UNIVERSITÀ DEGLI STUDI DI MILANO

DENOISING tramite RETI NEURALI - N2V

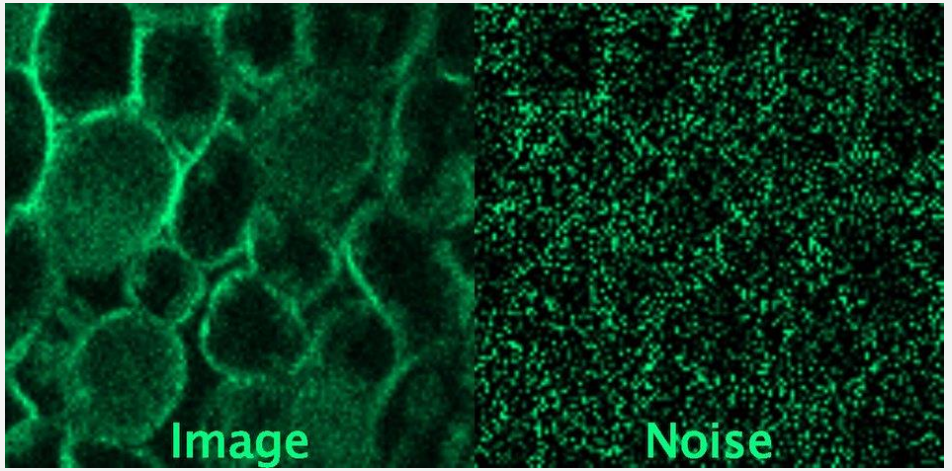
Corti Filippo
Dal Santo Giorgio
Donato Carlotta

Link di GitHub: https://github.com/Filippo-Corti/PrincipiEModelliDellaPercezione

Il Rumore nelle Immagini

Cos'è il **Rumore**:

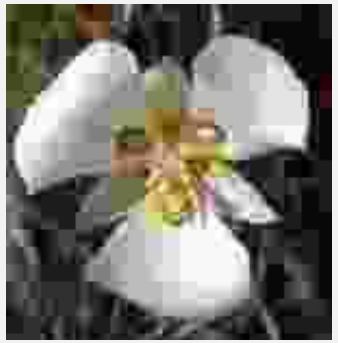
- **Distorsione casuale** che altera i pixel, riducendo la chiarezza dell'immagine.
- **Artefatto** che può influenzare l'analisi dei dati.
- Compromette **nitidezza, contrasto e dettagli** dell'immagine.



Unstructured noise

Come si forma il rumore

- **Rumore da sensore**: I sensori non catturano tutte le informazioni luminose.
- **Interferenze elettroniche**: Disturbi nei circuiti elettronici che alterano i segnali.
- **Limitazioni hardware**: Qualità inferiore dei dispositivi di acquisizione (bassa risoluzione).
- **Condizioni ambientali**: Fluttuazioni di temperatura, illuminazione instabile o vibrazioni.
- **Compressione dei dati**: Algoritmi di compressione che riducono la dimensione del file ma causano artefatti.



Result of heavy JPEG compression

Classificazione del Rumore

1. **Rumore Strutturato**: configurazione regolare o pattern riconoscibile, facilmente identificabile.
    - *Rumore Sale e Pepe*: pixel bianchi e neri casuali sparsi nell'immagine.
    - *Rumore a Bande* (Banding Noise): linee orizzontali o verticali regolari nell'immagine.

Example of Salt and Pepper Noise

2. Rumore Non Strutturato: completamente casuale e non segue alcun pattern.

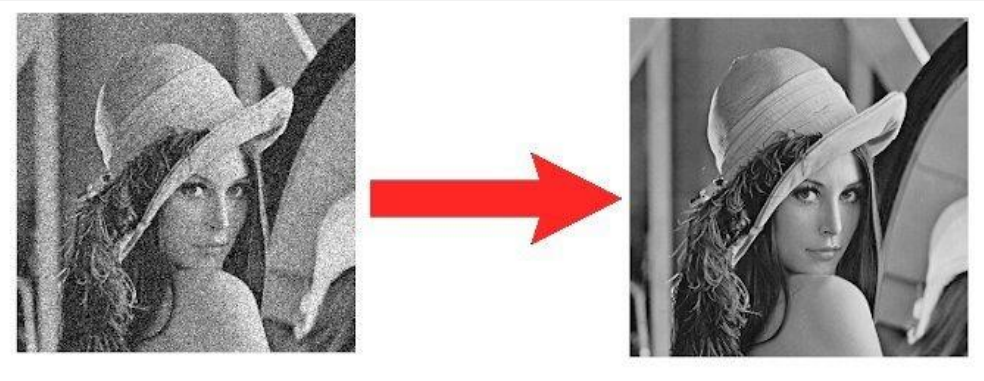
- *Rumore Gaussiano:* casuale e segue una distribuzione normale (gaussiana).
- *Rumore di Poisson*: presente in immagini con scarsa illuminazione, presenta variazioni casuali di intensità.

Example of Gaussian Noise

Denoising di Immagini Digitali

Ridurre il rumore senza compromettere alcune caratteristiche essenziali:

- Le **zone uniformi** devono restare omogenee
- I **bordi** devono essere preservati
- Le **texture** devono essere mantenute
- Non devono essere generati nuovi **artefatti**

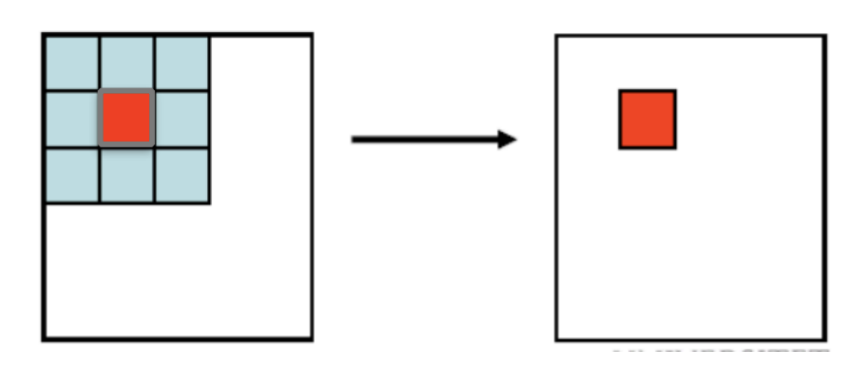


Approcci Principali al Denoising

1. **Filtraggio nel dominio spaziale**
2. **Filtraggio nel dominio delle trasformazioni**
3. **Metodi avanzati**
4. **Metodi basati sull'apprendimento automatico**

Filtraggio nel Dominio Spaziale

- **Definizione:** Operazioni sui pixel sfruttando correlazioni locali.



Filtri lineari:

- **Filtro medio:** Media dei pixel circostanti.
- **Filtro gaussiano:** Ponderazione centrata.

Filtri non lineari:

- **Filtro mediano:** Valore mediano dei pixel circostanti.
- **Filtro bilaterale:** Combina distanza spaziale e intensità per preservare i bordi.

Filtraggio nel Dominio delle Trasformazioni

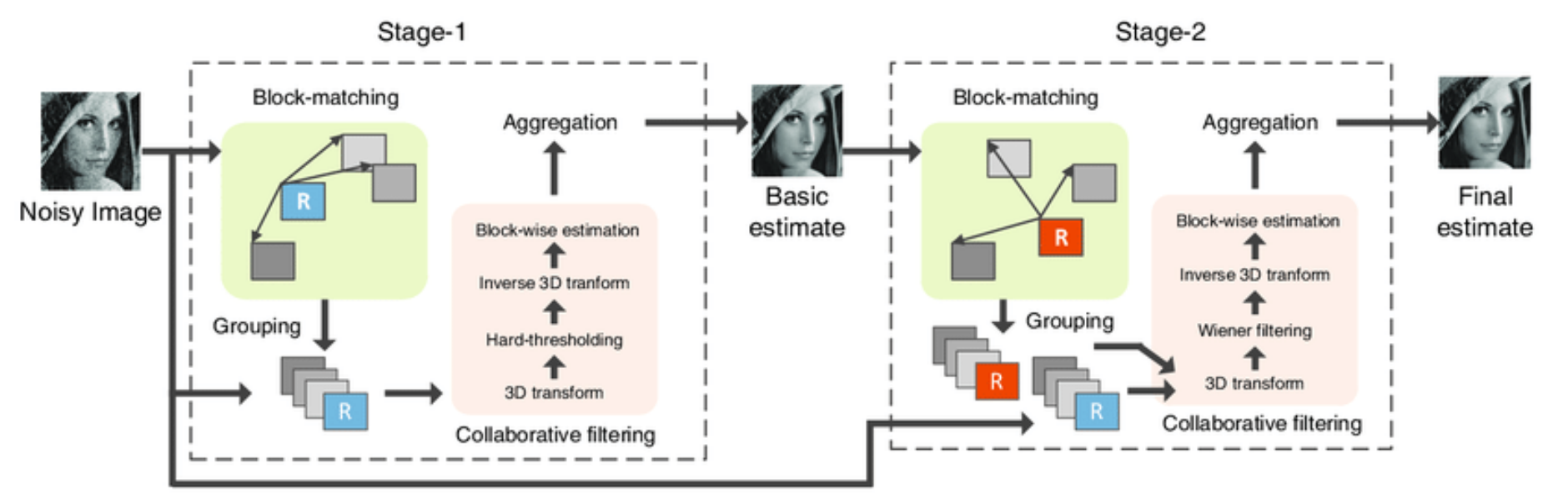
- **Processo:**
  1. Trasformare l'immagine in un dominio alternativo.
  2. Applicare filtri per ridurre il rumore.
  3. Tornare al dominio spaziale.

- **Tecniche principali:**
  - Trasformata di Fourier (FT): Rimozione delle alte frequenze.
  - Trasformata wavelet (WT): Multi-scala per separare dettagli e rumore.
  - Trasformazioni adattive: Si adattano alle caratteristiche specifiche dell'immagine e del rumore

Metodi Avanzati: BM3D

Adotta un approccio iterativo in due stadi costituiti dalle seguenti operazioni di base:

- **Block Matching:** Regioni simili nell'immagine vengono raggruppate.
- **Collaborative Filtering:** Applicazione di filtri nel dominio wavelet.
- **Aggregation:** Combina blocchi ripuliti.

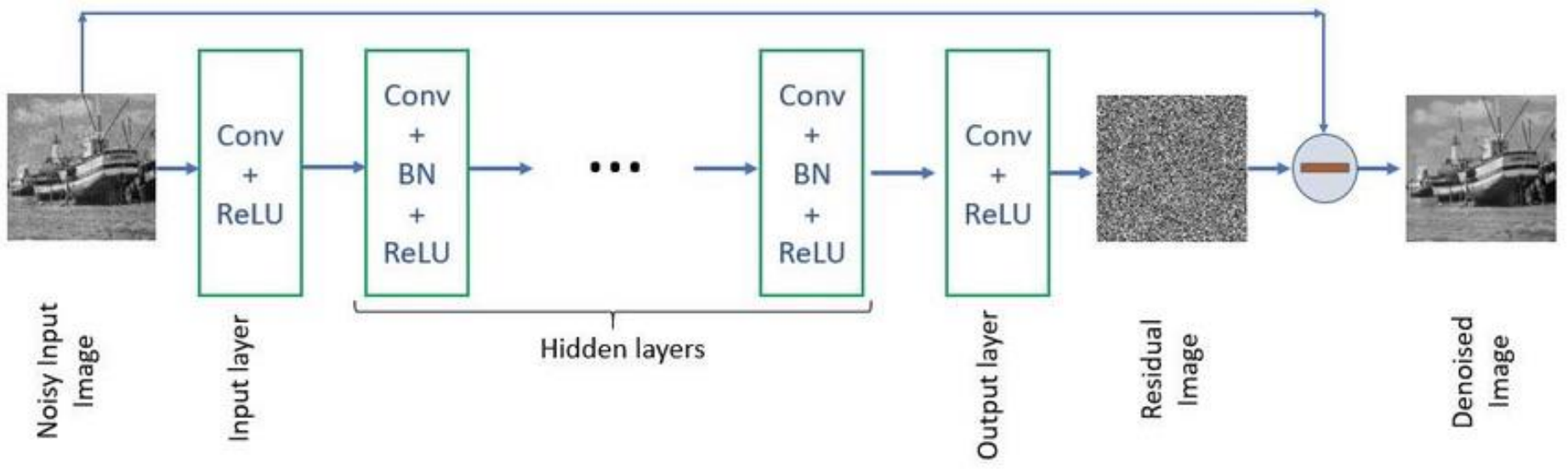


Alta qualità di riduzione del rumore con dettagli preservati.

Apprendimento Automatico

1. Convolutional Neural Networks (CNN):

Architettura DnCNN: predice il rumore, lo sottrae per ottenere l'immagine pulita.

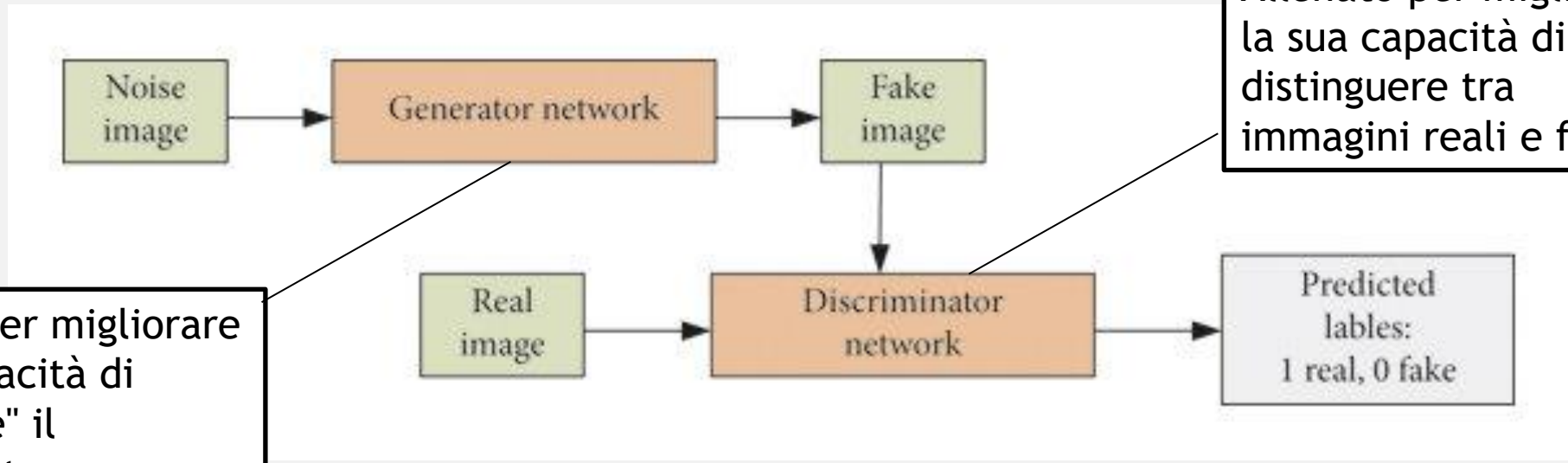


- Input: immagine rumorosa
- Output previsto: rumore sintetico aggiunto
- Loss Function: differenza tra il rumore predetto dalla rete e il rumore aggiunto

2. Generative Adversarial Networks (GAN):

Due reti neurali che lavorano in competizione tra loro:

- Generatore: produce immagini pulite che siano il più simili possibile a quelle reali.
- Discriminatore: valuta l'immagine e cerca di distinguere tra immagini reali e false.



Allenato per migliorare la sua capacità di distinguere tra immagini reali e false.

Allenato per migliorare la sua capacità di "ingannare" il Discriminatore.

Approcci Innovativi

- **Noise2Noise (N2N):**
  - Modello allenato su due immagini rumorose della stessa scena.
  - Immagini basate sullo stesso contenuto che sono "rumorose" in modi diversi.
  - **Vantaggio:** Non richiede immagini pulite.

- **Noise2Void (N2V):**
  - Lavora su una sola immagine rumorosa.
  - **Vantaggio:** Perfetto per contesti con dati limitati.

L’Algoritmo Noise2Void

Ingredienti:

1. **L'Architettura**: Quale modello di Rete Neurale è più adatto?

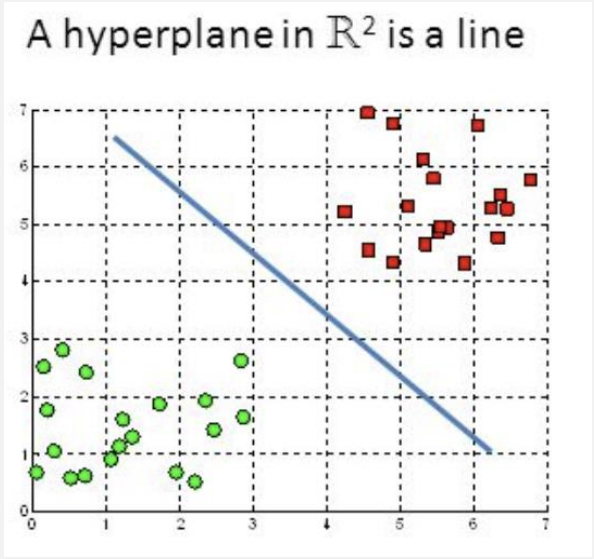
2. **L'Algoritmo vero e proprio**: Come fa l'Algoritmo a rimuovere il rumore?

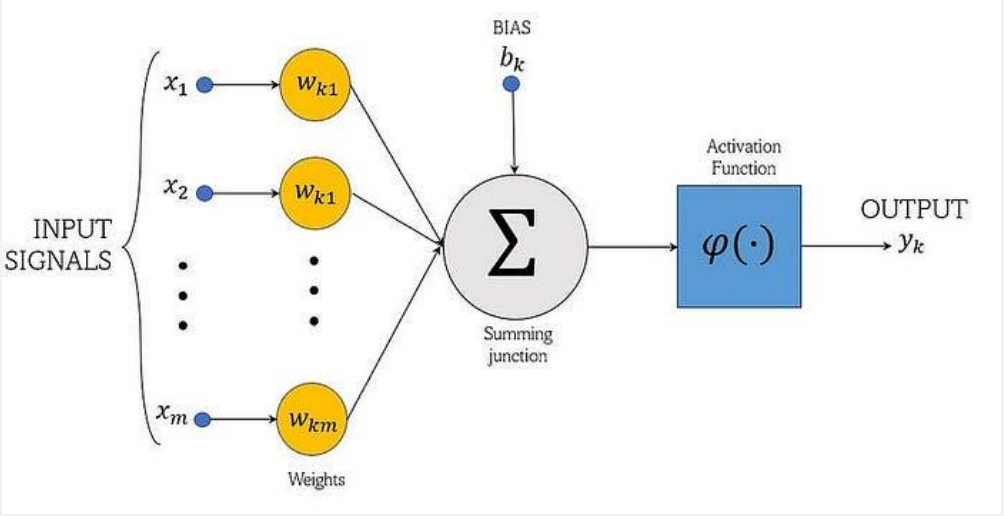
3. **La Loss Function**: A cosa fa riferimento, se non c'è una Ground Truth?

1. Un'Architettura per N2V...

A. Il Percettrone

- **Perché?** Per simulare un Neurone Umano
- **Per cosa?** Per Task di Classificazione Binaria Lineare
- **In che modo?**

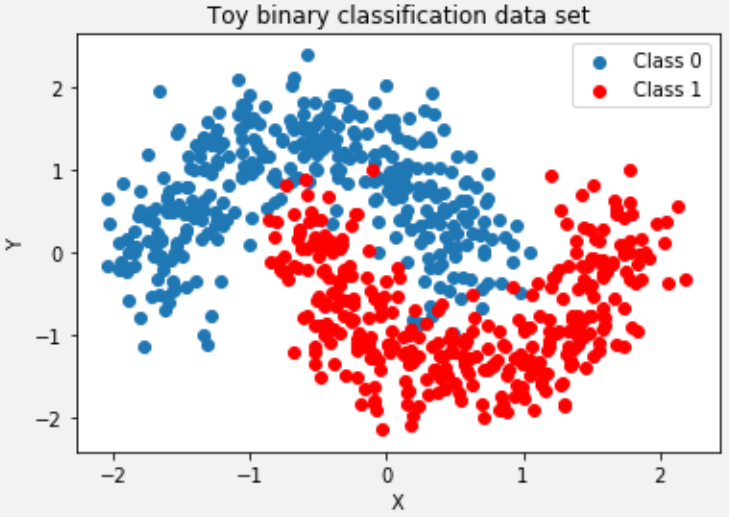


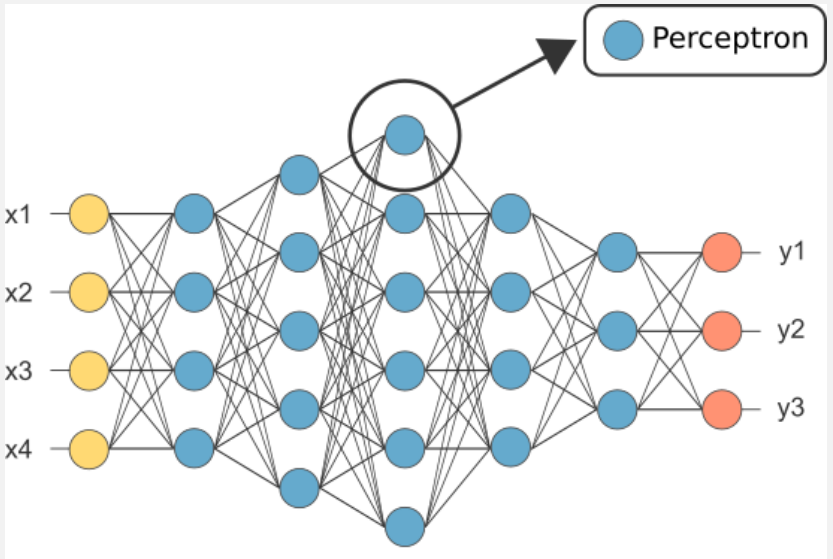


1. Un'Architettura per N2V...

B. Il Multi-Layer Perceptron (MLP)

- **Perché?** Per simulare il Sistema Nervoso Umano
- **Per cosa?** Per Task di Classificazione Non Lineare
- **In che modo?**





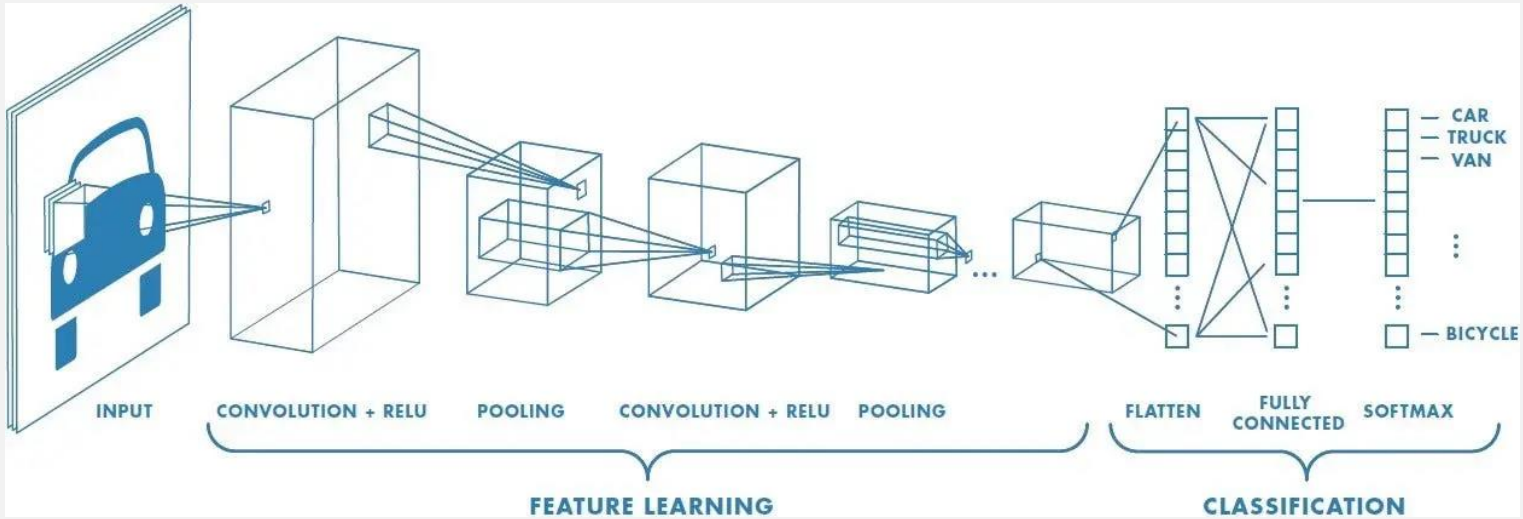
Nell'i-esimo Hidden Layer:

- Ogni Neurone (Percettrone) utilizza un Vettore di Pesi (N, 1).
- Complessivamente il Layer utilizza una Matrice di Pesi (N, M).

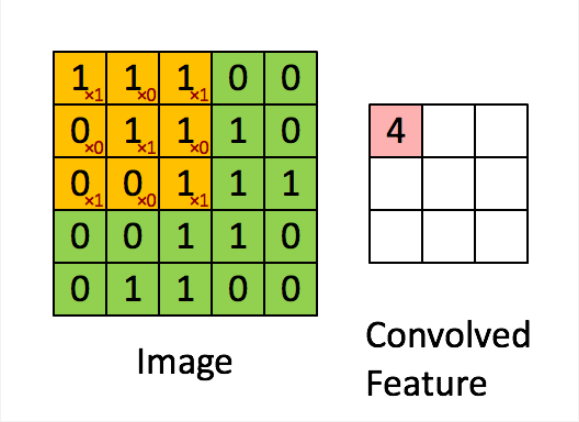
1. Un'Architettura per N2V...

C. Le Convolutional Neural Network (CNN)

- **Perché?** Per ottimizzare gli MLP con Input N-Dimensionali
- **Per cosa?** Per Image Recognition e Classification
- **In che modo?**



L'Operatore di Convoluzione



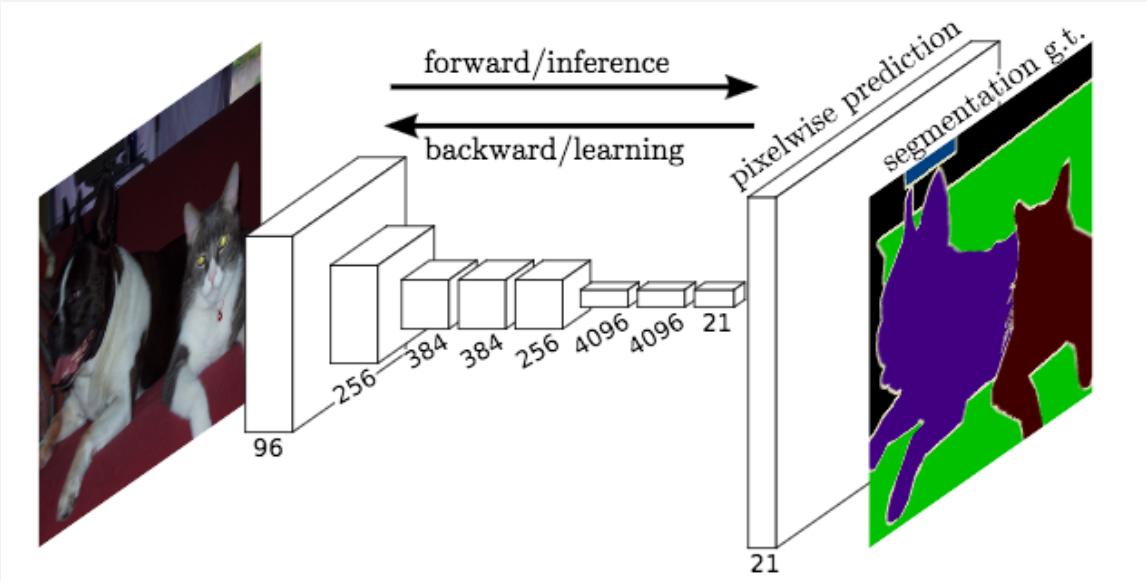
Rispetto agli MLP:

- Ogni Neurone utilizza un Filtro HxH.
- Le Feature Maps mantengono la Spazialità

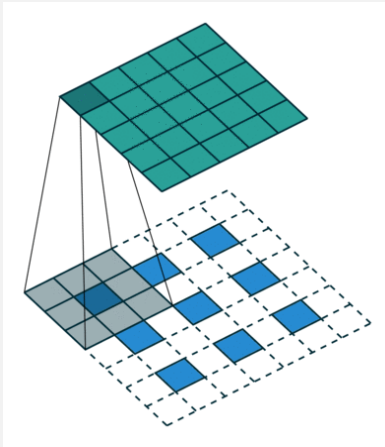
1. Un'Architettura per N2V...

D. Le Fully Convolutional Network (FCN)

- **Perché?** Per calcolare output N-Dimensionali.
- **Per cosa?** Per Task di Segmentazione (e non solo)
- **In che modo?**



L'Operatore di Deconvoluzione



Rispetto alle CNN:

- Non c'è più una parte Fully Connected
- E' necessario fare Upsampling

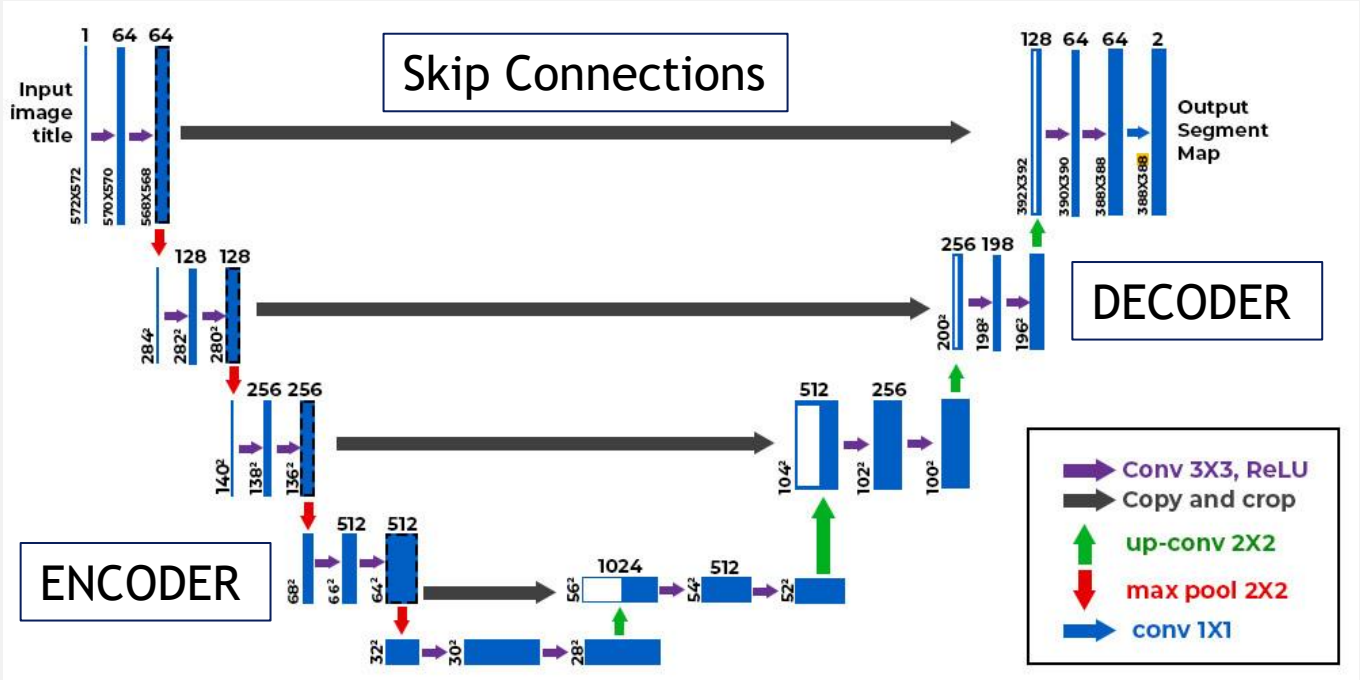
FCN-32, FCN-16, FCN-8

1. Un'Architettura per N2V...

E. U-Net

Le Trasformazioni di U-Net:
- Aumentano la Robustezza
- Consentono un minor numero di Input

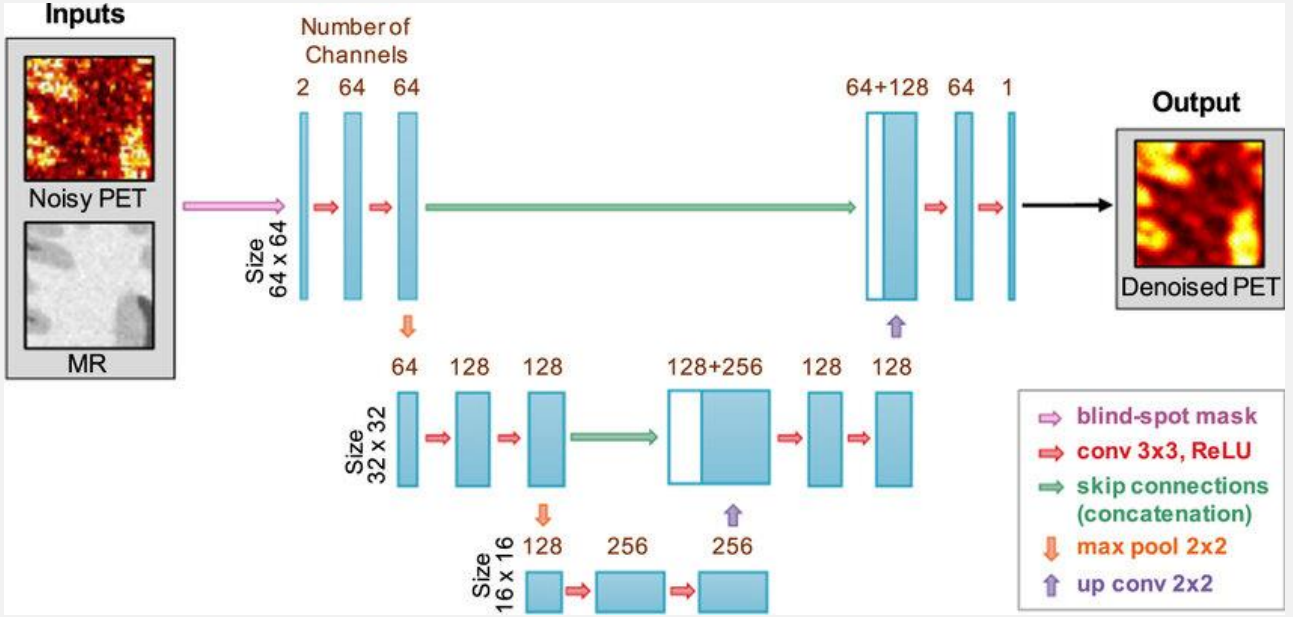
- **Perché?** Per perfezionare le FCN.
- **Per cosa?** Per Segmentazione in ambito Medico (e non solo)
- **In che modo?**



1. Un'Architettura per N2V...

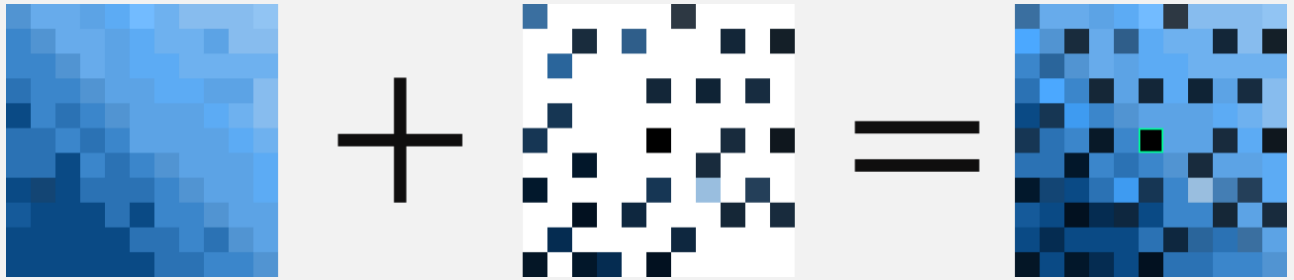
... è una U-Net

- ✓ Gestisce **Immagini** in Input
- ✓ Produce **Immagini** in Output
- ✓ Mantiene **Semantica** e **Spazialità**
- ✓ Richiede **piccole quantità** di Dati di Training



2. Un Algoritmo di Denoising

Formalizzando il Problema:



- L'immagine in Input è x = s + n
- Due assunzioni:
  - $P(s_i \mid s_j) \neq P(s_i)$ ⟵ ovvero i pixel del Segnale s sono Statisticamente Dipendenti
  - $P(n_i \mid n_j) = P(n_i)$ ⟵ ovvero i pixel del Rumore n sono Statisticamente Indipendenti

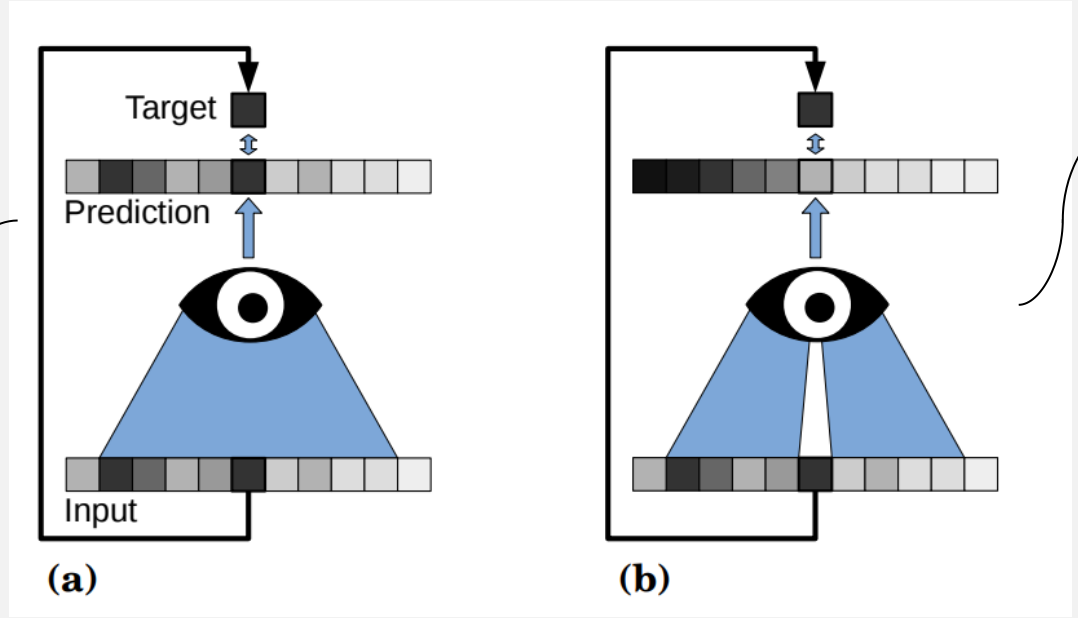
In pratica:

- I pixel che circondano un certo x_i sono legati tra loro solo dal segnale originale
- Osservando i pixel che circondano un certo x_i, con un quantità sufficiente di Addestramento possiamo far emergere s_i.

2. Un Algoritmo di Denoising

Una Blind-Spot Network:

Senza un Punto Cieco, ogni Predizione degenererà nel Pixel Centrale (l'"Identità")



Con un Punto Cieco, forziamo la Rete a predirre un risultato appropriato al contesto

L'area blu è il campo recettivo della nostra Rete!

- L'idea è buona, ma in termini di efficienza?

2. Un Algoritmo di Denoising

N2V Manipulate

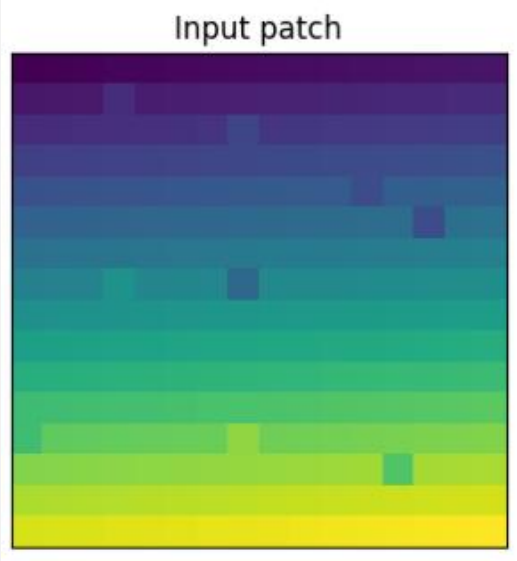
- Non possiamo permetterci di predire un pixel alla volta, proviamo ad aumentare i punti ciechi:

1. Patching: L'Immagine viene suddivisa in maniera randomica in Patch 64x64.

2. Sampling: In ogni Patch, selezioniamo N pixel in maniera (quasi) randomica.

3. Masking: Oscuriamo i Pixel Selezionati, sostituendoli con un altro pixel nelle loro vicinanze.

4. Predicting: Lasciamo che la Rete predica un nuovo valore per quei Pixel, dando in input le patch alla U-Net.

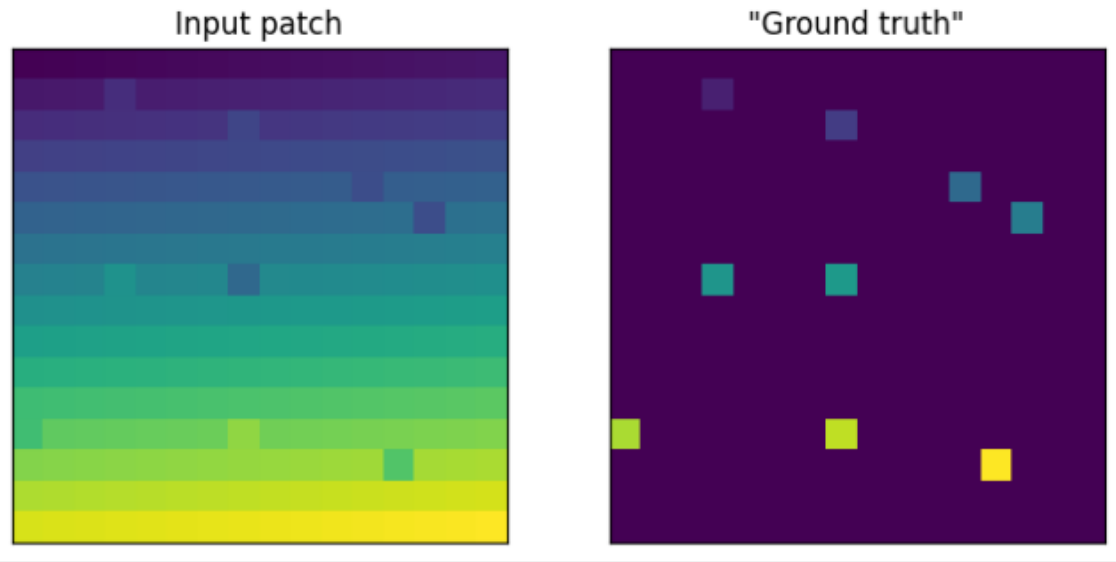


3. Una Loss Function senza Ground Truth

N2V Loss Function

- Tipicamente la Loss Function, necessaria per il Training, è calcolata a partire da un Ground Truth
- Abbiamo però detto che N2V non richiede una Ground Truth per funzionare
- Con che criterio possiamo aggiustare i Parametri della Rete?

Soluzione: Utilizziamo gli unici valori dell'immagine che non vengono dati in input alla Rete, ovvero i Pixel mascherati!



Mean Squared Error:

$$\text{Loss} = \frac{1}{N_{\text{masked}}} \sum_{i \in \text{masked}} (x'_i - x_i)^2$$

L'Algoritmo Noise2Void

Perché funziona?

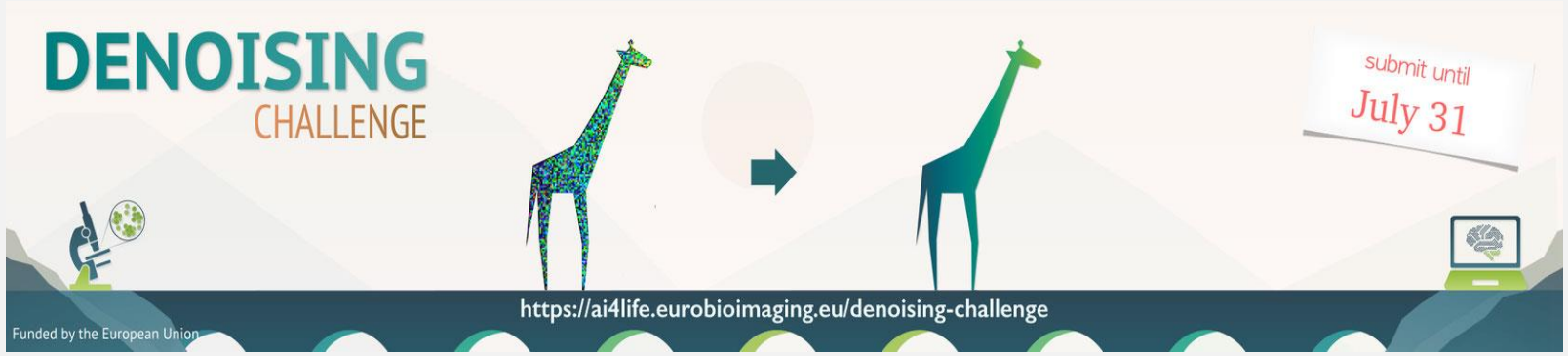
- E' sempre difficile giustificare il funzionamento di una Rete Neurale...
- Intuitivamente possiamo dire che:
  - Se vengono scelti per il Masking dei Pixel costituiti prevalentemente da **Rumore**, ci aspettiamo che prevalga la predizione fornita dalla Rete sulla base dei Pixel circostanti, andando dunque a correggere tale Pixel.
  - Se vengono scelti per il Masking dei Pixel rappresentanti il **Segnale Pulito**, ci aspettiamo che la Loss Function penalizzi la predizione della Rete e vengano mantenuti i Pixel originali.
- … il tutto su un numero sufficientemente grande di dati ed epoche di Training.

N2V - Grand Challenge

AI4Life Microscopy Denoising Challenge: volta a migliorare le immagini di microscopia biologica affette da diversi tipi di rumore (sia strutturato che non strutturato).

- Utilizzo di **dataset standardizzati**, contenenti immagini biologiche.
- Ogni dataset include un riferimento (**ground truth**) impiegati per la validazione dei risultati, non durante l'addestramento degli algoritmi.

N2V elimina la necessità di immagini pulite per l'addestramento, rendendolo adatto a contesti, come questo, dove ottenere dati di riferimento è complesso o costoso.



Struttura della challenge

Le performance degli algoritmi sono valutate attraverso diverse **metriche**:

- Structural Similarity Index (**SSIM**): analizza la somiglianza strutturale tra le immagini.
- Peak Signal-to-Noise Ratio (**PSNR**): valuta attraverso un grado di fedeltà.

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{MSE} \right)$$

I risultati dei partecipanti sono poi confrontati con approcci tradizionali e di nuova generazione, consentendo di identificare le soluzioni più efficaci.

Implementazione - Codice Python 1

- Libreria **CAREamist** che sfrutta framework popolari come **PyTorch**.

1. Create the Training Configuration:

```
{'algorithm_config': {'algorithm': 'n2v',
                      'loss': 'n2v',
                      'lr_scheduler': {'name': 'ReduceLROnPlateau',
                                       'parameters': {}},
                      'model': {'architecture': 'UNet',
                                'conv_dims': 2,
                                'depth': 2,
                                'final_activation': 'None',
                                'in_channels': 4,
                                'independent_channels': True,
                                'n2v2': False,
                                'num_channels_init': 32,
                                'num_classes': 4},
                      'optimizer': {'name': 'Adam',
                                    'parameters': {'lr': 0.0001}}},
...
```

Modello della U-Net

Numero di Layer di Profondità

Numero di Feature Maps al primo Livello

```
...
'data_config': {'axes': 'SCYX',
                'batch_size': 32,
                'data_type': 'array',
                'patch_size': [64, 64],
                'transforms': [{'flip_x': True,
                                'flip_y': True,
                                'name': 'XYFlip',
                                'p': 0.5},
                               {'name': 'XYRandomRotate90', 'p': 0.5},
                               {'masked_pixel_percentage': 0.2,
                                'name': 'N2VManipulate',
                                'roi_size': 11,
                                'strategy': 'uniform',
                                'struct_mask_axis': 'none',
                                'struct_mask_span': 5}]},
 'experiment_name': 'n2v_jump_cell_painting_chwise',
...
                'num_epochs': 100},
 'version': '0.1.0'}
```

Specchio

Rotazione

N2V

Trasformazioni Standard + N2VManipulate

Implementazione - Codice Python 2

2. Train the Model:

- L'addestramento del Modello avviene creando un oggetto **CAREamist**.
- Il metodo .**train** del Modello richiede che vengano fornite le immagini di Training e Validation.

```
# Before proceding, make sure your GPU is available to PyTorch or the training will be very slow

careamist = CAREamist(source=config, work_dir="notebooks/models/jump")

# train model
print(f"Training starting now...")
careamist.train(train_source=train_images, val_source=val_images)
print("Training ended!")
```

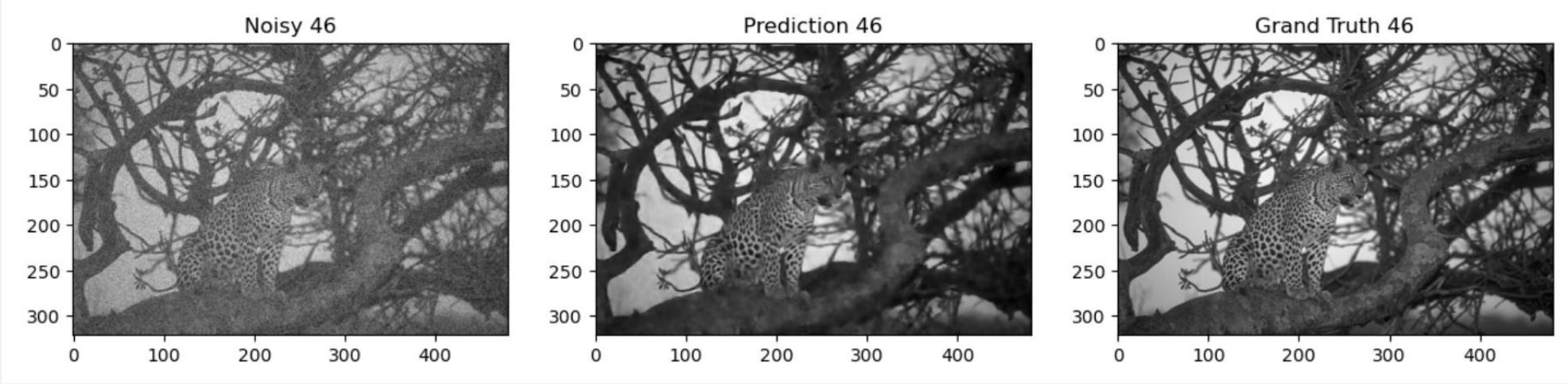
Implementazione - Codice Python 3

3. Generate Predictions:

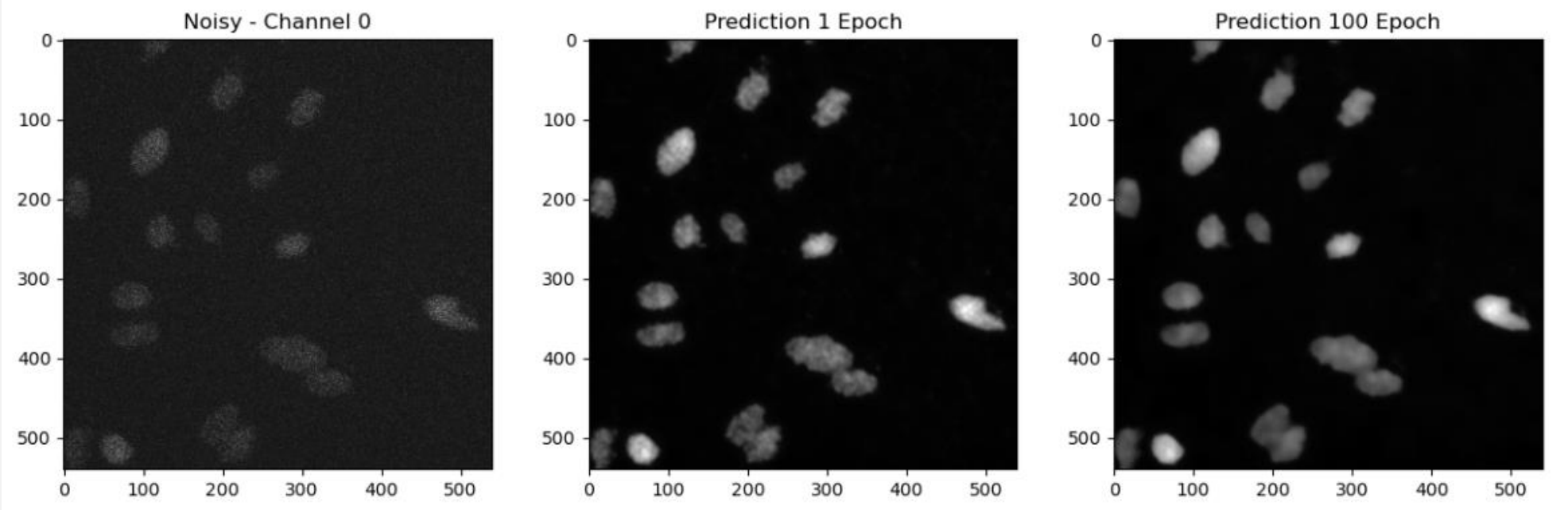
- Osserviamo il comportamento del Modello chiedendogli di **predire le Immagini Pulite** per alcuni campioni rumorosi.
- Tramite il metodo .**predict** del Modello utilizziamo l'ultimo Checkpoint disponibile per predirre le prime 10 immagini del Dataset.

```
output_path = "notebooks/predictions/jump/predictions.tiff"
predict_counter = 10 # The number of images we want to predict

predictions = []
for i in range(predict_counter):
    print(f"Predicting batch number {i}")
    pred_batch = careamist.predict(source=train_images[i], data_type='array', axes='CYX')
    predictions.append(pred_batch)

predictions = np.concatenate(predictions, axis=0).squeeze()
os.makedirs(os.path.dirname(output_path), exist_ok=True)
tifffile.imwrite(output_path, predictions)
print(f"TIFF file saved to {output_path}")
```

I Nostri Risultati



Si evidenzia la differenza tra la Prediction dell'algoritmo e l'Immagine Denoised



Si evidenzia la differenza tra diversi numeri di epoche nella Prediction

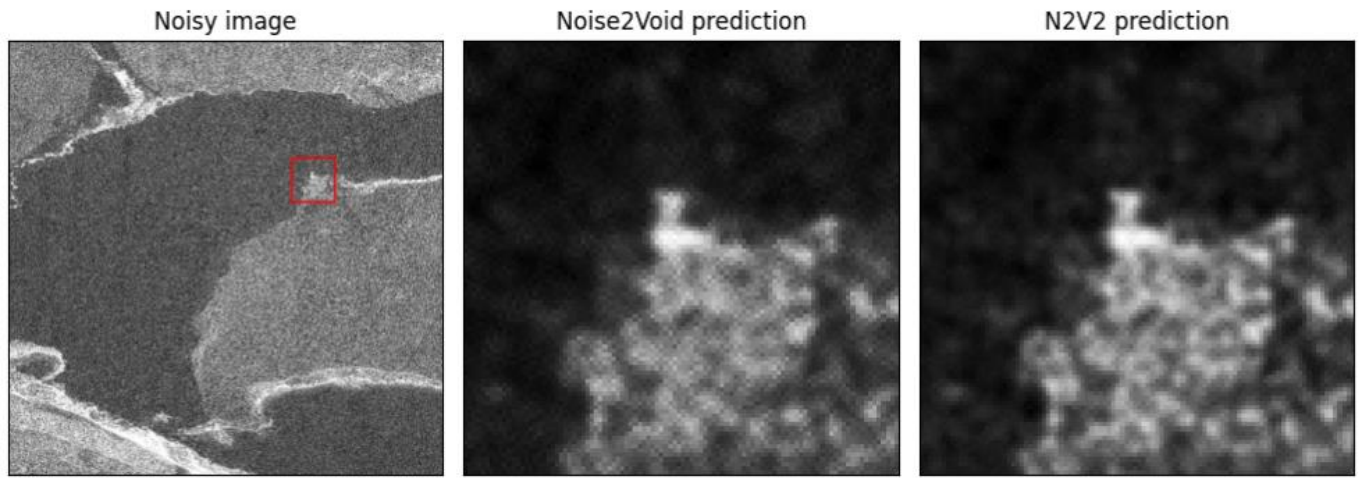
Blurry Images in N2V - N2V2

N2V soffre di **blurry images**: generazione di immagini con un aspetto sfocato.

- **Origine del problema**: N2V sfrutta il contesto locale di un pixel rumoroso per predirne il valore corretto, tuttavia ciò implica una perdita di dettagli fini e texture.

Questo problema viene risolto nella versione successiva, **Noise2Void2** (N2V2), attraverso:

- **Predizione probabilistica**: mantiene i dettagli e i bordi più nitidi.
- **Nuova funzione di perdita**: basata su distribuzioni, riduce il rischio di sovra-lisciamento.
- **Mascheramento ottimizzato**: aiuta a comprendere meglio il contesto locale.



Rumore nel Sistema Visivo Umano

Cos'è il Rumore nel Sistema Visivo Umano?

Si tratta di una fonte di disturbo che influenza la capacità di percepire e interpretare correttamente gli stimoli visivi.

- **Rumore Esterno**: Proviene dall'ambiente circostante: scarsa illuminazione, bagliori o riflessi.
- **Rumore Interno**: Origine da fattori neurali e biologici all'interno del sistema visivo tra cui:
  - Attività Spontanea dei Neuroni
  - Variazioni nella Trasmissione Sinaptica
  - Interazioni Casualizzate tra Reti Neurali

Somiglianze N2V e Sistema Visivo Umano

Emergono somiglianze in tre ambiti principali:

1. **Utilizzo del contesto**
2. **Apprendimento**
3. **Illusioni e Rumore Strutturato**

Utilizzo del contesto

- **Noise2Void**: Predice pixel rumorosi basandosi sulla correlazione spaziale.
- **Sistema Visivo Umano**:
  - Riempimento percettivo (es. punto cieco).
  - Usa il contesto locale per compensare mancanze (occlusioni, discontinuità).

Apprendimento

- **Noise2Void**: Modello auto supervisionato che apprende dalle immagini rumorose senza una ground truth.
- **Sistema Visivo Umano**:
  - Apprendimento percettivo dinamico basato sull'esperienza, senza necessitare di una "verità di riferimento".
  - Migliora nel tempo riconoscendo pattern e oggetti.

Illusioni e Rumore Strutturato

- **Noise2Void**:
  - Difficoltà a distinguere rumore strutturato dal segnale reale.
  - Può interpretare erroneamente rumore con pattern regolari come segnale.
- **Sistema Visivo Umano**:
  - Soggetto a illusioni ottiche.
  - Errori nella rappresentazione visiva in presenza di ambiguità.

Grazie per l'attenzione!

Bibliografia:

Shahsuvarov, Murad. (2023). Deep Learning Architectures.

Ronneberger, Olaf et altri (2016). U-Net: Convolutional Networks for Biomedical Image Segmentation.

Krull, Alexander et altri. (2019). Noise2Void – Learning Denoising from Single Noisy Images.

Careamics Github - Noise2Void

AI4life - mdc 2024. Grand-challenge

Karlinsky, Michaeli, Nishino (2023). Computer Vision – ECCV 2022 Workshops.

Fan, Zhang. (2019). Brief review of image denoising techniques.

Hurlbert. (2000). Visual perception: Learning to see through noise.

Zhang, Wang. (2020). An Analysis and Implementation of the BM3D Image Denoising Method.