Graphs and graph theory

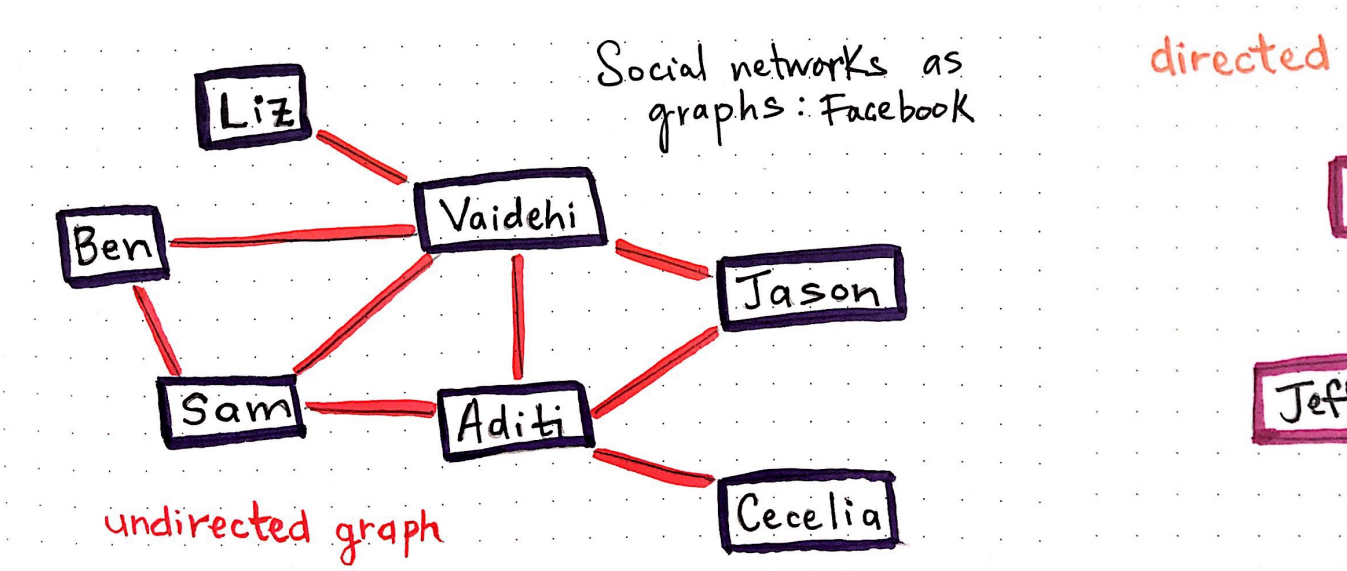
A set V of vertices and a set E of unordered and ordered pairs of vertices; denoted by $G(V, E)$. An unordered pair of vertices is said to be an **edge**, while an ordered pair is said to be an **arc**. A graph containing edges alone is said to be **non-oriented**; a graph containing arcs alone is said to be **oriented**. An arc (or edge) can begin and end at the same vertex, in which case it is known as a **loop**.

One says that an edge u, v connects two vertices u and v, while an arc (u, v) begins at the vertex u and ends at the vertex v. Vertices connected by an edge or a loop are said to be **adjacent**. Edges with a common vertex are also called **adjacent**. An edge (arc) and any one of its two vertices are said to be **incident**.

There are various ways of specifying a graph. Let $u_1, ..., u_n$ be the vertices of a graph $G(V, E)$ and let $e_1, ..., e_m$ be its edges. The **adjacency matrix** corresponding to G is the matrix $A = (a_{i,j})$ in which the element $a_{i,j}$ equals the number of edges (arcs) which join the vertices u_i and u_j (go from u_i to u_j) and $a_{i,j} = 0$ if the corresponding vertices are not adjacent. A sequence of edges $(u_0, u_1), ..., (u_{r-1}, u_r)$ is called an **edge progression** connecting the vertices u_0 and u_r. An edge progression is called a **chain** if all its edges are different and a simple chain or path if all its vertices are different. A closed (simple) chain is also called a (simple) **cycle**.

Matrice di adiacenza. Wolfram MathWorld

The degree of a vertex u_i of a graph G, denoted by d_i, is the number of edges incident with that vertex. The **length** of an edge progression (chain, simple chain) is equal to the number of edges in the order in which they are traversed. The length of the shortest simple chain connecting two vertices u_i and u_j in a graph G is said to be the distance $d(u_i, u_j)$ between u_i and u_j. The quantity $min_{u_i} max_{u_j} d(u_i, u_j)$ is called the **diameter**, while a vertex u_0 for which $max_{u_j} d(u_i, u_j)$ assumes its minimum value is called a centre of G. A graph can contain more than one centre or no centre at all.
Graph. Encyclopedia of Mathematics



Graph Convolutional Network

Il punto di forza delle CNN e delle RNN è la loro capacità di saper sfruttare al meglio la conoscenza delle interconnessioni fra i dati in input. Ad esempio un filtro convolutivo si basa sul fatto che i dati necessari ad elaborare un singolo pixel (per estrarne una feature) si trovino nei pixel a lui vicini (tipicamente, a due o tre pixel di distanza), mentre i pixel più distanti possano essere ignorati. Da qui si può evidenziare come dietro questo concetto vi sia un grafo, in quanto la *vicinanza* dei pixel equivale a rappresentare l'immagine come un grafo dalla struttura perfettamente regolare:

Wolfram MathWorld

Sfruttare questa informazione di vicinanza tra i pixel è il cuore di una rete convolutiva, sebbene i pixel in un'immagine sono interconnessi in modo estremamente regolare. C'è modo per sfruttare l'informazione contenuta nel grafo senza senza sacrificare efficienza o flessibilità delle architetture nel caso di grafi irregolari (i.e. con nodi quasi isolati, alcuni centrali etc. . .)?

Definito un grafo di N nodi, su ciascuno dei quali è definito un segnale $x_n \in R^C$ (dove C è il numero di "canali" del signale del segnale, i.e. 3 colori per un pixel). Denotata con X la matrice $N \times C$ che colleziona su ogni riga il segnale definito sul rispettivo nodo. Infine, A sarà la matrice $N \times N$ di adiacenza.

Per comprendere il funzionamento delle **GCN**, ignoriamo per un attimo le connessioni fra i vari nodi. In questo caso, una rete neurale "classica" che operasse su ciascun nodo si comporrebbe di strati del tipo:

$$g(X) = ReLU(XW)$$

dove W è la matrice di pesi dello strato e $ReLU(s) = max(0, s)$ è la nostra funzione di attivazione che aggiunge non-linearità ($ReLU$ solo come esempio, andrebbe bene una qualsiasi funzione di attivazione). Questo schema processa ogni nodo in maniera **indipendente**, ignorando completamente le connessioni fra di essi e di conseguenza una quantità potenzialmente molto importante di informazione.

Consideriamo però questa semplice variante:

$$g(X) = ReLU(\mathbf{A}XW)$$

dove A è la matrice di adiacenza introdotta prima. La **logica è simile a quella di una rete convolutiva**: l'informazione elaborata da ciascun "filtro" dipende solo dagli immediati vicini del nodo, mentre le stesse operazioni (definite dalla matrice W) vengono applicate su tutti i nodi del grafo in simultanea.
GEOMETRIC DEEP LEARNING: GRAPH CONVOLUTIONAL NETWORK. IAML

TODO: Qui c'è da esplorare la GRAPH CONVOLUTION E GRAPH FOURIER TRANSFORM

Convolutional Neural Networks

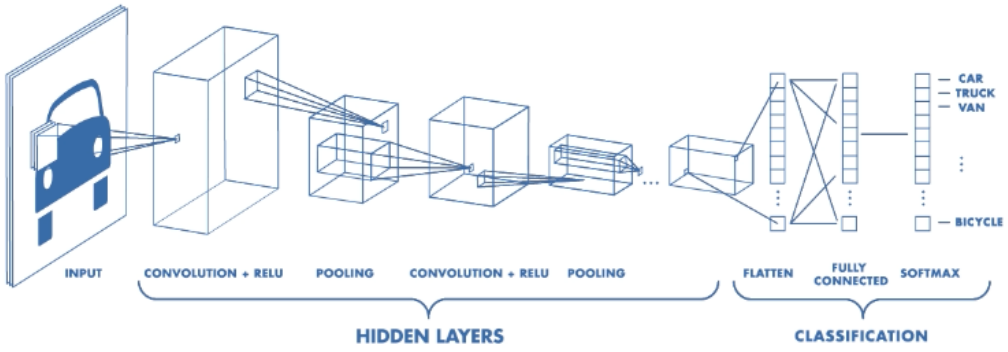


Figure 1:

Per sfruttare al meglio la correlazione spaziale delle informazioni contenute nelle immagini il modello più classico di rete neurale, il Multilayer Perceptron (MLP) è stato abbandonato in favore delle CNN. Questo per diversi motivi: il quantitativo di *pesi* di un immagine diventa velocemente ingestibile al crescere della risoluzione dell'immagine (i.e. un immagine 224 x 224

x 3 ha circa 150000 pesi); un altro problema è che l'MLP reagisce in modo differente se ha in input un immagine o una sua versione traslata: l'**MLP non è translation invariant**.

Nelle CNN l'approccio è diverso: per analizzare l'influenza di (ad esempio) pixel vicini fra loro si usano dei filtri, di una dimensione specificata dall'utente, e si spostano su tutta l'immagine. Per ogni punto dell'immagine viene effettuata un'operazione di convoluzione con il filtro.

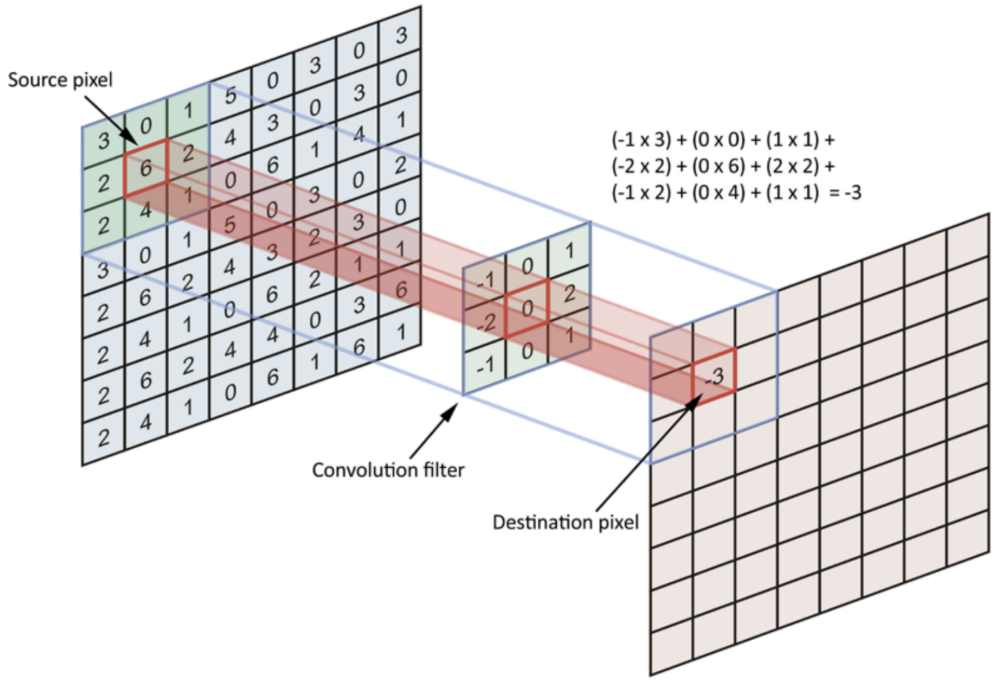


Figure 2:

Ogni filtro sostanzialmente può essere considerato un *feature identifier*, in quanto viene attivato da una funzione ($ReLU, tanh, ELU$, etc...) quando l'immagine presa dal filtro corrisponde alla feature cercata. Di seguito, la visualizzazione di alcuni filtri:

Un primo livello convolutivo può ad esempio aiutare a identificare feature molto semplici (i.e. curve e linee rette, forme geometriche), un successivo livello può identificare feature più complesse (i.e. occhi, bocca, naso etc...) e altri livelli ancora possono rilevare dei volti completi:

Dopo aver effettuato questa operazione con tutti i filtri (il cui numero è deciso anch'esso dall'utente) il risultato è una **feature map**. I filtri vengono aggiornati durante il training della rete.

Nel passare i filtri sull'immagine per mantenere costanti le dimensioni delle feature maps si utilizzano diverse strategie di *padding*:

Successivamente, tra i vari livelli convoluzionali, esistono dei livelli di **Pooling**, volti a ridurre il numero di parametri e di computazione nella rete, riducendone le dimensioni spaziali. In

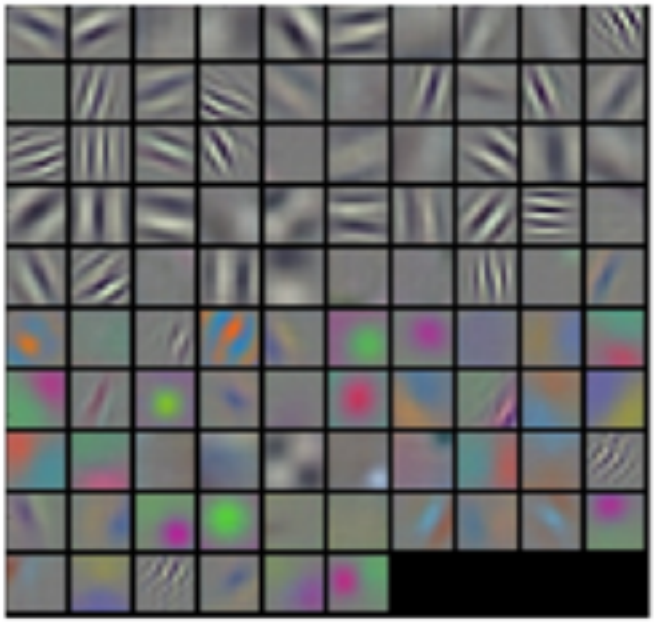


Figure 3:

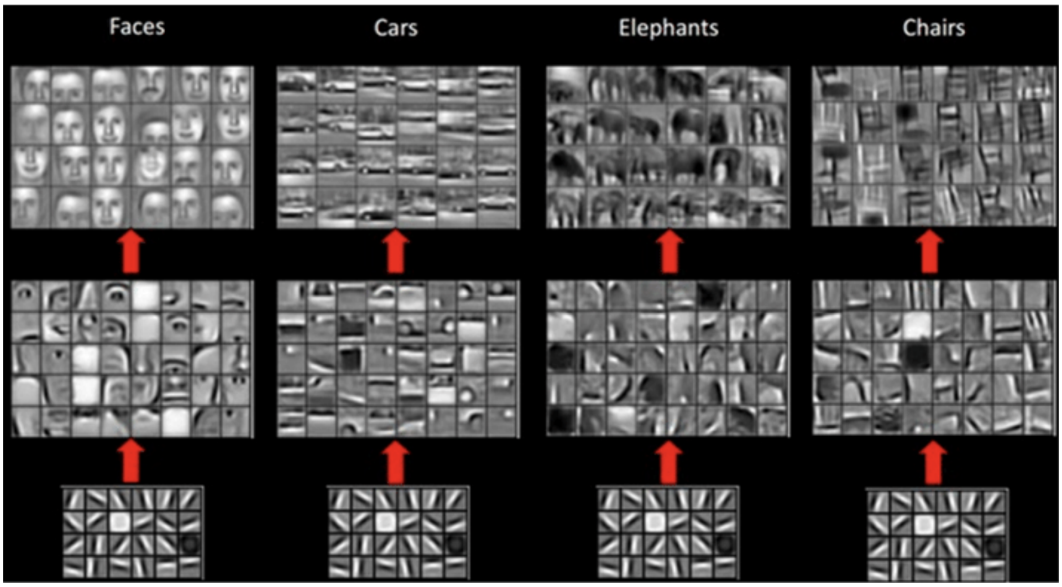


Figure 4:

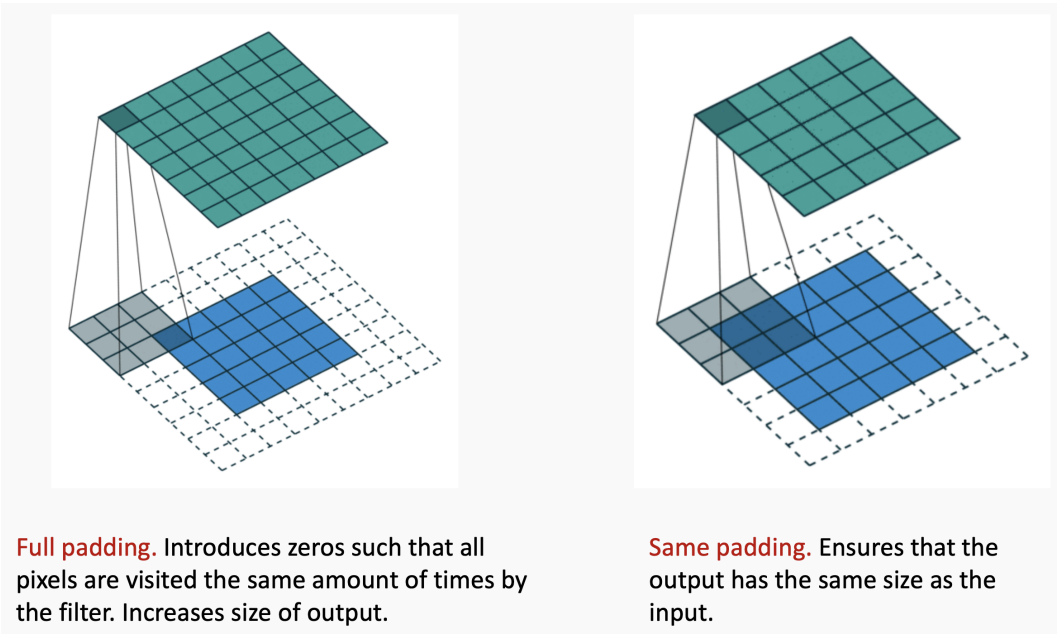


Figure 5:

questi livelli si possono svolgere diverse operazioni, sebbene nella maggior parte dei casi si utilizzeranno dei livelli di **Max-pooling**, che prendono attraverso un meccanismo di sliding windows che prende solo il valore massimo da un filtro, applicato su tutte le porzioni di immagini.

Simple Introduction to Convolutional Neural Networks
Basics of the Classic CNN
Basic Overview of Convolutional Neural Network (CNN)

Graph Neural Networks

Graph neural networks (GNNs) are deep learning based methods that operate on graph domain.