primo layer, può servire per creare linee (verticali) parallele nel secondo layer, oppure la forma di un occhio può essere usata in layer successivi sia per “comporre” il viso di un elefante che il viso di un pinguino



Rappresentazione gerarchica della conoscenza

Se invece avessimo solo un hidden layer (come negli MLP visti nella precedente lezione), non potremmo *condividere e combinare* sotto-strutture per formare strutture più complesse

Il neurone che deve riconoscere l'elefante e quello che deve riconoscere il pinguino dovrebbero *entrambi* avere dei parametri (ovvero dei pesi nelle loro connessioni con l'hidden/l'input layer) che "codificano" la struttura di un occhio, aumentando a dismisura il numero di parametri che sarebbe "sprecato" per *reimparare* sotto-concetti comuni ad altri concetti più generici

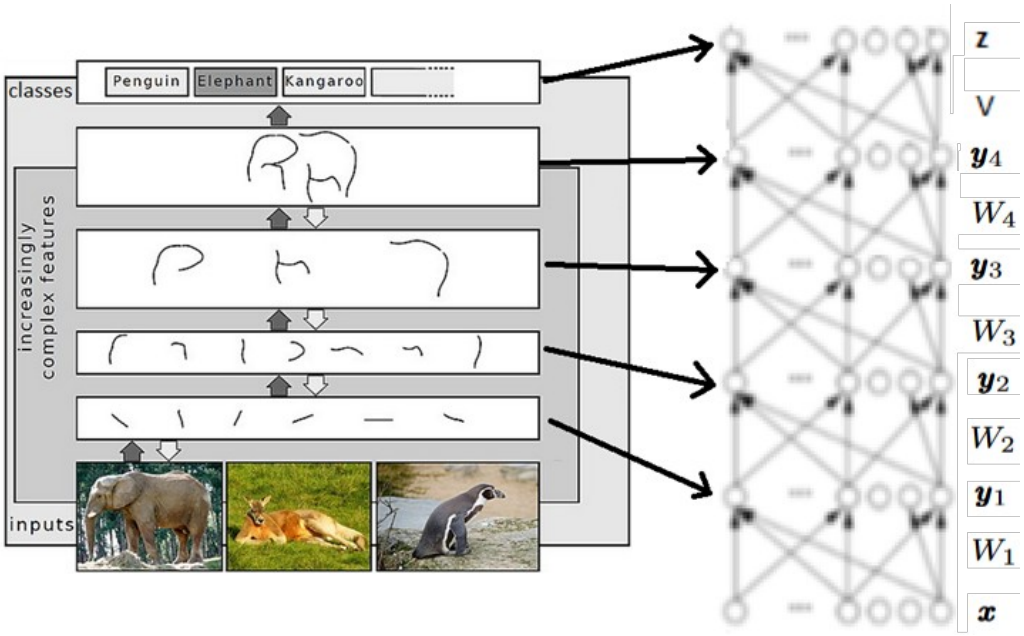


Rappresentazione gerarchica della conoscenza

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Quindi, sebbene in teoria un hidden layer sarebbe sufficiente per ottenere la stessa capacità rappresentativa di una rete profonda, quest'ultima permette di organizzare l'informazione riducendo il numero dei pesi da stimare ->

-> meno parametri, minore il rischio di overfitting

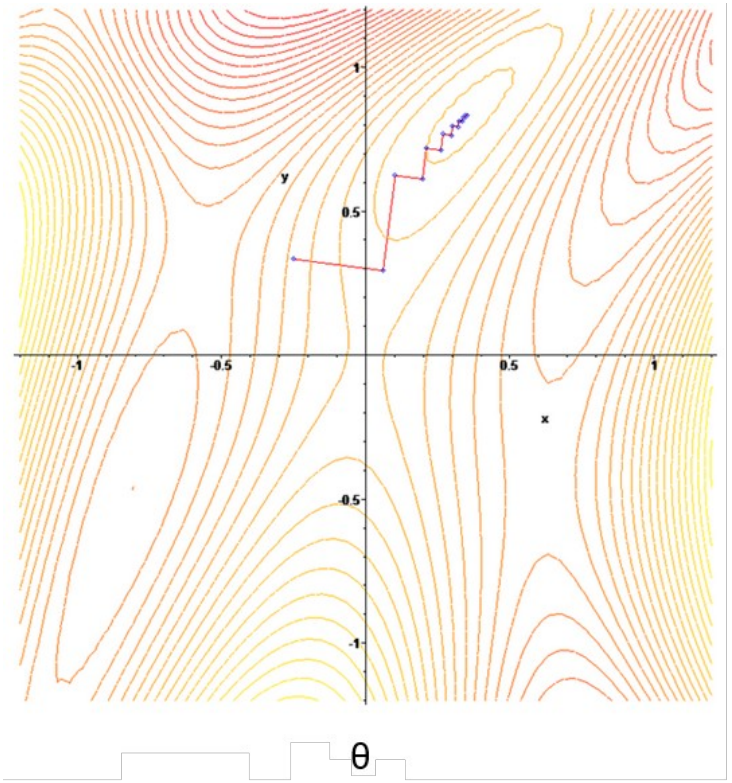


Emersione «spontanea» della gerarchia

Ma come viene creata questa gerarchia della rappresentazione?

Il processo è in gran parte spontaneo e ricorda quello che abbiamo visto con lo XOR problem

Nello spazio dei parametri ($\theta = \{W_1, W_2, \ldots W_n, \boldsymbol{v}\}$), i punti (le soluzioni) in cui le matrici degli hidden layer $W_1, W_2, \ldots W_n$ portano ad una rappresentazione gerarchica corrispondono a valori della loss più bassi (minimi locali)



Creazione «forzata» della gerarchia

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Spesso, però, quest'organizzazione gerarchica viene anche imposta (o incoraggiata…) utilizzando una specifica topologia della rete che si discosta dall'MLP

L'idea base è costruire una rete che abbia una topologia delle connessioni diversa dall'MLP e che sia tale da incoraggiare quest'organizzazione gerarchica dell'informazione

Vedremo questa cosa con particolare riferimento alle reti "convoluzionali" ("Convolutional Neural Network", CNN)

Convolutional Neural Network

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Convolutional Neural Network

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Le CNN sono usate soprattutto in Computer Vision

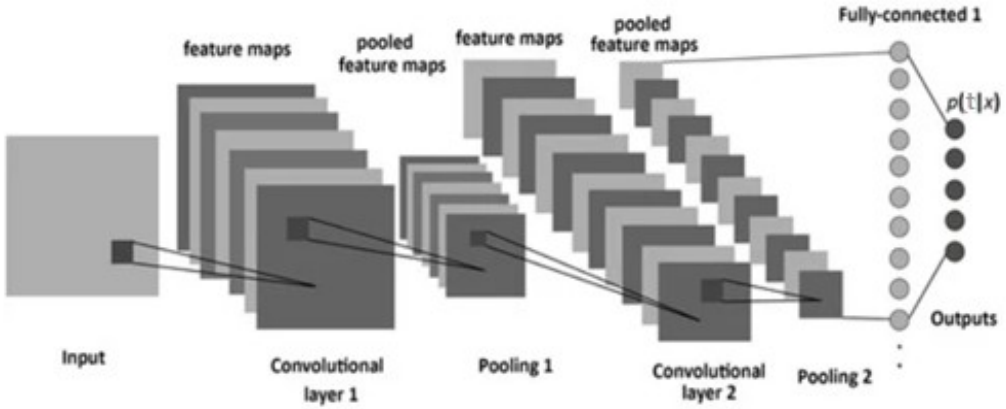
Le vedremo più da vicino sia per il loro specifico interesse pratico, sia per chiarire cosa si intende per topologia della rete che induce più o meno forzatamente una rappresentazione gerarchica della conoscenza

Convolutional Neural Network

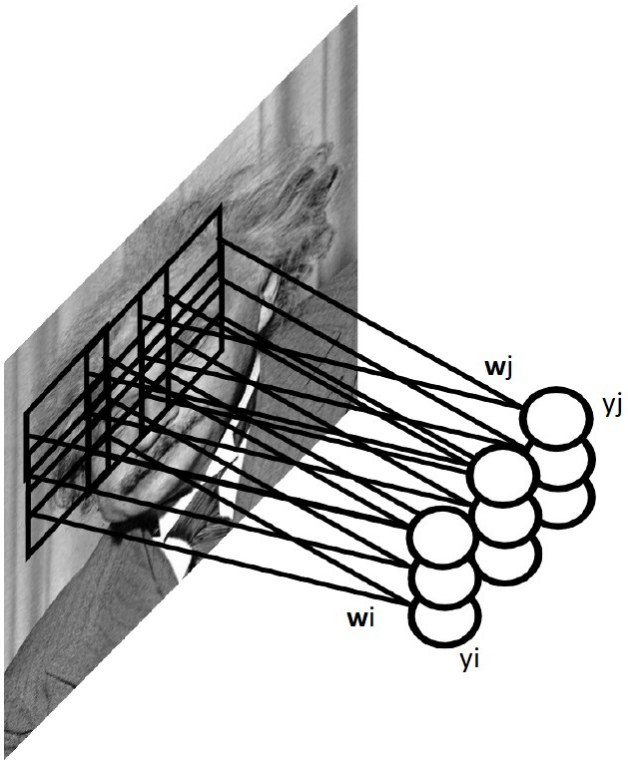
L'idea principale è rappresentare l'informazione nei pesi della rete in maniera *invariante rispetto a traslazioni* perchè una sotto-struttura visiva (e.g., una linea, un occhio, ecc.) può trovarsi in parti diverse dell'immagine e vogliamo *ri-usare* gli stessi pesi per rappresentare la stessa sotto-struttura, indipendentemente dalla sua posizione

Rispetto ad un MLP, i neuroni nell'input layer e negli hidden layer sono disposti spazialmente (ovvero, hanno una struttura 2D)

La parte finale è un (piccolo) MLP



Convolutional Neural Network

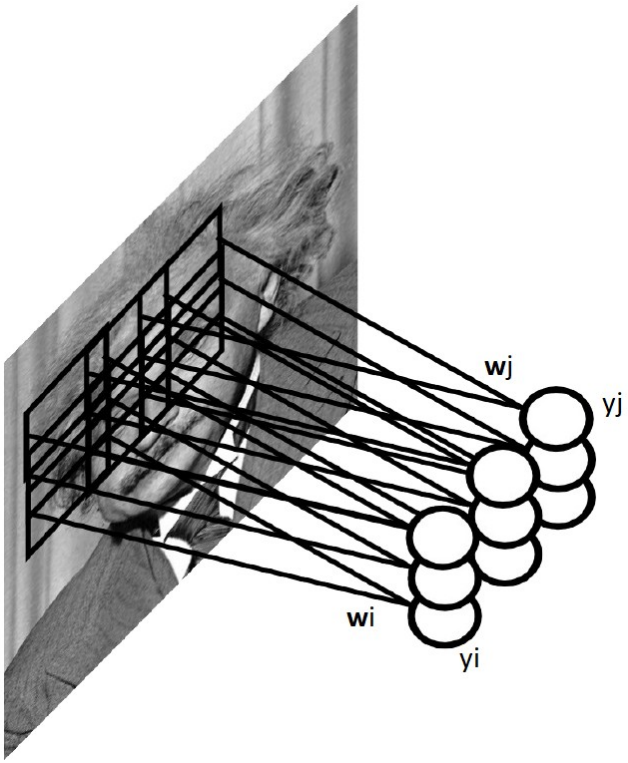


Differentemente da un MLP, nelle CNN i pixel dell'immagine in input non vengono linearizzati ma mantengono la loro struttura 2D

L'idea base è che ogni neurone negli hidden layer è connesso solo con una piccola finesta di neuroni del layer precedente

In questo modo il neurone ha un "campo visivo" (receptive field) ristretto: "vede" solo un dettaglio dell'immagine, per cui può imparare a riconoscere solo strutture semplici (limitato livello di astrazione)

Convolutional Neural Network



Nello stesso layer, hidden unit (y_j) in posizioni traslate rispetto ad y_i sono connesse con delle corrispondenti finestre traslate dell'immagine in input e *condividono gli stessi pesi* ($\boldsymbol{w}_i = \boldsymbol{w}_j$)

Siccome y_j ed y_i hanno anche la stessa nonlinearità ($f()$), il "pattern di attivazione" (ovvero la sotto-struttura dell'immagine che attiva il neurone) y_j è lo stesso pattern che attiva y_i

Ad ese., questo pattern potrebbe essere una linea, o un cerchio o un'altra struttura più o meno complessa, ed è indipendente dalla posizione dell'immagine in cui si trova, ovvero è *invariante rispetto a traslazioni*

I pesi contenuti in $\boldsymbol{w} = \boldsymbol{w}_i = \boldsymbol{w}_j$ costituiscono il *filtro* del layer convolutivo e rappresentano il pattern di attivazione in maniera, appunto, *invariante rispetto a traslazioni*

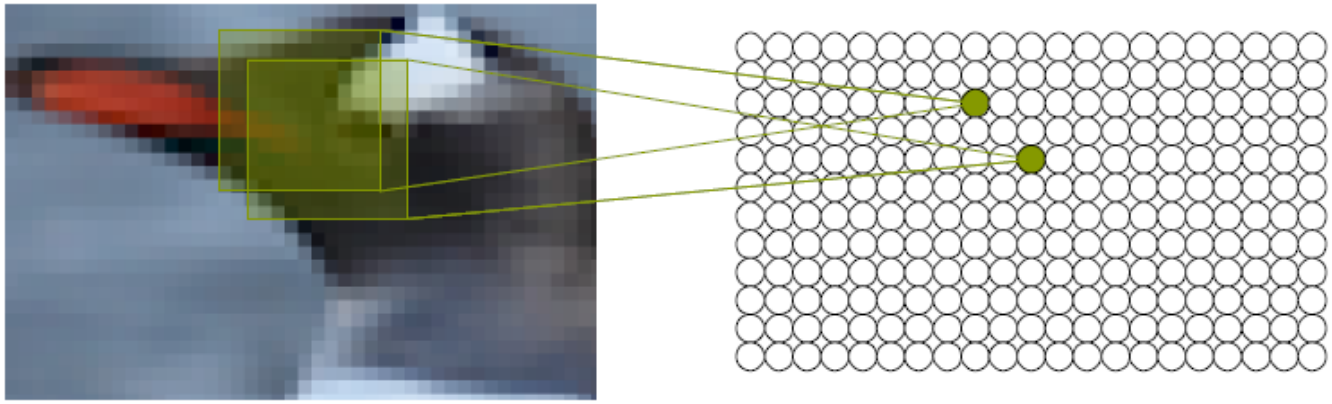
Struttura spaziale

Nell’esempio in figura, il “filtro” (***w***), composto da 12x12 pesi, è ripetuto per tutta l’immagine

Questo filtro convolutivo ha solo 144 parametri

La finestra nell’immagine in input (12 x 12) è il *receptive field* del neurone

I neuroni dell’hidden layer sono anch’essi disposti in una struttura 2D

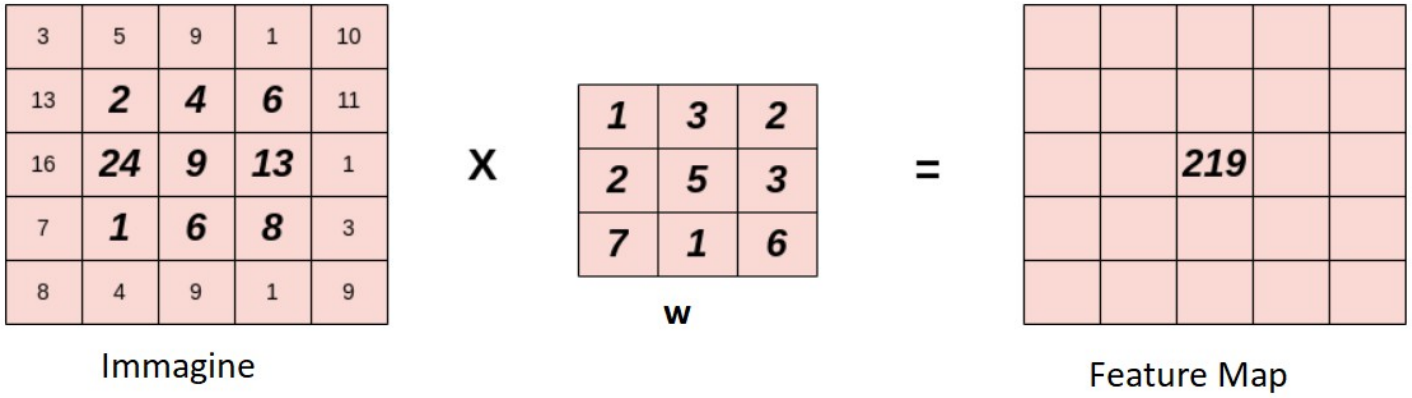


Convoluzione

Questo tipo di reti si chiamano “convolutive” e i pesi sono organizzati in “filtri” perchè ricordano la convoluzione usata in Image Processing

Infatti, si può immaginare che il filtro “scorra” sull’immagine (come nella convoluzione), producendo un output (uno scalare) per ogni sua “sovrapposizione” ad una finestra dell’immagine di grandezza uguale al filtro

Il valore in output di tutti i neuroni che usano quel filtro corrisponde all’*activation map* (o *feature map* o *channel*)



Convoluzione: esempio

Immagine

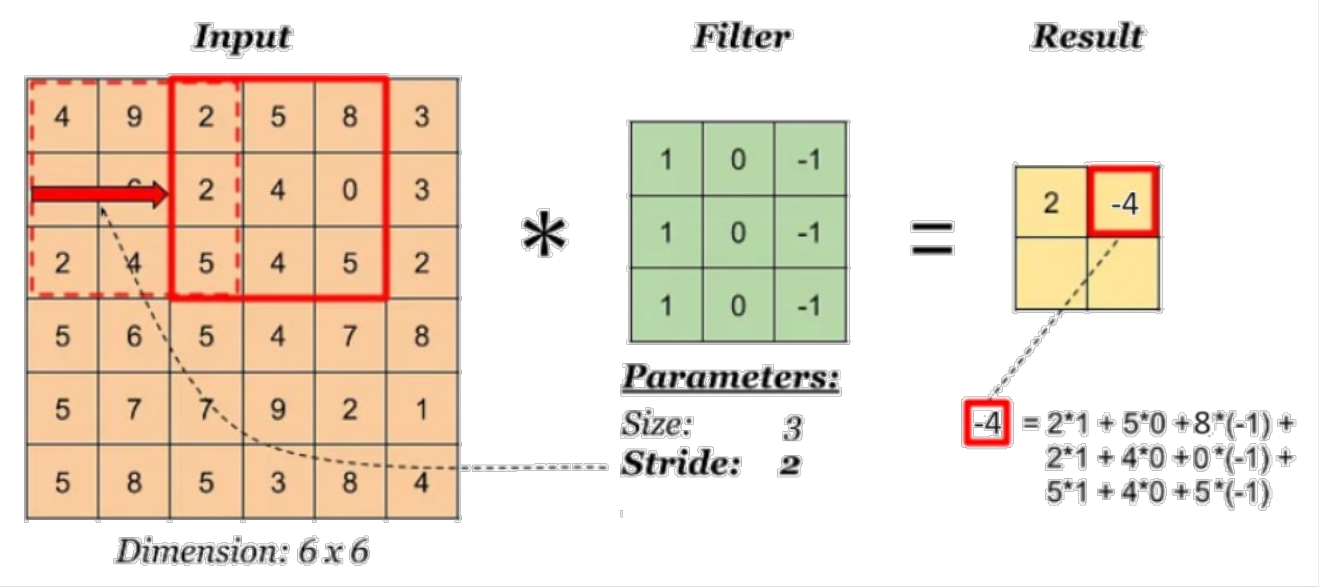
0	0	0.5	1	0.5	0	0
0	0	0.5	1	0.5	0	0
0	0	0.5	1	0.5	0	0
0	0	0.5	1	0.5	0	0
0	0	0.5	1	0.5	0	0
0	0	0.5	1	0.5	0	0
0	0	0.5	1	0.5	0	0

Filtro

Feature Map

Risoluzione spaziale delle activation map

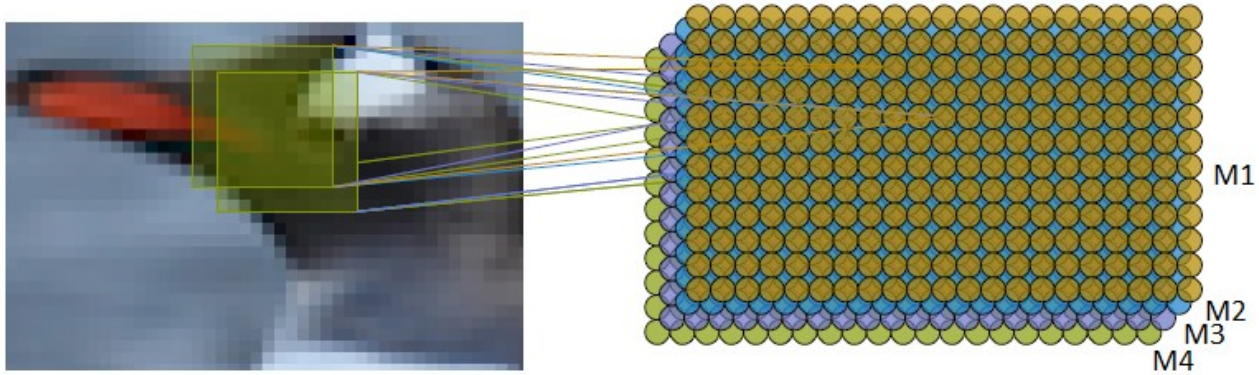
In un dato layer convolutivo, *m x m* è la *spatial size* del filtro
s è lo *stride*: stabilisce il passo di traslazione orizzontale e verticale della finestra in input
Se *s* < *m* allora le finestre saranno parzialmente sovrapposte
Se *A x L* è la risoluzione della mappa in input, quella in output sarà (più o meno…) *A/s x L/s*



Filtri multipli per ogni layer

Il primo hidden layer ha *c* diversi filtri $\boldsymbol{w}_1$, $\boldsymbol{w}_2$, … $\boldsymbol{w}_c$, che corrispondono ad altrettante *activation map* M_1, M_2, …, M_c

Complessivamente M_1, M_2, …, M_c costituiscono l'input del layer successivo



Ese.: $\mathbf{w}_1$, potrebbe essere un filtro "specializzato" per riconoscere segmenti verticali, $\mathbf{w}_2$ per riconoscere segmenti obliqui, ecc. Le feature map corrispondenti rappresenteranno la presenza di questi pattern in ogni zona dell'immagine

Profondità del filtro

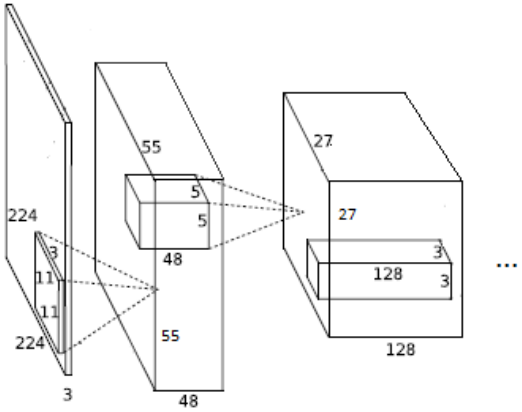
L'immagine in input è organizzata in 3 channel (RGB)

In generale, se ho *c* filtri nel layer i-esimo, $\mathbf{w}_1^i$, … $\mathbf{w}_c^i$, allora avrò M_1^i, …, M_c^i mappe in input al layer *i+1*

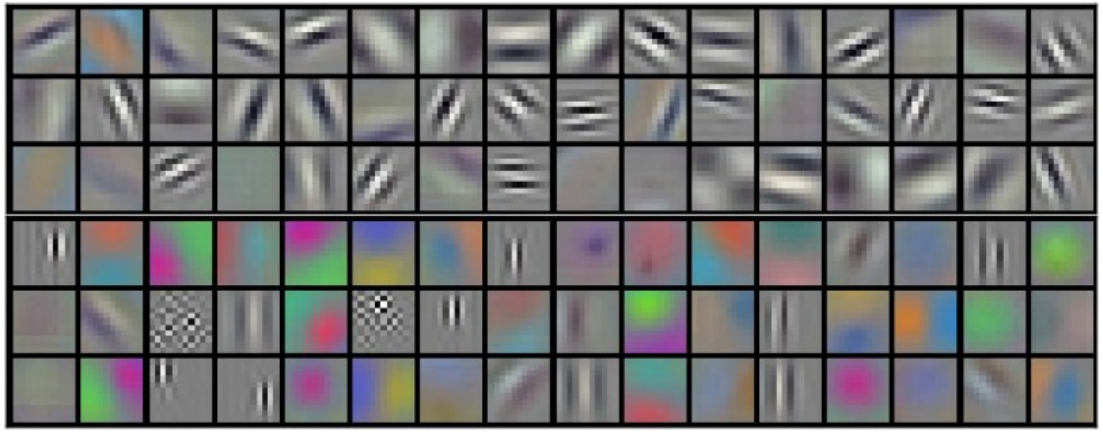
Nell'hidden layer successivo ogni filtro $\mathbf{w}_j^{i+1}$ ha dimensioni $m_{i+1} \times m_{i+1} \times d_{i+1}$, dove $d_{i+1} = c$ è la profondità del filtro al layer $i + 1$

I filtri hanno una profondità per poter *combinare* i valori di attivazione di filtri diversi applicati al layer precedente

La rete può in tal modo combinare pattern di attivazione diversi del layer precedente (e.g., segmenti con orientazione varia) per costruire pattern più astratti



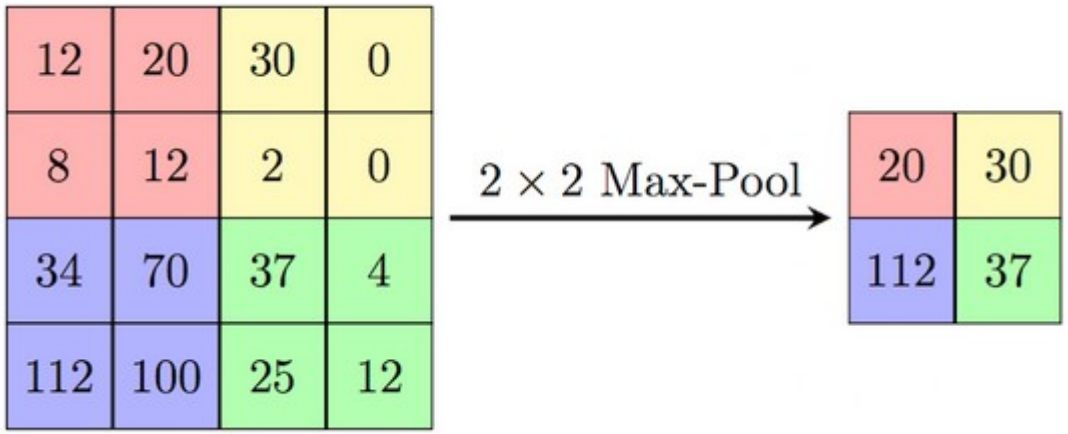
Esempio di filtri reali per il primo layer



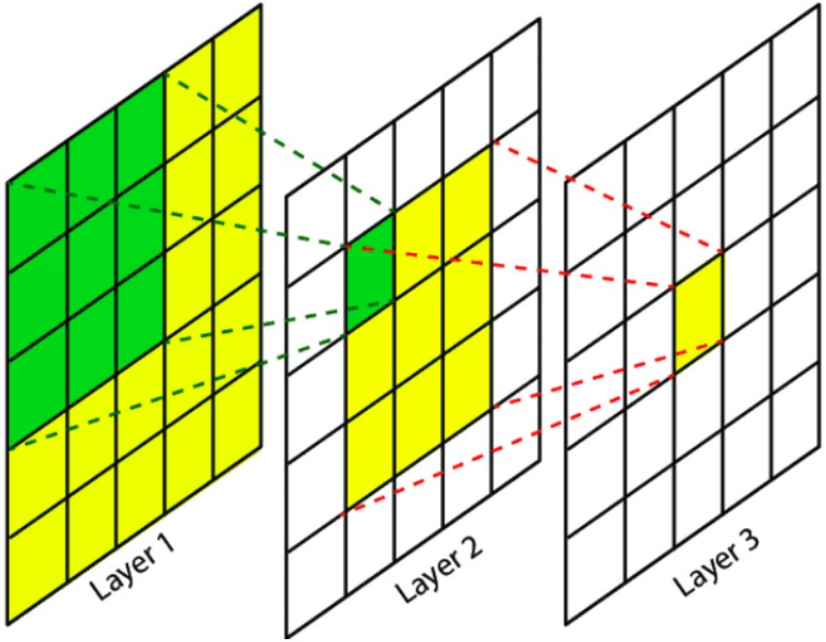
(Parziale) invarianza al cambiamento di scala

Con uno stride $s > 1$ riduco la risoluzione delle feature map in output di un layer rispetto a quelle in input

Un effetto simile posso ottenerlo con il *max pooling*



Receptive field



Il receptive field è la finestra nell'immagine che influenza la funzione calcolata di un determinato hidden neuron (la parte di immagine che quel neurone "vede")

Nei primi layer è piccolo, quindi possono essere rappresentate solo strutture piccole (basso livello di astrazione)

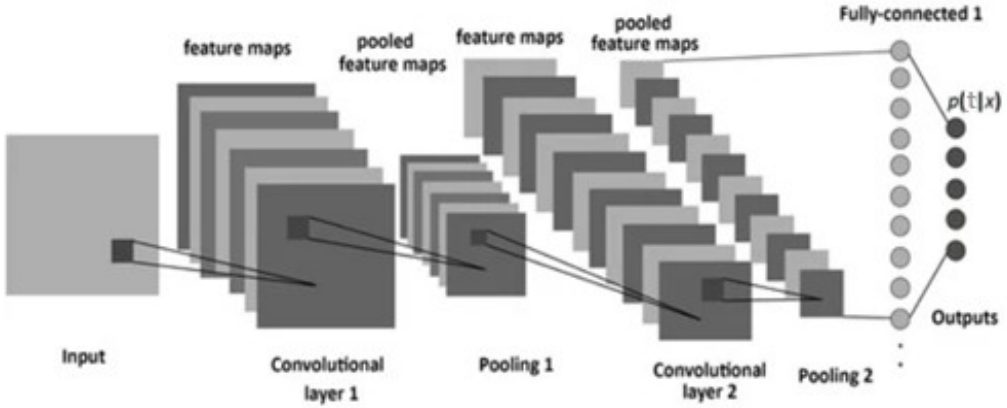
Nei layer successivi il receptive field aumenta progressivamente

Se la risoluzione delle activation map diminuisce (e.g., per effetto di uno stride > 1 o di maxpooling layer), il receptive field aumenterà ancora di più

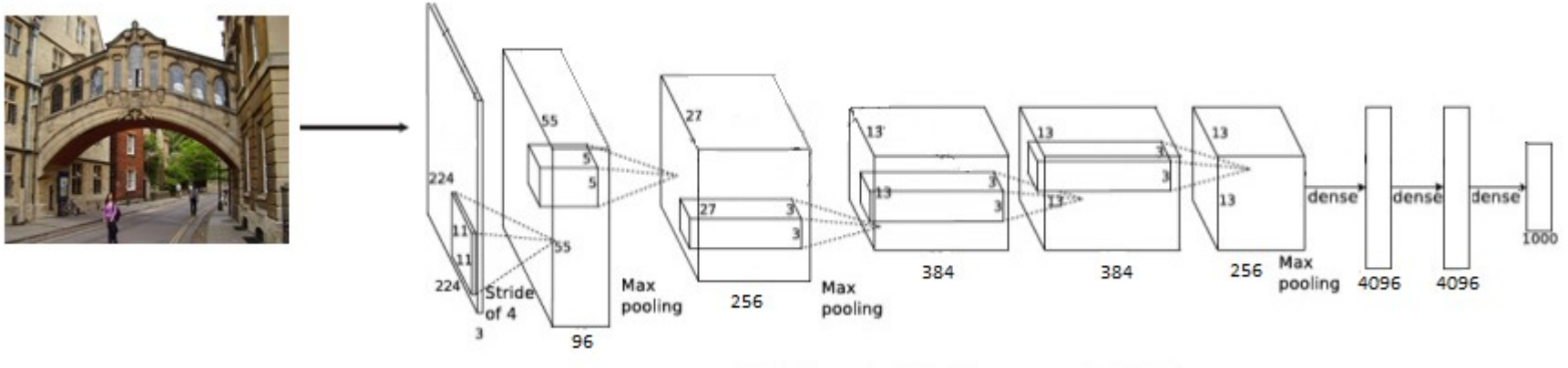
Man mano che il receptive field aumenta, possono essere rappresentate strutture sempre più grandi (maggiore livello di astrazione)

Esempio di architettura CNN completa

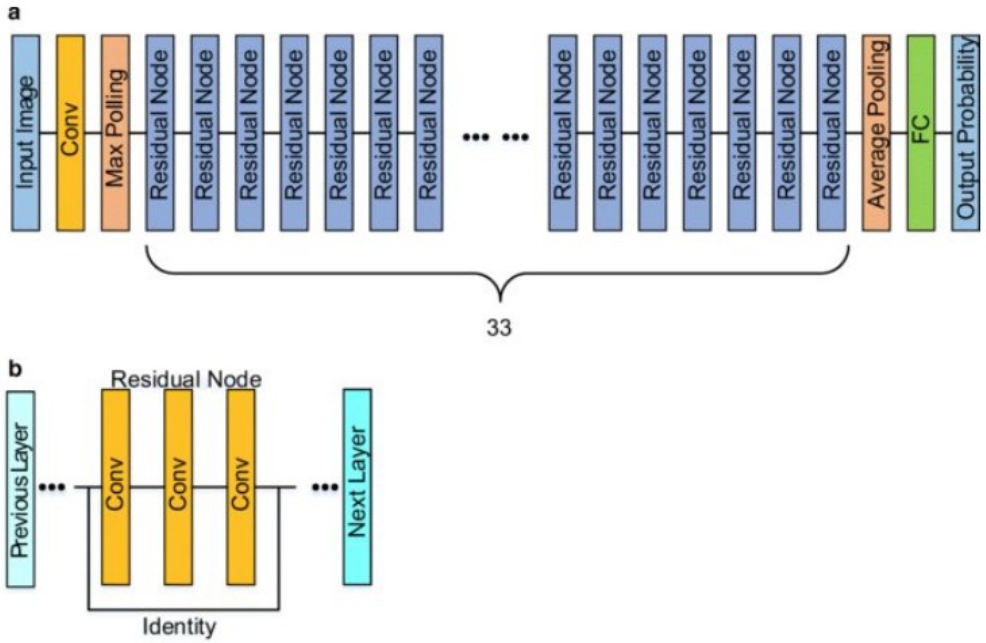
Le feature map dell'ultimo layer convolutivo vengono linearizzate (i.e., trasformate in un vettore, come abbiamo visto fare all'inizio di questa lezione per l'input dell'MLP) e date in input alla parte finale della ANN che è un piccolo MLP



Esempio di architettura CNN completa: AlexNet (2012)



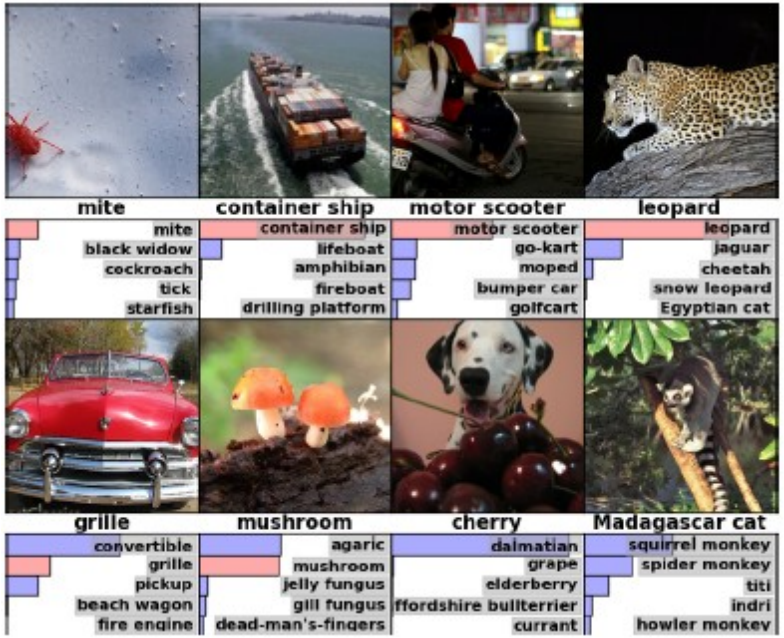
ResNet 101 (2015)



44.5 Milioni di parametri

Esempio: Image Classification

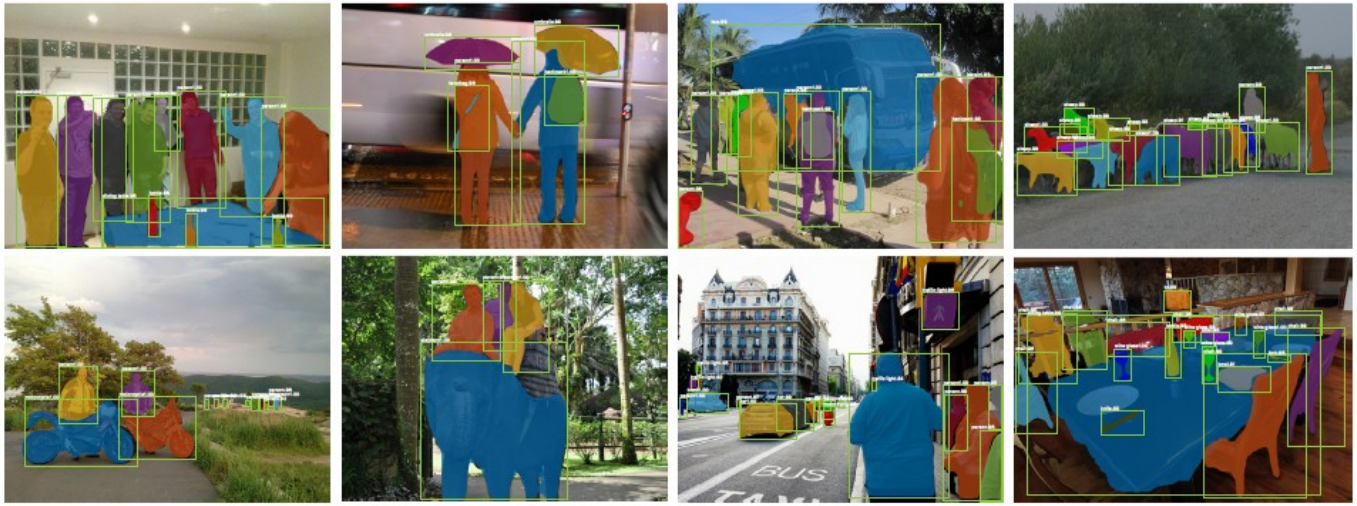
AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA



Risultati di AlexNet addestrata con 1.3 milioni di immagini etichettate

Object detection e segmentation

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA



Automotive

Source: A. Shashua, S. Seitz

Conclusioni

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Quando usare il Deep Learning?

1. Quando ho un dominio con tanti sample di training che permettano di addestrare ANN con milioni (o miliardi) di parametri senza fare overfitting
2. Quando non ho una conoscenza a priori del domino da poter usare per creare delle feature sufficientemente informative

Ad esempio, in ambito medico, se l'esperto di medicina sa che per predire una certa diagnosi sono necessari degli esami specifici, posso creare un dataset di feature (numeriche/categoriche) che descrivono, paziente per paziente, l'esito di quell'esame

In quel caso, posso usare un MLP con un solo hidden layer (oppure una SVM, oppure una Random Forest, oppure…) . Tipicamente, aggiungere hidden layer quando ho già feature informative in partenza non serve a niente…

Referimenti

- Tool per visualizzare la rappresentazione gerarchica di una CNN: https://mriquestions.com/convolutional-network.html