



EMOTIONS RECOGNITION

utilizzando reti neurali
convoluzionali

INDICE

01

Introduzione

02

Strumenti

03

Architettura

04

Algoritmi di
ottimizzazione

05

Feature
Classification

06

Implementazione



01

INTRODUZIONE



COS'È UN'EMOZIONE

L'emozione è uno stato mentale e fisiologico associato a modificazioni psicologiche o a stimoli interni ed esterni.

Rende più efficace la reazione dell'individuo a situazioni in cui si rende necessaria una risposta immediata ai fini della sopravvivenza.

Per questo non utilizza processi cognitivi ed elaborazione cosciente.

LE ESPRESSIONI FACCIALI

- Le emozioni sono innate nell'essere umano.
- Le espressioni facciali corrispondenti a un'emozione sono universalmente riconosciute, indifferentemente da luogo, tempo e cultura.



CARATTERISTICHE DELLE EMOZIONI



1. Rabbia

| Parte del corpo | Caratteristica |
|-----------------|-------------------------|
| Sopracciglia | Abbassate e ravvicinate |
| Mento | Alzato |
| Sguardo | Intenso |



2. Disgusto

| Parte del corpo | Caratteristica |
|-----------------|----------------------------|
| Naso | Arricciato |
| Bocca | Labbro superiore sollevato |
| Sopracciglia | Abbassate |



3. Paura

| Parte del corpo | Caratteristica |
|-----------------|----------------|
| Sopracciglia | Corrugate |
| Bocca | Aperta |
| Occhi | Spalancati |



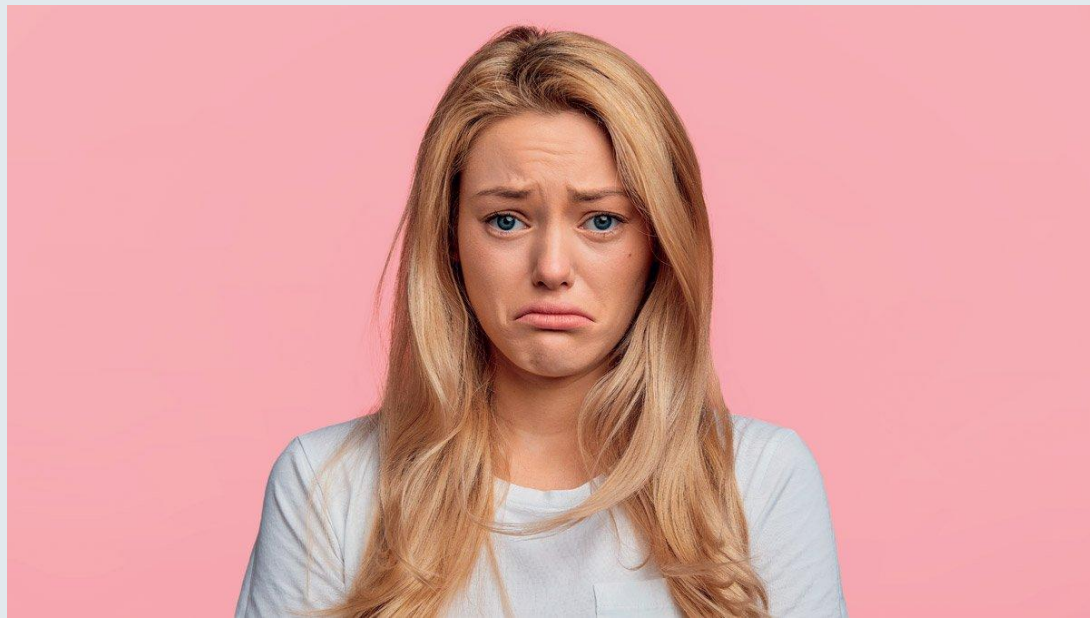
4. Felicità

| Parte del corpo | Caratteristica |
|-----------------|-----------------------------------|
| Guance | Sollevate |
| Labbra | Angoli sollevati |
| Occhi | Angoli esterni a zampa di gallina |



5. Tristezza

| Parte del corpo | Caratteristica |
|-----------------|------------------|
| Sopracciglia | Corrugate |
| Labbra | Angoli abbassati |
| | |



6. Sorpresa

| Parte del corpo | Caratteristica |
|-----------------|----------------|
| Mascella | Caduta |
| Sopracciglia | Alzate |
| Occhi | Spalancati |





EMOTION RECOGNITION



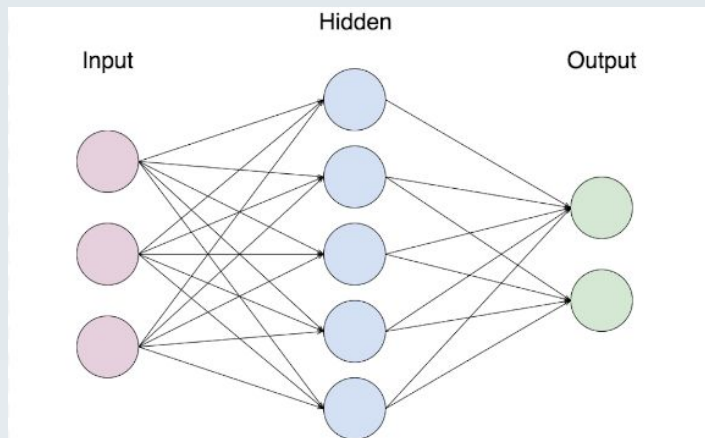
COS'È L'EMOTION RECOGNITION?

- L'emotion recognition è una tecnica usata per capire che emozione prova una persona.
- Grazie alle espressioni facciali o al tono di voce

CAMPI DI APPLICAZIONE

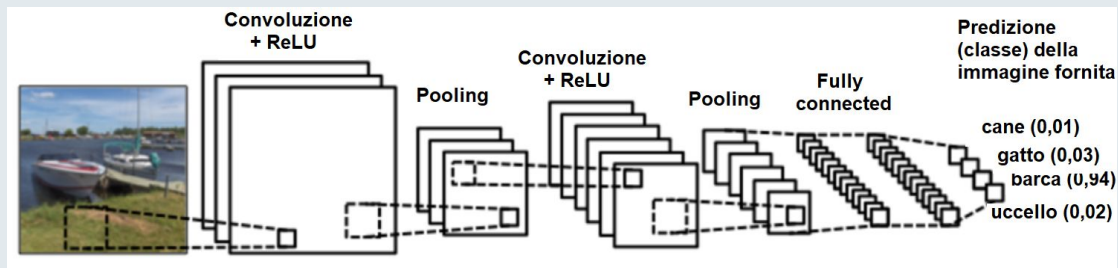
- **Campo dell'accessibilità:** Utile a persone non vedenti e affette da sindrome dello spettro autistico a comprendere le emozioni delle persone che hanno di fronte
- **Campo delle risorse umane:** Utile per capire se una persona sta mentendo e se è veramente interessata alla posizione lavorativa aperta
- **Assistenza clienti:** Utile per capire il grado di soddisfazione dei clienti
- **Automobilistico:** Avvisare il Conducente di stanchezza e attuare piani per tenerlo sveglio

DIFFERENZA TRA NN E CNN



NN

CNN



DIFFERENZA TRA NN E CNN

ANN è composta da diversi layer di neuroni fortemente interconnessi

A causa della sua profondità si potrebbero verificare i seguenti problemi:

- Costo computazionale elevato

- La discesa del gradiente esploderebbe

DIFFERENZA TRA NN E CNN

Nella CNN, invece, si hanno delle caratteristiche che riducono di molto i costi computazionali in caso di image recognition.

Le CNN condividono i pesi a gruppi. Neuroni diverso dello stesso livello eseguono lo stesso tipo di operazione su porzioni diverse dell'input



02

STRUMENTI




Dataset Fer2013

INTRODUCTION



Fer2013 contiene circa 30.000 immagini facciali di diverse espressioni con dimensioni limitate a 48×48 e le label principali possono essere suddivise in 7 tipi:

- 0=Angry,
- 1=Disgust,
- 2=Fear,
- 3=Happy,
- 4=Sad ,
- 5=Surprise,
- 6=Neutral.



Estrazione dataset



```
from google.colab import drive
drive.mount('/content/drive')

data =
pd.read_csv('/content/drive/My
Drive/fer2013.csv')
```

PROBLEMI



```
graph LR; A[PROBLEMI] --- B[Imbalance Problem]; A --- C[Intra-Class Variation]; A --- D[Occlusion]; A --- E[Contrast Variation]; A --- F[Outliers]
```

Imbalance
Problem

Intra-Class
Variation

Occlusion

Contrast
Variation

Outliers



IMBALANCE PROBLEM

PROBLEMA

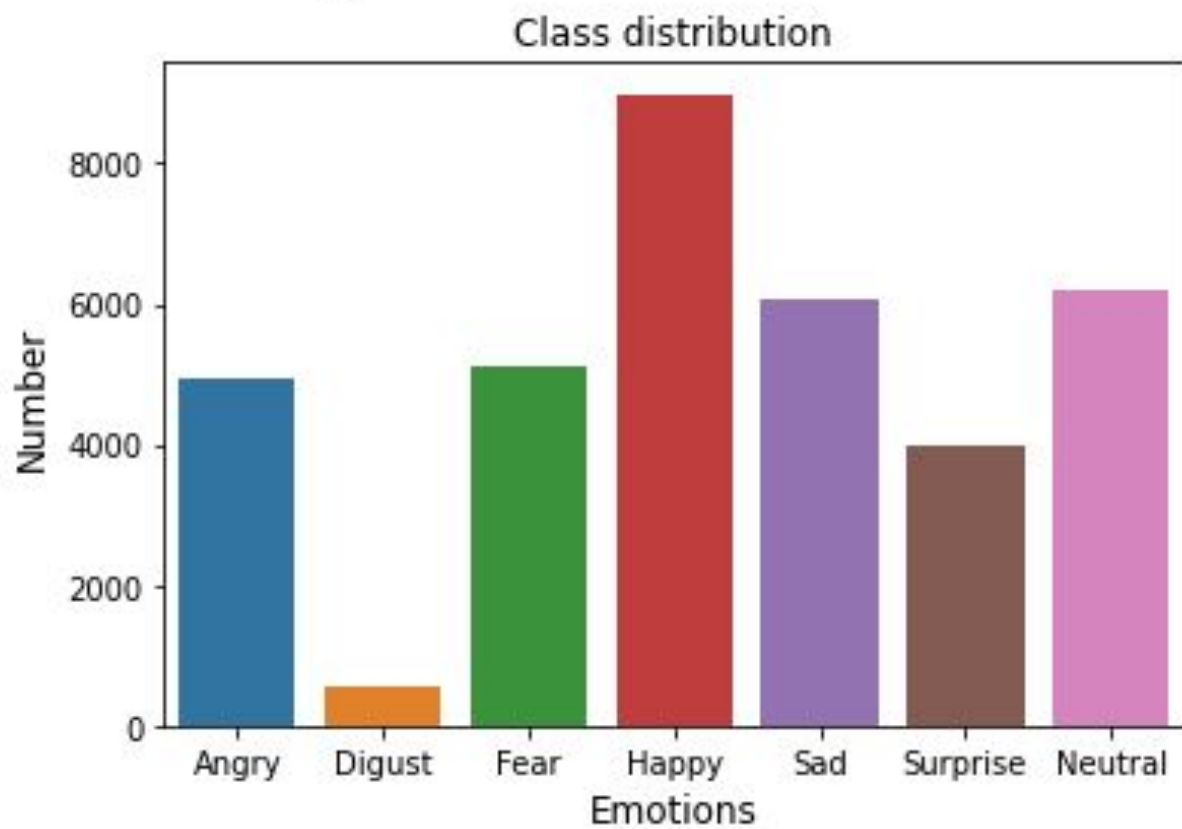
Squilibrio della Classificazione

SOLUZIONE

Data Augmentation

- GANS
- Scaling

| Fer2013 | 0=Angry | 1=Disgust | 2=Fear | 3=Happy | 4=Sad | 5=Surprise | 6=Neutral |
|----------|---------|-----------|--------|---------|-------|------------|-----------|
| Training | 3995 | 436 | 4096 | 7214 | 4830 | 3171 | 4965 |
| Testing | 958 | 111 | 1024 | 1774 | 1247 | 831 | 1322 |



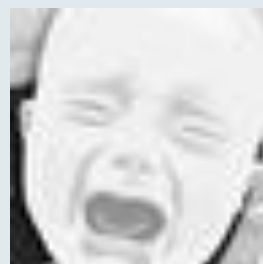
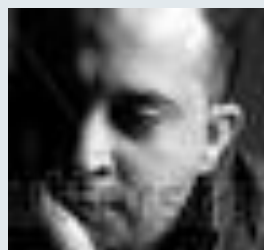
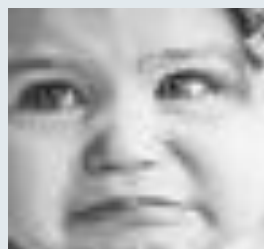
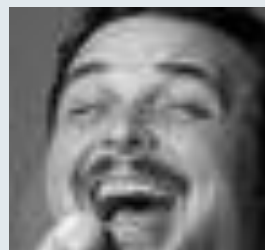
INTRA-CLASS VARIATION

“PROBLEMA”

Diverse tipologie di immagini all'interno di una classe

SOLUZIONE

Avoid Overfitting



Overfitting

Casi

Le prestazioni :
sul training e sul testing

Rilevazione sul dataset

- ▼ test
 - ▶ angry
 - ▶ disgust
 - ▶ fear
 - ▶ happy
 - ▶ neutral
 - ▶ sad
 - ▶ surprise
- ▼ train
 - ▶ angry
 - ▶ disgust
 - ▶ fear
 - ▶ happy
 - ▶ neutral
 - ▶ sad
 - ▶ surprise

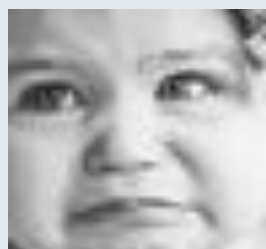
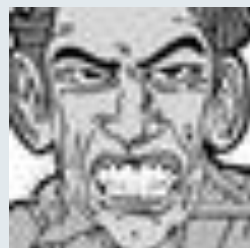
OCCLUSION /CONTRAST VARIATION/ OUTLIERS

SOLUZIONE

Scegliere immagini idonee e realistiche

PROBLEMA

Luminosità delle img, occhiali che coprono gli occhi, ecc





03

ARCHITETTURA

COMPONENTI DELL'ARCHITETTURA

- Feature Extraction
- Feature Classification
- Hidden Layers

MODELLO CNN

**Feature
Extraction**

Convoluzione

+

Pooling

Ex: data un'immagine
si rilevano due occhi,
una coda, delle zampe
e così via.

**Feature
Classification**

FCL

- Imparano come utilizzare le feature prodotte dalle convoluzioni per classificare correttamente le immagini.
- Funge da classificatore quindi assegna la probabilità che l'immagine sia un cane

LA CONVOLUZIONE

Definizione

In termini matematici è una combinazione tra due funzioni e ne produce una terza.

In questo caso :

Sottopone le immagini di input a una serie di filtri, ciascuno dei quali attiva determinate caratteristiche dalle immagini



Input X Filter → FEATURE MAP

Dimostrazione in 2D

- Si esegue una moltiplicazione matriciale
- Si eseguono più convoluzioni su un input
- I valori delle feature map è il risultato dell'operazione di convoluzione attraverso la funzione di attivazione (ReLU) -> non linearità

| | | | | |
|-----|-----|-----|---|---|
| 1x1 | 1x0 | 1x1 | 0 | 0 |
| 0x0 | 1x1 | 1x0 | 1 | 0 |
| 0x1 | 0x0 | 1x1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

| | | |
|---|--|--|
| 4 | | |
| | | |
| | | |

APPLICAZIONE FILTRO A IMMAGINE

Esempio:

INPUT



FILTER

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

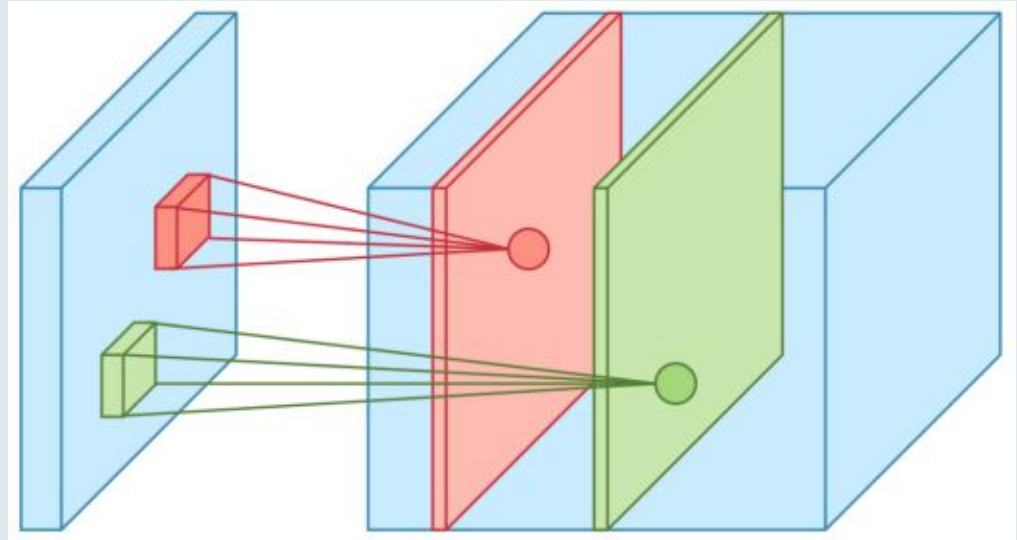
OUTPUT



FEATURE MAP

Dimostrazione in 3D

- Si tiene conto anche della profondità a causa dei canali di colore utilizzati in un'immagine (RGB)
- L'operazione di convoluzione su ciascun input, utilizzando un filtro diverso, viene eseguita in maniera indipendente
- Le feature map risultanti sono disgiunte



Ex: Se si usano 2 filtri ci sarebbero 2 feature map



```
model.add(Conv2D(32, (3, 3),  
                Padding = 'same',  
                kernel_initializer= 'he_normal'  
                ,  
                input_shape=(48, 48, 1)))
```

POOLING/RAGGRUPPAMENTO

- Tipo più comune : Max Pooling
- Si scorre una pooling window sull'input dopodichè si prende il valore massimo nella finestra .
- I livelli di pooling effettuano il downsampling

| | | | |
|---|---|---|---|
| 1 | 1 | 2 | 4 |
| 5 | 6 | 7 | 8 |
| 3 | 2 | 1 | 0 |
| 1 | 2 | 3 | 4 |

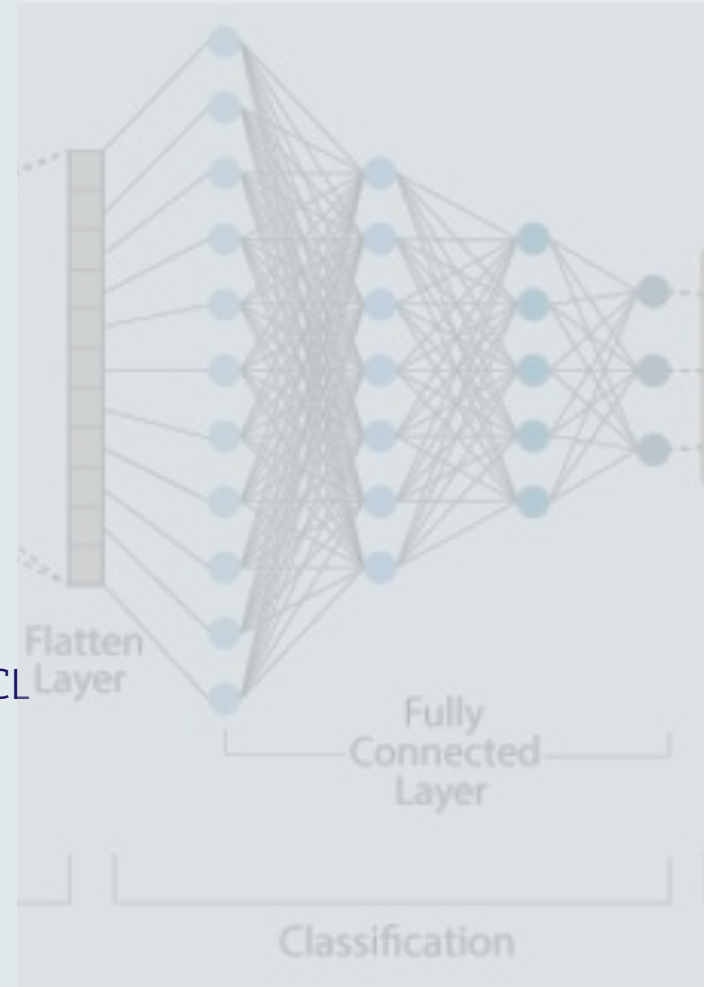
Max Pool with 2X2
window and stride 2

| | |
|---|---|
| 6 | 8 |
| 3 | 4 |

FEATURE CLASSIFICATION

Fully Connected Layer

- Prevede un vettore 1D di numeri
- Si effettua una sorta di appiattimento (FLATTEN)
- Una volta fatto il flattening dell'output del liv. di pooling finale -> diventa l'input per il FCL
- Imparano come utilizzare le feature prodotte dalle convoluzioni per classificare correttamente le immagini



FUNZIONAMENTO

```
graph TD; A[FUNZIONAMENTO] --- B[Il neurone nel fully connected layer rileva una certa feature]; A --- C[Conserva il suo valore.]; A --- D[Comunica questo valore sia alla classe "ANGRY" che alla classe "NEUTRAL".]; A --- E[Entrambe le classi controllano la funzionalità e decidono se è rilevante per loro.];
```

- Il neurone nel fully connected layer rileva una certa feature
- Conserva il suo valore.
- Comunica questo valore sia alla classe "ANGRY" che alla classe "NEUTRAL".
- Entrambe le classi controllano la funzionalità e decidono se è rilevante per loro.

FCL-CODICE

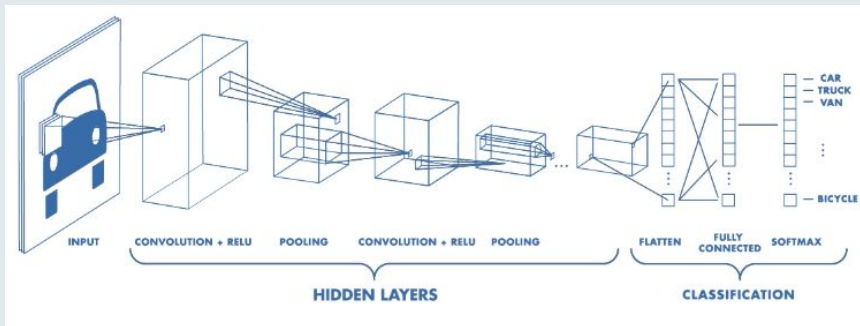
```
#Block-5
```

```
model.add(Flatten())  
model.add(Dense(64,  
kernel_initializer='he_normal'))  
model.add(Activation('relu'))  
model.add(BatchNormalization())  
model.add(Dropout(0.5))
```

```
#Block-7
```

```
model.add(Dense(7,  
kernel_initializer='he_normal'))  
model.add(Activation('softmax'))
```

HIDDEN LAYER



CONVOLUZIONE

| | | | | |
|---|---|---|---|---|
| 7 | 2 | 3 | 3 | 8 |
| 4 | 5 | 3 | 8 | 4 |
| 3 | 3 | 2 | 8 | 4 |
| 2 | 8 | 7 | 2 | 7 |
| 5 | 4 | 4 | 5 | 4 |

*

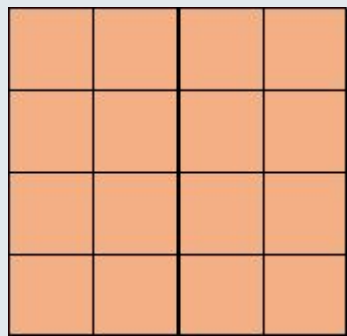
| | | |
|---|---|----|
| 1 | 0 | -1 |
| 1 | 0 | -1 |
| 1 | 0 | -1 |

=

| | | |
|---|--|--|
| 6 | | |
| | | |
| | | |

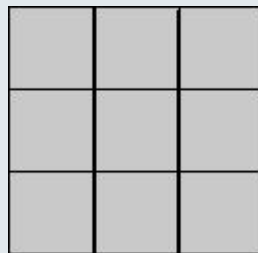
$$\begin{aligned} &7 \times 1 + 4 \times 1 + 3 \times 1 + \\ &2 \times 0 + 5 \times 0 + 3 \times 0 + \\ &3 \times -1 + 3 \times -1 + 2 \times -1 \\ &= 6 \end{aligned}$$

In particolare notiamo che:



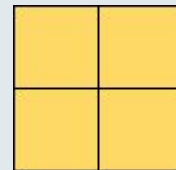
4 X 4

*



3 X 3

=



2 x 2

VOLUME
OUTPUT

$$W_{out} = \frac{(W_{in} - F + 2 \cdot \text{Padding})}{\text{Stride}} + 1$$

Volume input (W)

Volume filtro (F)

Stride

Padding

STRIDE

```
model.add(  
    Conv2D(32, (3, 3),  
    padding='same',  
    kernel_initializer='he_normal',  
    input_shape=(48, 48, 1))  
)
```

Si tratta del passo con cui facciamo scorrere il filtro.

Quando il passo è 1, spostiamo i filtri di un pixel alla volta.

Quando il passo è 2 (o di più) i filtri saltano di 2 o di più pixel alla volta.

Ciò comporta volumi di output più piccoli

Link: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D

strides

An integer or tuple/list of 2 integers, specifying the strides of the convolution along the height and width. Can be a single integer to specify the same value for all spatial dimensions. Specifying any stride value != 1 is incompatible with specifying any **dilation_rate** value != 1.

PADDING



```
graph LR; PADDING --- Valid; PADDING --- Same;
```

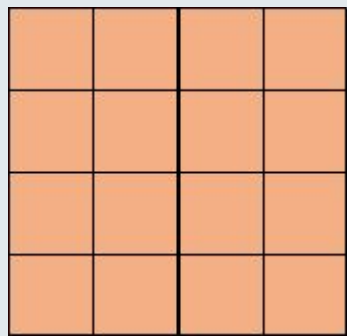
Valid

Same

```
model.add(  
    Conv2D(32, (3, 3),  
    padding='same',  
    kernel_initializer='he_normal',  
    input_shape=(48, 48, 1))  
)
```

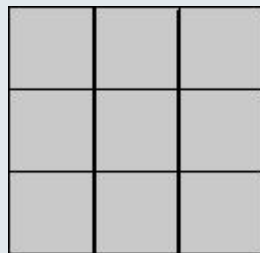
Si tratta di una cornice di zeri che si può aggiungere o meno all'input per far risultare la dimensione dell'output pari a quella dell'input.

Mettiamo in pratica la formula



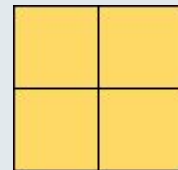
4 X 4

*



3 X 3

=



2 x 2

$$W_{out} = \frac{W_{in} - F + 2 \cdot \text{Padding}}{\text{Stride}} + 1 = \frac{4 - 3 + 2 \cdot 0}{1} + 1 = 2$$

E aggiungendo il padding...

| | | | | | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | | | | | 0 |
| 0 | | | | | 0 |
| 0 | | | | | 0 |
| 0 | | | | | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |

6 X 6

*

| | | |
|--|--|--|
| | | |
| | | |
| | | |

3 X 3

=

| | | | |
|--|--|--|--|
| | | | |
| | | | |
| | | | |
| | | | |

4 X 4

$$W_{out} = \frac{W_{in} - F + 2 \cdot \text{Padding}}{\text{Stride}} + 1 = \frac{4 - 3 + 2 \cdot 1}{1} + 1 = 4$$

VALID – no padding


```
#Block-1
model.add(Conv2D(32, (3,3),padding='valid',
kernel_initializer='he_normal',
input_shape=(48,48,1)))
#Block-2
model.add(Conv2D(64, (3,3),padding='valid',
kernel_initializer='he_normal'))
#Block-3
model.add(Conv2D(128, (3,3),padding='valid',
kernel_initializer='he_normal'))
```

| Layer (type) | Output Shape | Param # |
|-------------------|---------------------|---------|
| conv2d (Conv2D) | (None, 46, 46, 32) | 320 |
| conv2d_1 (Conv2D) | (None, 44, 44, 64) | 18496 |
| conv2d_2 (Conv2D) | (None, 42, 42, 128) | 73856 |

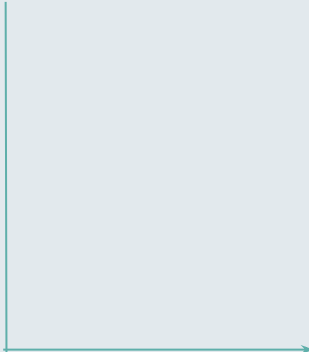
SAME – padding

```
#Block-1
model.add(Conv2D(32, (3,3),padding='same',
kernel_initializer='he_normal',
input_shape=(48,48,1)))
#Block-2
model.add(Conv2D(64, (3,3),padding='same',
kernel_initializer='he_normal'))
#Block-3
model.add(Conv2D(128, (3,3),padding='same',
kernel_initializer='he_normal'))
```

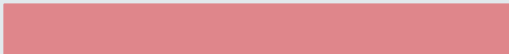
| Layer (type) | Output Shape | Param # |
|-------------------|---------------------|---------|
| conv2d_3 (Conv2D) | (None, 48, 48, 32) | 320 |
| conv2d_4 (Conv2D) | (None, 48, 48, 64) | 18496 |
| conv2d_5 (Conv2D) | (None, 48, 48, 128) | 73856 |



RELU è una funzione di attivazione



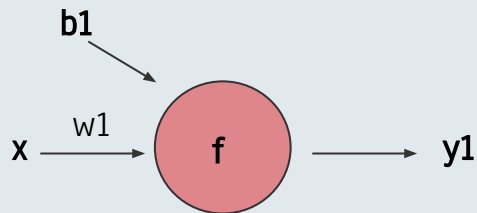
Una **funzione di attivazione**, è una funzione che viene aggiunta a una rete neurale artificiale per aiutare la rete ad apprendere modelli complessi nei dati.



Perché ci serve...

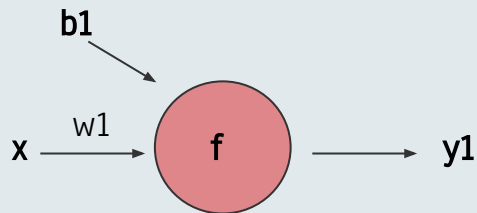
- Mantenere l'uscita limitata a un intervallo particolare.
- Introdurre la non linearità nella rete
 - Dobbiamo tener conto del tipo di dati che vogliamo modellare
 - Altrimenti porterebbero ad una profondità di un singolo strato indipendentemente dalla complessità della rete

Perchè non linearità?



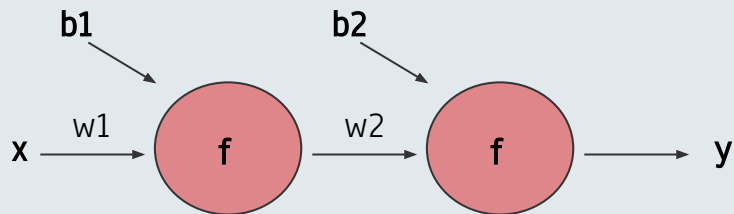
$$y1 = f(w1*x + b1)$$

Perchè non linearità?



$$y1 = f(w1*x + b1) \xrightarrow{f(x) = x} y1 = w1*x + b1$$

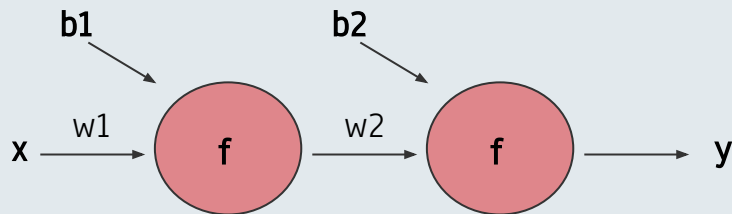
Perchè non linearità?



$$y1 = f(w1*x + b1) \xrightarrow{f(x) = x} y1 = w1*x + b1$$

$$y = f(w2*y1 + b2)$$

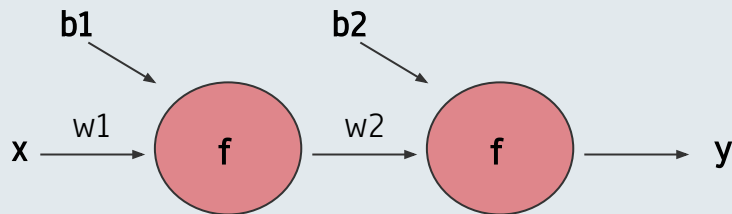
Perchè non linearità?



$$y1 = f(w1*x + b1) \xrightarrow{f(x) = x} y1 = w1*x + b1$$

$$y = f(w2*y1 + b2) = f(w2*(w1*x + b1) + b2)$$

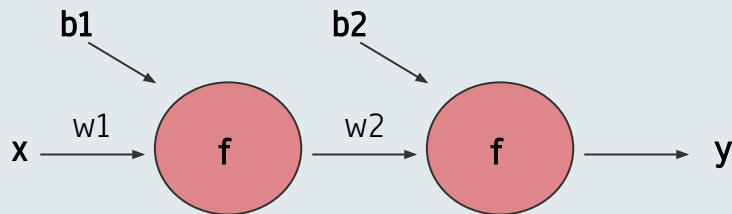
Perchè non linearità?



$$y1 = f(w1*x + b1) \xrightarrow{f(x) = x} y1 = w1*x + b1$$

$$y = f(w2*y1 + b2) = f(w2*(w1*x + b1) + b2) = f(w2*w1*x + w2*b1 + b2)$$

Perchè non linearità?



$$y1 = f(w1*x + b1) \xrightarrow{f(x) = x} y1 = w1*x + b1$$

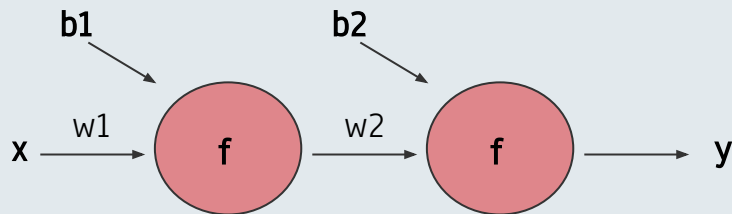
$$y = f(w2*y1 + b2) = f(w2*(w1*x + b1) + b2) = f(w2*w1*x + w2*b1 + b2)$$

$$w3 = w1*w2$$

$$b3 = w2*b1 + b2$$

$$y = f(w3*x + b3)$$

Perchè non linearità?



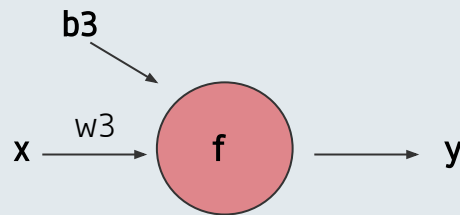
$$y1 = f(w1*x + b1) \xrightarrow{f(x) = x} y1 = w1*x + b1$$

$$y = f(w2*y1 + b2) = f(w2*(w1*x + b1) + b2) = f(w2*w1*x + w2*b1 + b2)$$

$$w3 = w1*w2$$

$$b3 = w2*b1 + b2$$

$$y = f(w3*x + b3)$$



Proprietà:



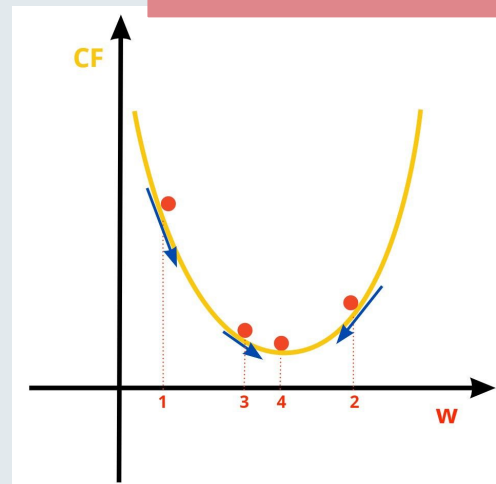
1. Problema del gradiente di fuga
2. Centrato sullo zero
3. Spese computazionali
4. Differenziabile

Fuga di gradiente

Nelle reti neurali durante la propagazione posteriore, ogni peso riceve un aggiornamento proporzionale alla derivata parziale della funzione di errore.

In alcuni casi, questo termine derivato è così piccolo da rendere gli aggiornamenti molto piccoli e quindi la convergenza sarà lenta o assente.

Allo stesso modo, se il termine derivato è molto grande, anche gli aggiornamenti saranno molto grandi. In tal caso, l'algoritmo supererà il minimo e non potrà convergere.



SOLUZIONE: scelta della funzione di attivazione corretta

Esistono molte funzioni di attivazione:

- Sigmoide
- Funzione Softmax
- Tangente Iperbolica (tanh)
- Rectifier Linear Unit
- Leaky relu
- Swish
- Elu
- Softplus and softsign
- Selu
- Attivazione lineare

Ne vediamo solo alcune, per approfondire consiglio il seguente link:

<https://netai.it/guida-rapida-alle-funzioni-di-attivazione-nel-deep-learning/#page-content>

Sigmoide

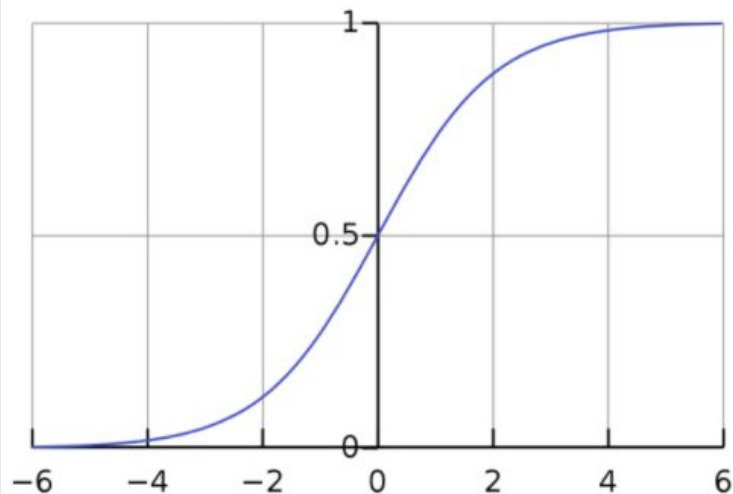
Pro:

- Limita l'uscita tra 0 e 1.
- Previsioni molto chiare per la classificazione binaria

Contro:

- Può causare il problema del gradiente in fuga
- Computazionalmente costoso
- Non è centrato sullo zero

$$S(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}$$



Softmax

Funzione comunemente utilizzata nello stato finale di una rete.

Prima di applicare softmax, alcuni componenti del vettore potrebbero essere negativi o maggiori di uno e potrebbero non sommarsi a 1 ma dopo aver applicato softmax ogni componente sarà compreso tra 0 e 1 e sommerà a 1, può quindi essere interpretato come probabilità.

The standard (unit) softmax function $\sigma : \mathbb{R}^K \rightarrow \mathbb{R}^K$ is defined by the formula

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \text{ for } i = 1, \dots, K \text{ and } \mathbf{z} = (z_1, \dots, z_K) \in \mathbb{R}^K$$

Relu (unità lineare rettificata)

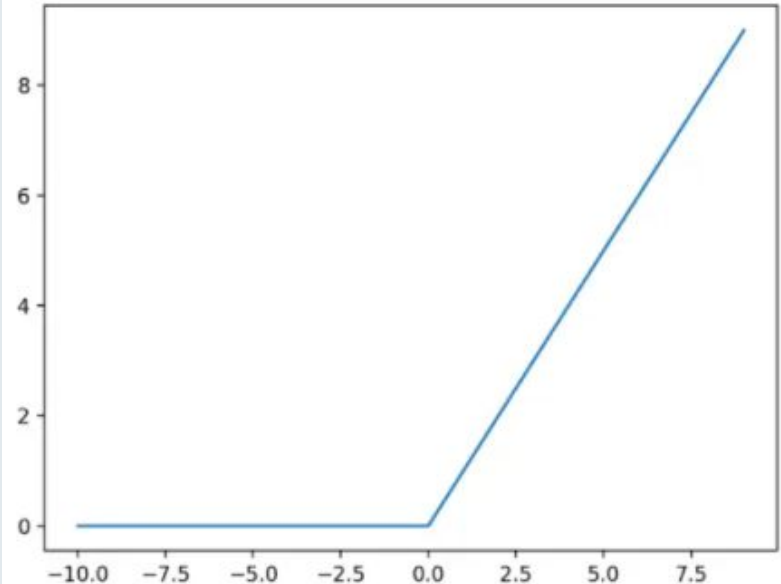
Pro:

- Facile da calcolare
- Non causa fuga di gradiente
- È veloce ed efficiente

Contro:

- Può uccidere dei neuroni per sempre

$$f(x) = x^+ = \max(0, x),$$



Esempio

Input



Relu



Quali funzioni utilizzare?...



RELU dovrebbe essere preferito per i livelli nascosti. Se sta causando il problema di relu morente, è necessario utilizzare le sue modifiche come **Leaky Relu**, **ELU**, **SELU**, ecc.

Per le reti profonde, **SWISH** ha prestazioni migliori di relu.

Negli ultimi livelli, per normalizzare gli output della rete è consigliabile usare **Softmax**.

Sigmoide e tanh dovrebbero essere evitati a causa del problema della fuga del gradiente.



#Aggiunta al modello della funzione di attivazione ReLU nei primi 6 blocchi della rete.

```
model.add(Activation('relu'))
```

#Nell'ultimo blocco abbiamo aggiunto la funzione di attivazione Softmax

```
model.add(Activation('softmax'))
```

TIPI DI POOLING

```
graph LR; A[TIPI DI POOLING] --- B[• MAX-POOLING]; A --- C[• AVERAGE-POOLING]; A --- D[• MIN-POOLING];
```

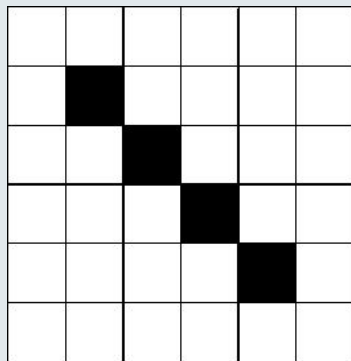
• MAX-POOLING

• AVERAGE-POOLING

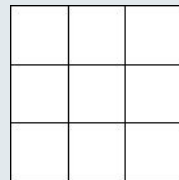
• MIN-POOLING

MAX-POOLING

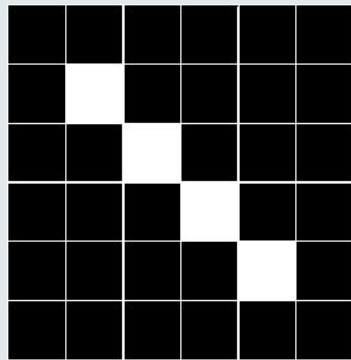
| | | | | | |
|-----|-----|-----|-----|-----|-----|
| 255 | 255 | 255 | 255 | 255 | 255 |
| 255 | 0 | 255 | 255 | 255 | 255 |
| 255 | 255 | 0 | 255 | 255 | 255 |
| 255 | 255 | 255 | 0 | 255 | 255 |
| 255 | 255 | 255 | 255 | 0 | 255 |
| 255 | 255 | 255 | 255 | 255 | 255 |



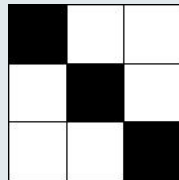
| | | |
|-----|-----|-----|
| 255 | 255 | 255 |
| 255 | 255 | 255 |
| 255 | 255 | 255 |



| | | | | | |
|---|-----|-----|-----|-----|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 255 | 0 | 0 | 0 | 0 |
| 0 | 0 | 255 | 0 | 0 | 0 |
| 0 | 0 | 0 | 255 | 0 | 0 |
| 0 | 0 | 0 | 0 | 255 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |

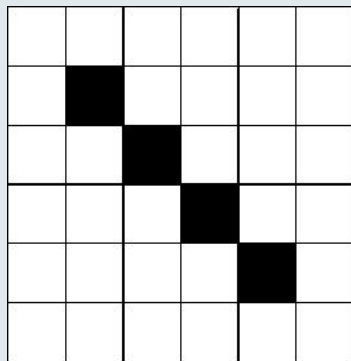


| | | |
|-----|-----|-----|
| 255 | 0 | 0 |
| 0 | 255 | 0 |
| 0 | 0 | 255 |

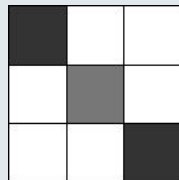


AVERAGE-POOLING

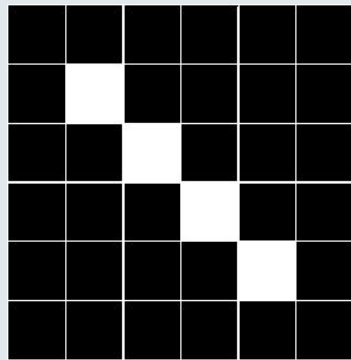
| | | | | | |
|-----|-----|-----|-----|-----|-----|
| 255 | 255 | 255 | 255 | 255 | 255 |
| 255 | 0 | 255 | 255 | 255 | 255 |
| 255 | 255 | 0 | 255 | 255 | 255 |
| 255 | 255 | 255 | 0 | 255 | 255 |
| 255 | 255 | 255 | 255 | 0 | 255 |
| 255 | 255 | 255 | 255 | 255 | 255 |



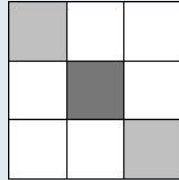
| | | |
|-----|-----|-----|
| 191 | 255 | 255 |
| 255 | 127 | 255 |
| 255 | 255 | 191 |



| | | | | | |
|---|-----|-----|-----|-----|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 255 | 0 | 0 | 0 | 0 |
| 0 | 0 | 255 | 0 | 0 | 0 |
| 0 | 0 | 0 | 255 | 0 | 0 |
| 0 | 0 | 0 | 0 | 255 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |

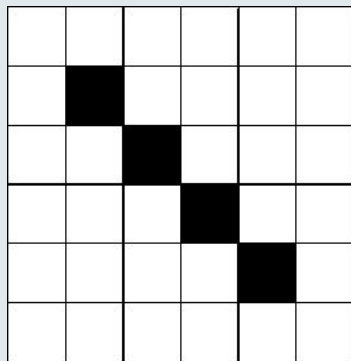


| | | |
|----|-----|----|
| 63 | 0 | 0 |
| 0 | 127 | 0 |
| 0 | 0 | 63 |

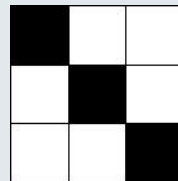


MIN-POOLING

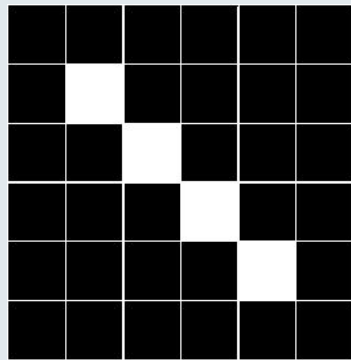
| | | | | | |
|-----|-----|-----|-----|-----|-----|
| 255 | 255 | 255 | 255 | 255 | 255 |
| 255 | 0 | 255 | 255 | 255 | 255 |
| 255 | 255 | 0 | 255 | 255 | 255 |
| 255 | 255 | 255 | 0 | 255 | 255 |
| 255 | 255 | 255 | 255 | 0 | 255 |
| 255 | 255 | 255 | 255 | 255 | 255 |



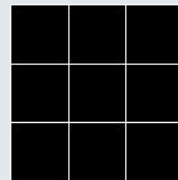
| | | |
|-----|-----|-----|
| 0 | 255 | 255 |
| 255 | 0 | 255 |
| 255 | 255 | 0 |



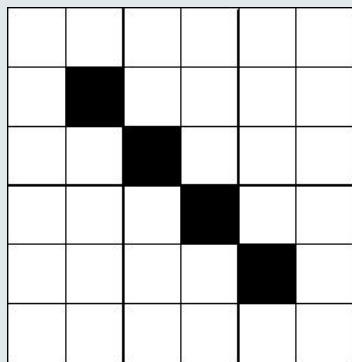
| | | | | | |
|---|-----|-----|-----|-----|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 255 | 0 | 0 | 0 | 0 |
| 0 | 0 | 255 | 0 | 0 | 0 |
| 0 | 0 | 0 | 255 | 0 | 0 |
| 0 | 0 | 0 | 0 | 255 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |



| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |



| | | | | | |
|-----|-----|-----|-----|-----|-----|
| 255 | 255 | 255 | 255 | 255 | 255 |
| 255 | 0 | 255 | 255 | 255 | 255 |
| 255 | 255 | 0 | 255 | 255 | 255 |
| 255 | 255 | 255 | 0 | 255 | 255 |
| 255 | 255 | 255 | 255 | 0 | 255 |
| 255 | 255 | 255 | 255 | 255 | 255 |

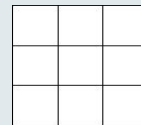


Max-pooling

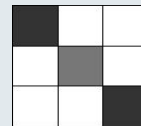
Average-pooling

Min-pooling

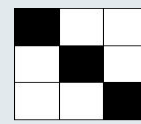
| | | |
|-----|-----|-----|
| 255 | 255 | 255 |
| 255 | 255 | 255 |
| 255 | 255 | 255 |



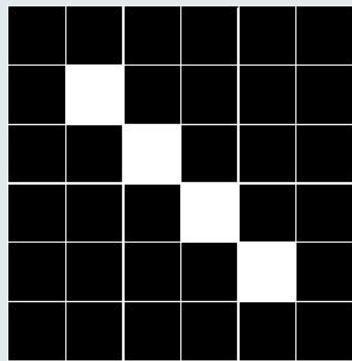
| | | |
|-----|-----|-----|
| 191 | 255 | 255 |
| 255 | 127 | 255 |
| 255 | 255 | 191 |



| | | |
|-----|-----|-----|
| 0 | 255 | 255 |
| 255 | 0 | 255 |
| 255 | 255 | 0 |



| | | | | | |
|---|-----|-----|-----|-----|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 255 | 0 | 0 | 0 | 0 |
| 0 | 0 | 255 | 0 | 0 | 0 |
| 0 | 0 | 0 | 255 | 0 | 0 |
| 0 | 0 | 0 | 0 | 255 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |

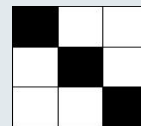


Max-pooling

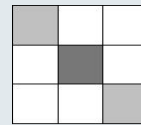
Average-pooling

Min-pooling

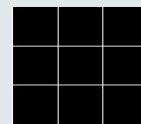
| | | |
|-----|-----|-----|
| 255 | 0 | 0 |
| 0 | 255 | 0 |
| 0 | 0 | 255 |



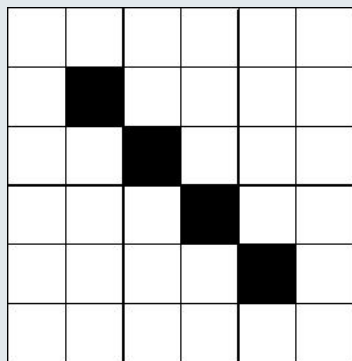
| | | |
|----|-----|----|
| 63 | 0 | 0 |
| 0 | 127 | 0 |
| 0 | 0 | 63 |



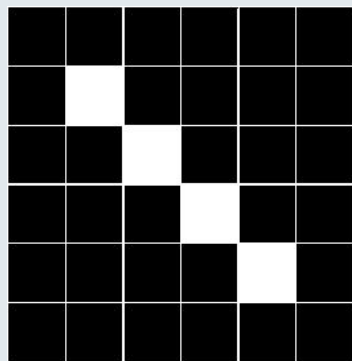
| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |



| | | | | | |
|-----|-----|-----|-----|-----|-----|
| 255 | 255 | 255 | 255 | 255 | 255 |
| 255 | 0 | 255 | 255 | 255 | 255 |
| 255 | 255 | 0 | 255 | 255 | 255 |
| 255 | 255 | 255 | 0 | 255 | 255 |
| 255 | 255 | 255 | 255 | 0 | 255 |
| 255 | 255 | 255 | 255 | 255 | 255 |

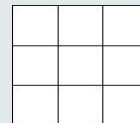


| | | | | | |
|---|-----|-----|-----|-----|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 255 | 0 | 0 | 0 | 0 |
| 0 | 0 | 255 | 0 | 0 | 0 |
| 0 | 0 | 0 | 255 | 0 | 0 |
| 0 | 0 | 0 | 0 | 255 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |



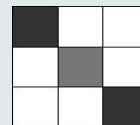
Max-pooling

| | | |
|-----|-----|-----|
| 255 | 255 | 255 |
| 255 | 255 | 255 |
| 255 | 255 | 255 |



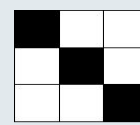
Average-pooling

| | | |
|-----|-----|-----|
| 191 | 255 | 255 |
| 255 | 127 | 255 |
| 255 | 255 | 191 |



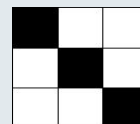
Min-pooling

| | | |
|-----|-----|-----|
| 0 | 255 | 255 |
| 255 | 0 | 255 |
| 255 | 255 | 0 |



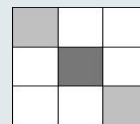
Max-pooling

| | | |
|-----|-----|-----|
| 255 | 0 | 0 |
| 0 | 255 | 0 |
| 0 | 0 | 255 |



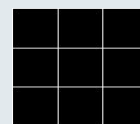
Average-pooling

| | | |
|----|-----|----|
| 63 | 0 | 0 |
| 0 | 127 | 0 |
| 0 | 0 | 63 |



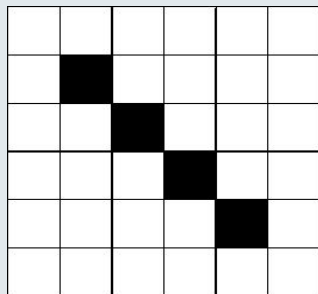
Min-pooling

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |



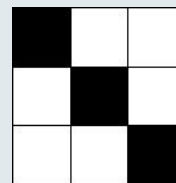
COSA NOTIAMO?

| | | | | | |
|-----|-----|-----|-----|-----|-----|
| 255 | 255 | 255 | 255 | 255 | 255 |
| 255 | 0 | 255 | 255 | 255 | 255 |
| 255 | 255 | 0 | 255 | 255 | 255 |
| 255 | 255 | 255 | 0 | 255 | 255 |
| 255 | 255 | 255 | 255 | 0 | 255 |
| 255 | 255 | 255 | 255 | 255 | 255 |



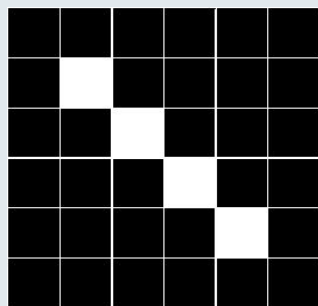
Min-pooling

| | | |
|-----|-----|-----|
| 0 | 255 | 255 |
| 255 | 0 | 255 |
| 255 | 255 | 0 |



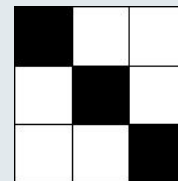
Il raggruppamento minimo offre risultati migliori per le immagini con sfondo bianco e oggetto nero

| | | | | | |
|---|-----|-----|-----|-----|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 255 | 0 | 0 | 0 | 0 |
| 0 | 0 | 255 | 0 | 0 | 0 |
| 0 | 0 | 0 | 255 | 0 | 0 |
| 0 | 0 | 0 | 0 | 255 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |



Max-pooling

| | | |
|-----|-----|-----|
| 255 | 0 | 0 |
| 0 | 255 | 0 |
| 0 | 0 | 255 |

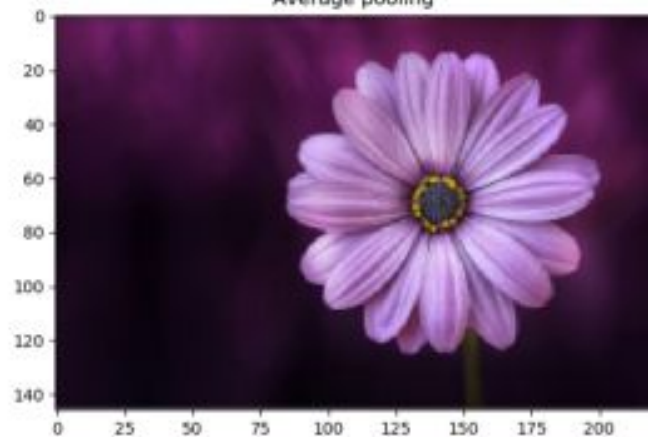


Il raggruppamento massimo offre risultati migliori per le immagini con sfondo nero e oggetto bianco

Original Image



Average pooling



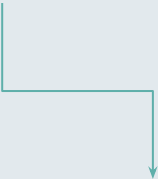
Max pooling



Min pooling



Viene utilizzato il **pooling massimo** perchè lo sfondo in queste immagini viene reso nero per ridurre il costo di calcolo.



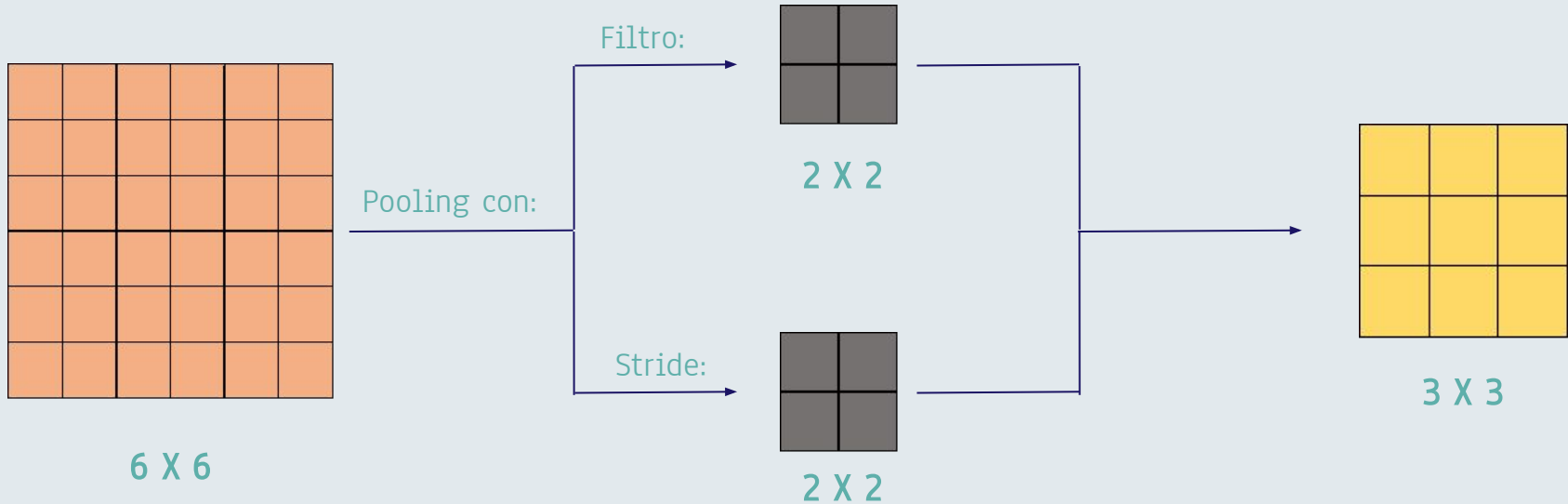
```
graph TD; A[Viene utilizzato il pooling massimo perchè lo sfondo in queste immagini viene reso nero per ridurre il costo di calcolo.] --> B[NERO = 0<br/>BIANCO=255];
```

NERO = 0
BIANCO=255



```
model.add(MaxPooling2D(pool_size=(2,2)))
```

NOTIAMO ANCHE CHE:



Anche qui si potrebbe usare il padding, ma non lo faremo per ridurre il numero di dati da trasportare nella nostra rete.

Quindi:

| Layer (type) | Output Shape | Param # |
|-------------------------------------|--------------------|---------|
| conv2d_6 (Conv2D) | (None, 48, 48, 32) | 320 |
| activation_18 (Activation) | (None, 48, 48, 32) | 0 |
| max_pooling2d_20 (MaxPoolin g2D) | (None, 24, 24, 32) | 0 |
| conv2d_7 (Conv2D) | (None, 24, 24, 64) | 18496 |
| activation_19 (Activation) | (None, 24, 24, 64) | 0 |
| max_pooling2d_21 (MaxPoolin g2D) | (None, 12, 12, 64) | 0 |



04

**ALGORITMI
DI OTTIMIZZAZIONE**

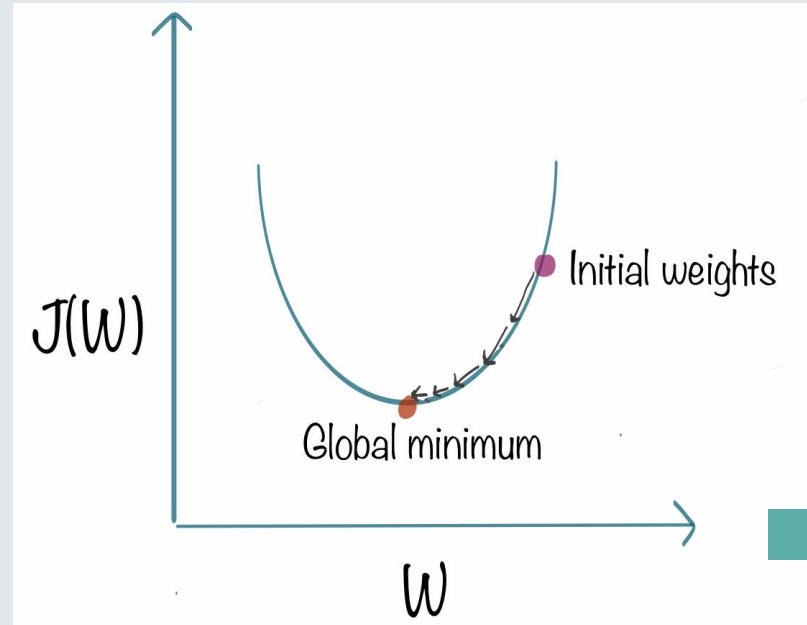
COSA SONO?

Algoritmi matematici che permettono di trovare punti di minimo della funzione di costo (cost function) per aggiornare i parametri della rete (pesi e bias)

DISCESA DEL GRADIENTE

Metodo utilizzato per calcolare il minimo della funzione di costo e calcolare l'errore di predizione dei neuroni della rete per fare l'aggiornamento dei suoi parametri.

$C = (y^{\wedge} - y)^2$, dove y^{\wedge} è l'output predetto e y la classificazione corretta.



BATCH SIZE E EPOCHS

Sono due iperparametri della rete neurale che servono all'algoritmo di ottimizzazione (GD) per ottimizzare al meglio il training del modello.

BATCH SIZE

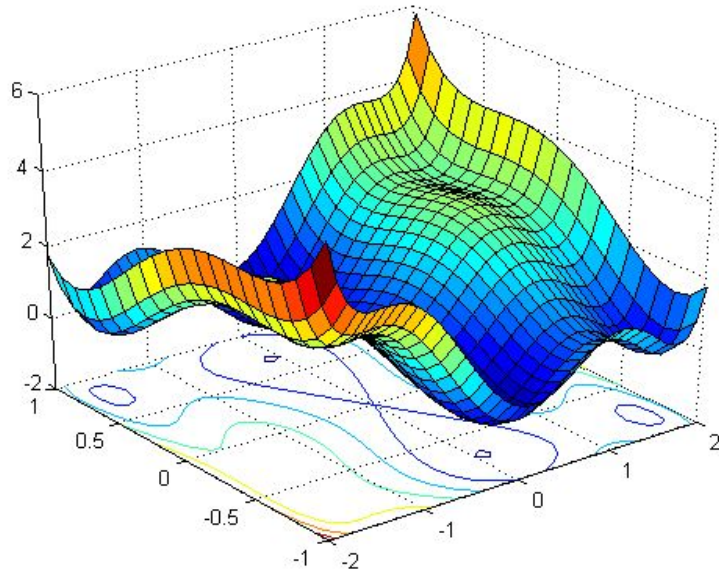
Rappresenta quanti esempi/data points vengono mostrati e analizzati dal modello prima che esso faccia l'update dei suoi parametri.

EPOCHS

Il numero di epoche è un iperparametro che definisce il numero di volte a cui il modello sarà sottoposto all'intero dataset (training set), cioè quante volte il modello analizzerà tutti gli esempi presenti nel training set.

LOSS (loss e val_loss)

La loss function (o funzione di costo) è una funzione che *valuta le prestazioni* di un modello. Rappresenta una perdita e quindi l'obiettivo di un buon apprendimento è la *minimizzazione* della loss function.



ACCURACY (accuracy e val_accuracy)

In un sistema di classificazione, i campioni di test vengono forniti al modello e vengono conteggiati gli esempi correttamente elaborati. Infine viene calcolata l'accuratezza come rapporto tra i campioni correttamente elaborati e il totale dei campioni inviati al modello.

$$accuracy = \frac{pattern_{OK}}{pattern_{TOT}}$$

ESEMPIO

```
model.compile(loss='categorical_crossentropy', optimizer=Adam(learning_rate=0.001), metrics=['accuracy'])
```

```
model.fit(train_X, train_Y, batch_size=64, epochs=25, validation_data=(test_X, test_Y))
```

```
loss_and_accuracy = model.evaluate(test_X, test_Y)  
print(loss_and_metrics)
```

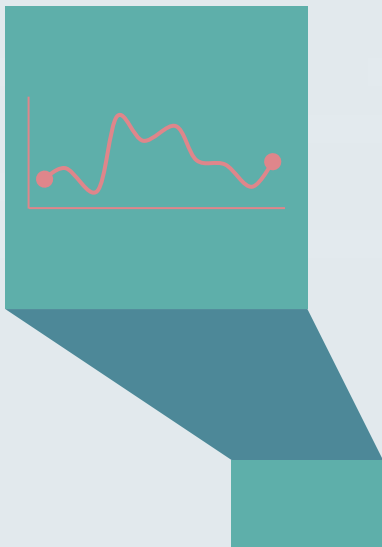
ESEMPIO

```
↳ Epoch 1/100
898/898 [=====] - 34s 35ms/step - loss: 2.0711 - accuracy: 0.2362 - val_loss: 1.6500 - val_accuracy: 0.3433
Epoch 2/100
898/898 [=====] - 31s 34ms/step - loss: 1.5726 - accuracy: 0.3886 - val_loss: 1.3413 - val_accuracy: 0.4851
Epoch 3/100
898/898 [=====] - 31s 34ms/step - loss: 1.3980 - accuracy: 0.4675 - val_loss: 1.2741 - val_accuracy: 0.5093
Epoch 4/100
898/898 [=====] - 31s 34ms/step - loss: 1.3111 - accuracy: 0.5063 - val_loss: 1.1768 - val_accuracy: 0.5525
Epoch 5/100
898/898 [=====] - 31s 34ms/step - loss: 1.2413 - accuracy: 0.5375 - val_loss: 1.2091 - val_accuracy: 0.5286
Epoch 6/100
898/898 [=====] - 31s 34ms/step - loss: 1.1888 - accuracy: 0.5571 - val_loss: 1.0974 - val_accuracy: 0.5818
Epoch 7/100
898/898 [=====] - 31s 34ms/step - loss: 1.1325 - accuracy: 0.5883 - val_loss: 1.0768 - val_accuracy: 0.5949
Epoch 8/100
898/898 [=====] - 31s 34ms/step - loss: 1.0956 - accuracy: 0.6011 - val_loss: 1.0624 - val_accuracy: 0.6013
Epoch 9/100
898/898 [=====] - 31s 34ms/step - loss: 1.0457 - accuracy: 0.6229 - val_loss: 1.1232 - val_accuracy: 0.5837
Epoch 10/100
898/898 [=====] - 31s 34ms/step - loss: 1.0002 - accuracy: 0.6409 - val_loss: 1.0307 - val_accuracy: 0.6144
Epoch 11/100
898/898 [=====] - 31s 34ms/step - loss: 0.9470 - accuracy: 0.6625 - val_loss: 1.0159 - val_accuracy: 0.6339
Epoch 12/100
898/898 [=====] - 31s 35ms/step - loss: 0.9002 - accuracy: 0.6795 - val_loss: 1.0054 - val_accuracy: 0.6358
Epoch 13/100
898/898 [=====] - 31s 35ms/step - loss: 0.8594 - accuracy: 0.6967 - val_loss: 1.0261 - val_accuracy: 0.6336
Epoch 14/100
898/898 [=====] - 31s 35ms/step - loss: 0.8119 - accuracy: 0.7185 - val_loss: 1.0128 - val_accuracy: 0.6400
Epoch 15/100
898/898 [=====] - 31s 35ms/step - loss: 0.7613 - accuracy: 0.7358 - val_loss: 1.0117 - val_accuracy: 0.6434
Epoch 16/100
```

DISCESA DEL GRADIENTE

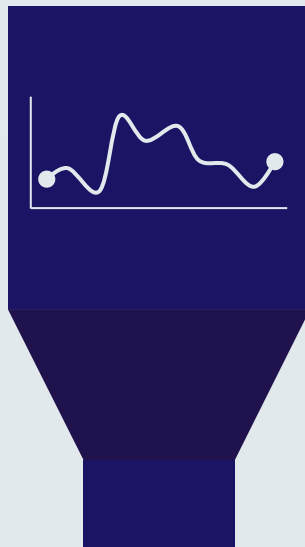
BGD

Batch
Gradient Descent



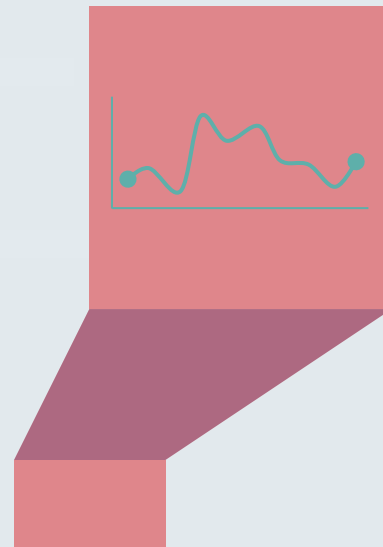
MBGD

Mini-Batch
Gradient Descent



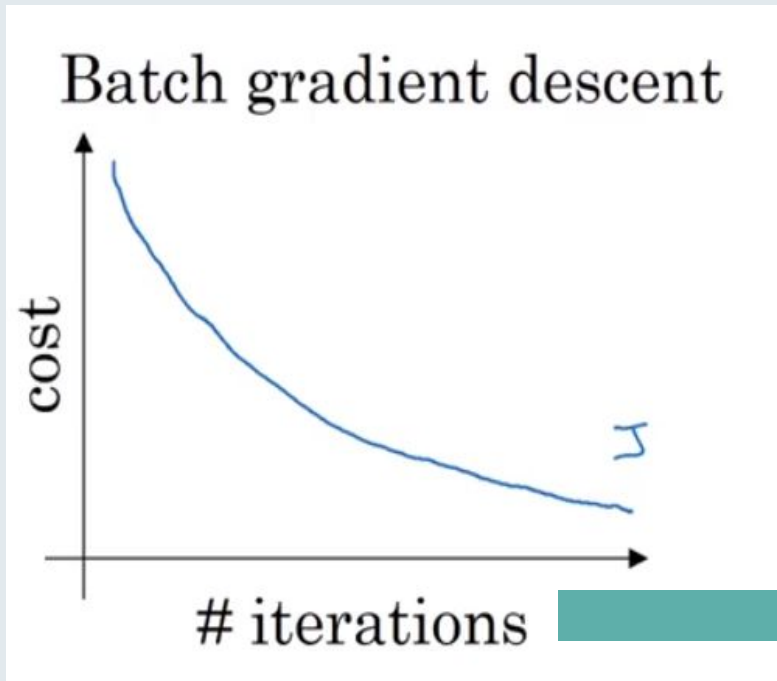
SGD

Stochastic
Gradient Descent



BGD (Batch Gradient Descent)

Prevede l'analisi di tutti gli esempi del dataset da parte del modello prima di aggiornare i parametri. La funzione di loss decresce più lentamente ma con poche oscillazioni. La rete *apprende meglio*.



CONTRO

Serve una maggiore
potenza computazionale



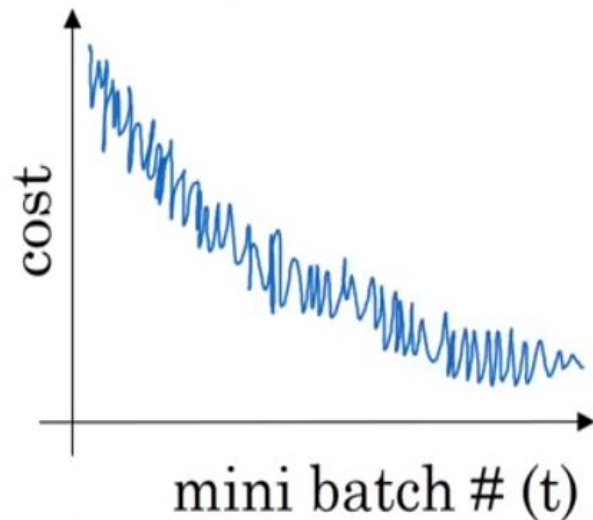
PRO

Il modello apprende
dall'intero dataset

MBGD (Mini-Batch Gradient Descent)

Divide i dati del dataset in gruppi (batch) e fa allenare la rete sottoponendo un gruppo alla volta di esempi, *dopo ogni batch* si aggiorneranno i pesi calcolando l'errore medio della cost function.

Mini-batch gradient descent



CONTRO

Non raggiunge la stessa
precisione di BGD

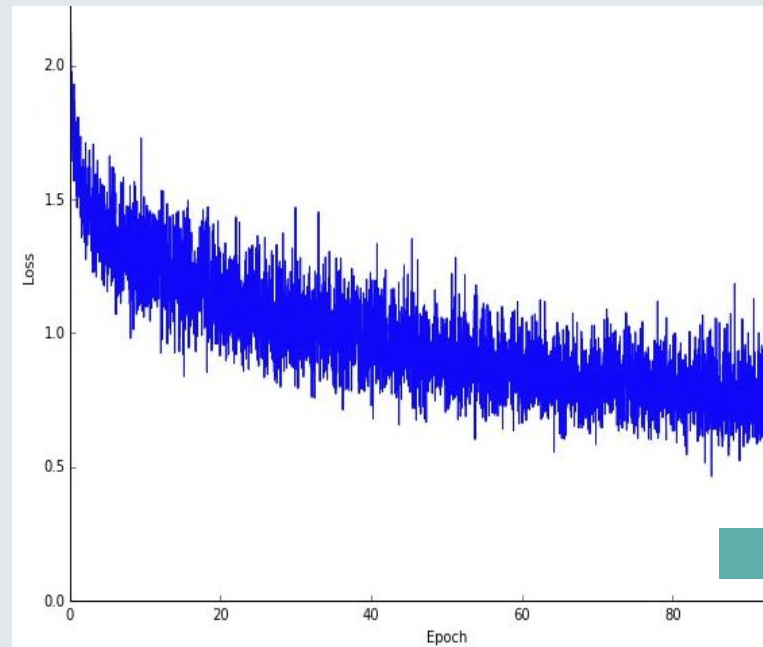


PRO

E' la soluzione migliore come rapporto
potenza computazionale/convergenza

SGD (Stochastic Gradient Descent)

I pesi della rete vengono aggiornati facendo analizzare solamente *un esempio per volta*, la rete perciò non apprende dall'intero dataset. Non si avrà mai una totale convergenza della loss function.



CONTRO

Poco preciso e non raggiunge una
convergenza totale



PRO

Algoritmo veloce ad apprendere in quanto i
pesi vengono aggiornati ad ogni esempio

CONFRONTO



- Batch gradient descent
- Mini-batch gradient Descent
- Stochastic gradient descent



BATCH NORMALIZATION

E' una tecnica per l'addestramento di DNN che standardizza gli input/output ad ogni livello della rete per stabilizzare il processo di apprendimento ed evitare sbilanciamenti di pesi tra i diversi layer.

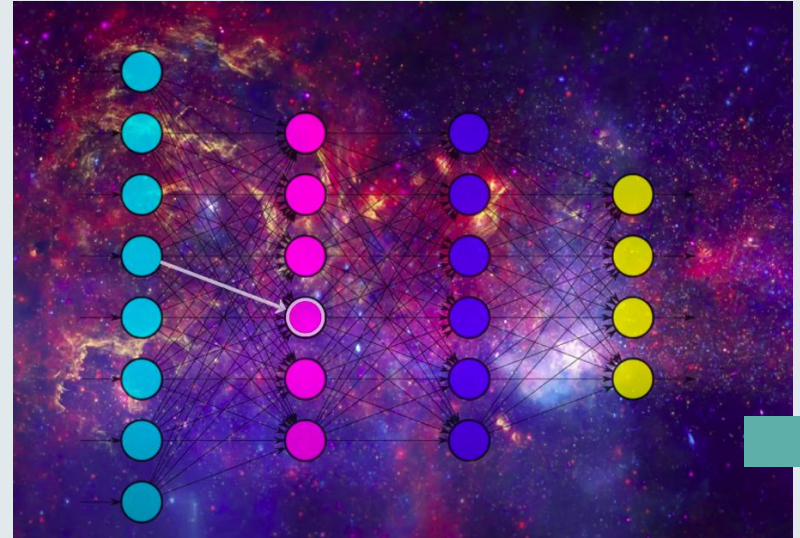
Evita il fenomeno noto come ***internal covariate shift***

INTERNAL COVARIATE SHIFT

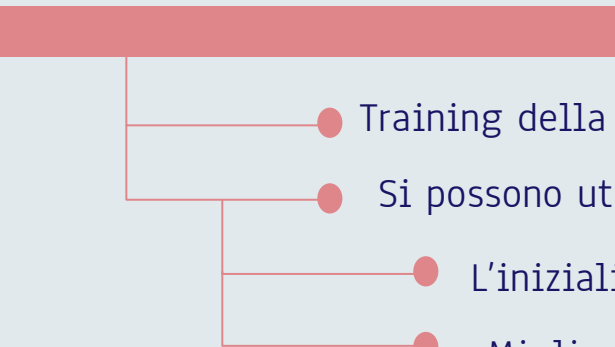

Di cosa si tratta?

Con *internal covariate shift* ci si riferisce al fatto che ogni strato della rete, ad ogni iterazione, viene esposto ad un input completamente *diverso* rispetto all'iterazione precedente.

Il cambiamento degli input ad ogni strato rappresenta un problema perchè gli strati della rete devono continuamente adattarsi ad una nuova distribuzione dei valori di ingresso, quindi, per migliorare la fase di training, occorre fissare questa distribuzione.



QUALI VANTAGGI...

- 
- Training della rete più veloce
 - Si possono utilizzare tassi di apprendimento più alti
 - L'inizializzazione dei pesi della rete incide meno sul tempo
 - Miglioramento delle performance in generale
- 

IMPLEMENTAZIONE

$$\mu_B \leftarrow \frac{1}{m} \sum_{i=1}^m x_i$$

Si trova la deviazione standard del batch

$$\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

Il valore viene moltiplicato per Gamma e sommato a Beta

1°

2°

3°

4°

Si trova il valore medio per il batch corrente

$$\sigma_B^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2$$

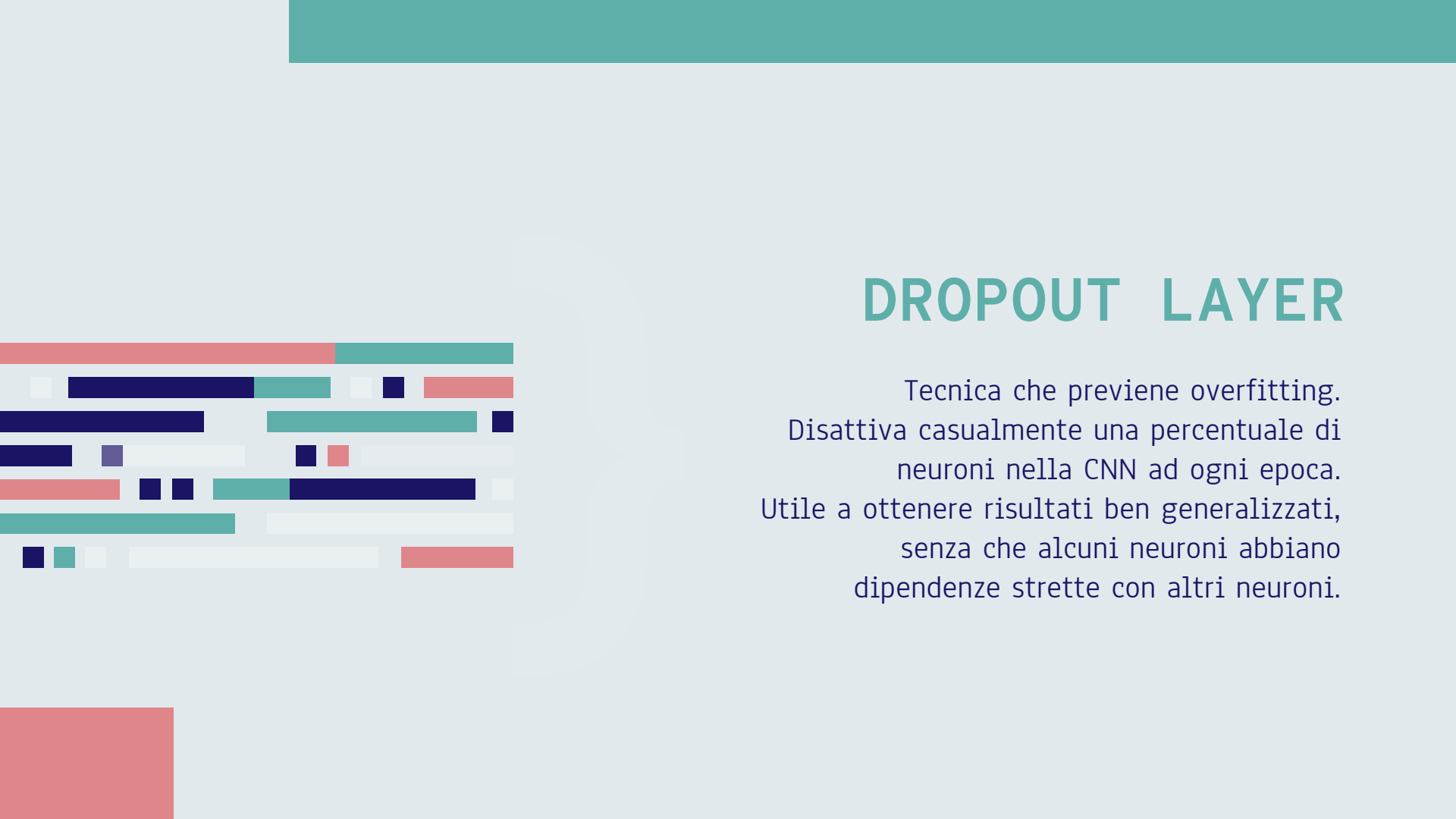
Si normalizza il valore con l'equazione

$$y_i \leftarrow \gamma \hat{x}_i + \beta$$



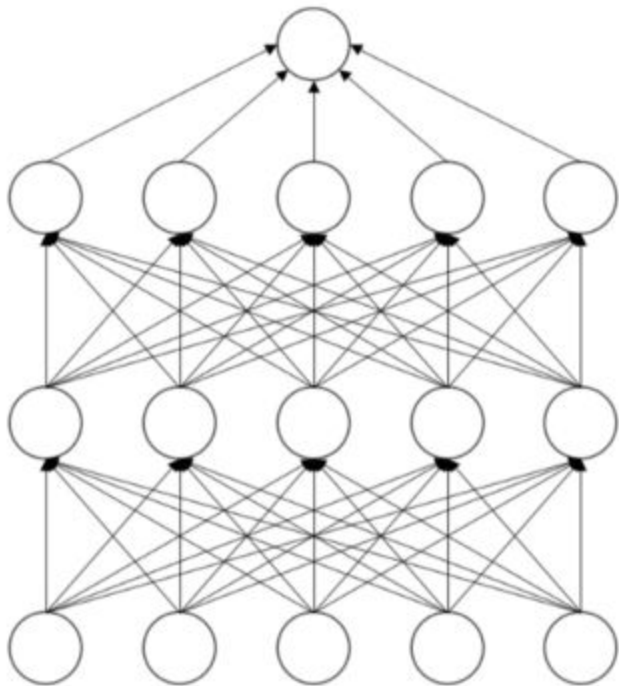
05

FEATURE CLASSIFICATION

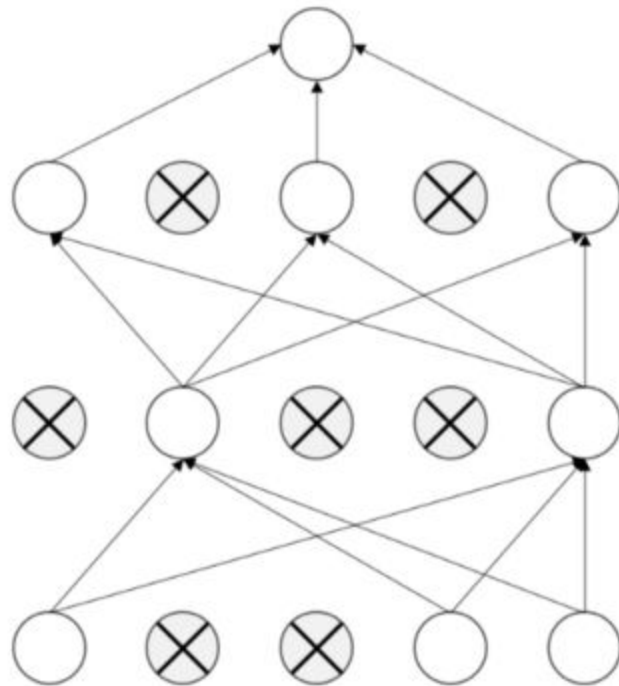


DROPOUT LAYER

Tecnica che previene overfitting.
Disattiva casualmente una percentuale di
neuroni nella CNN ad ogni epoca.
Utile a ottenere risultati ben generalizzati,
senza che alcuni neuroni abbiano
dipendenze strette con altri neuroni.



Standard Neural Net



After applying dropout

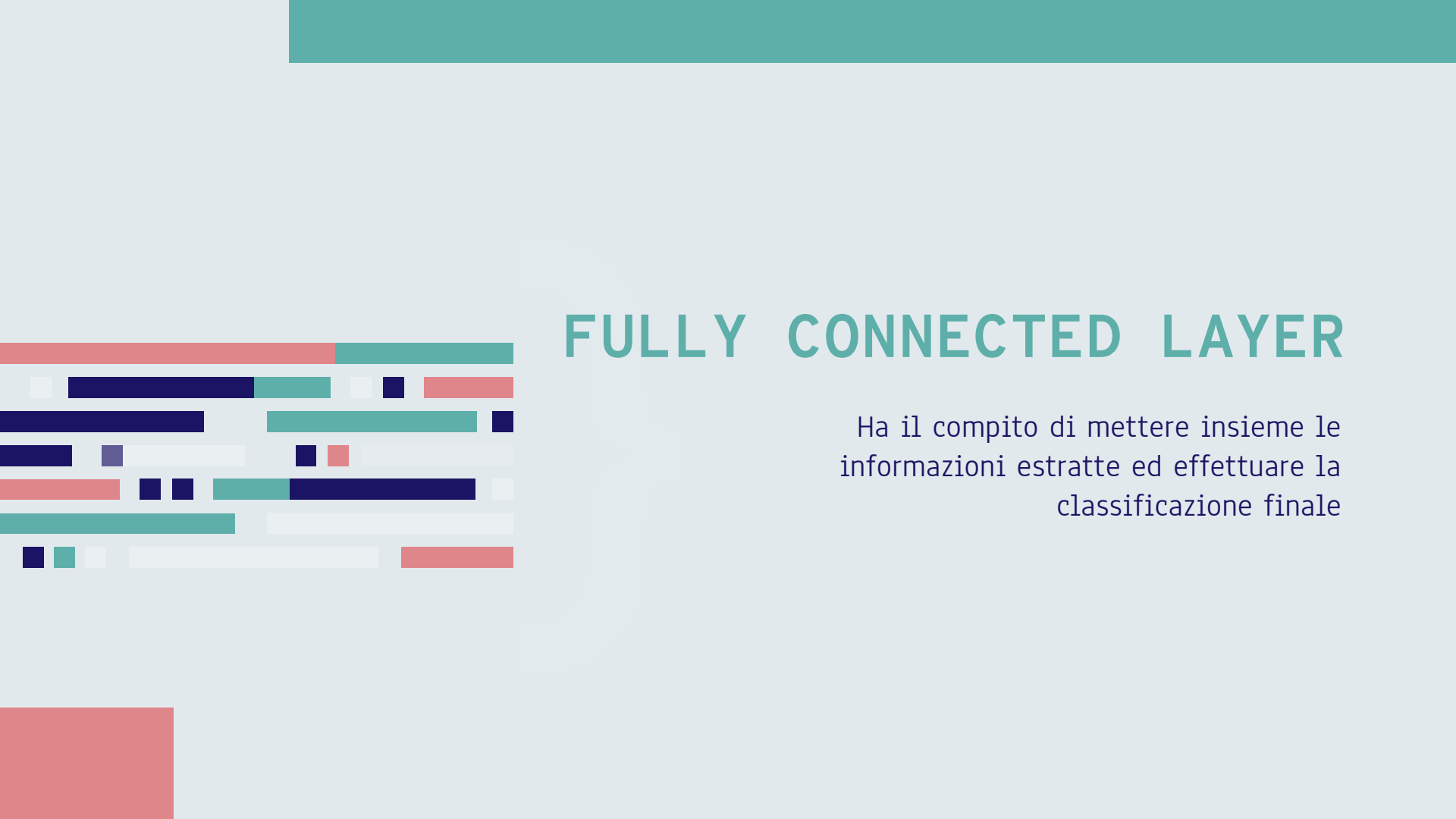
DROPOUT LAYER

CONTRO

Funzione di costo non ben definita

PRO

Generalizza meglio il modello



FULLY CONNECTED LAYER

Ha il compito di mettere insieme le informazioni estratte ed effettuare la classificazione finale

SOFTMAX LAYER

Viene applicata la funzione di Softmax come ultima funzione per normalizzare i pesi tra 0 e 1 alla fine del training.

Il suo scopo è di rendere i pesi delle probabilità, enfatizzando i valori più grandi, nascondendo quelli più piccoli rispetto al valore massimo

SOFTMAX LAYER(2)

La funzione di softmax ha la proprietà di trasformare alcuni componenti vettoriali che non sono nell'intervallo 0 e 1 in un valore nell'intervallo, facendo in modo che la somma di tutti i valori sia 1



06

IMPLEMENTAZIONE

In Google Colab

Riferimenti

- <https://andreaprovino.it/vgg-16-cnn-networks-deep-learning-engineer-italia/>
- <https://www.kaggle.com/deadskull17/fer2013>
- <https://viso.ai/computer-vision/deepface/>
- <https://neurohive.io/en/popular-networks/vgg16/>
- <https://analyticsindiamag.com/my-first-cnn-project-emotion-detection-using-convolutional-neural-network-with-tpu/>
- <https://netai.it/guida-rapida-alle-funzioni-di-attivazione-nel-deep-learning/#page-content>
- <https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/>
- <https://medium.com/swlh/emotion-detection-using-opencv-and-keras-771260bbd7f7>
- <https://medium.com/themlblog/how-to-do-facial-emotion-recognition-using-a-cnn-b7bbae79cd8f>
- <https://analyticsindiamag.com/my-first-cnn-project-emotion-detection-using-convolutional-neural-network-with-tpu/>
- <https://www.developersmaggioli.it/blog/ia-un-po-di-nozioni-prima-della-pratica/>
- <https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2#7d8a>

Riferimenti (2)

- <https://www.youtube.com/watch?v=yN7qfBhfGqs>
- <https://medium.com/@alessandropadrin/come-velocizzare-il-training-delle-reti-neurali-5b66e3aa6a82>
- https://en.wikipedia.org/wiki/Activation_function
- <https://qastack.it/programming/9782071/why-must-a-nonlinear-activation-function-be-used-in-a-backpropagation-neural-net>
- https://it.wikipedia.org/wiki/Problema_della_scomparsa_del_gradiente
- <https://ichi.pro/it/funzioni-di-attivazione-relu-e-softmax-148521511785096>
- <https://ichi.pro/it/capire-relu-la-funzione-di-attivazione-piu-popolare-in-5-minuti-4784117649999>
- <https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/>
- https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D
- <https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/>
- <https://123dok.org/article/pooling-layer-utilizzo-layer-convoluzionali-cnn.q7wx29oo>
- <https://medium.com/@bdhuma/which-pooling-method-is-better-maxpooling-vs-minpooling-vs-average-pooling-95fb03f45a9>

GRAZIE PER L'ATTENZIONE!

Riccardo Tenuta,

Chiara Bublil,

Jahaira Sulca,

Alex Zani