

OUTLINE

Introduzione

- Workflow progetto

- Obiettivi

Tecnologie

- Backend

- Frontend

Il software

- Tool analisi

- Tool predizione

- Tool training

Basi teoriche

- Epilessia

- Reti neurali

- Reti neurali Convoluzionali

Training

- Dataset

WORKFLOW PROGETTO

- ▶ studio EEG e crisi epilettiche
- ▶ studio teorico CNN
- ▶ studio tool di sviluppo
- ▶ creazione di un dataset di allenamento bilanciato
- ▶ creazione e allenamento rete per le predizioni
- ▶ costruzione di architettura web backend e frontend

WORKFLOW PROGETTO

- ▶ studio EEG e crisi epilettiche
- ▶ studio teorico CNN
- ▶ studio tool di sviluppo
- ▶ creazione di un dataset di allenamento bilanciato
- ▶ creazione e allenamento rete per le predizioni
- ▶ costruzione di architettura web backend e frontend

WORKFLOW PROGETTO

- ▶ studio EEG e crisi epilettiche
- ▶ studio teorico CNN
- ▶ studio tool di sviluppo
- ▶ creazione di un dataset di allenamento bilanciato
- ▶ creazione e allenamento rete per le predizioni
- ▶ costruzione di architettura web backend e frontend

WORKFLOW PROGETTO

- ▶ studio EEG e crisi epilettiche
- ▶ studio teorico CNN
- ▶ studio tool di sviluppo
- ▶ creazione di un dataset di allenamento bilanciato
- ▶ creazione e allenamento rete per le predizioni
- ▶ costruzione di architettura web backend e frontend

WORKFLOW PROGETTO

- ▶ studio EEG e crisi epilettiche
- ▶ studio teorico CNN
- ▶ studio tool di sviluppo
- ▶ creazione di un dataset di allenamento bilanciato
- ▶ creazione e allenamento rete per le predizioni
- ▶ costruzione di architettura web backend e frontend

WORKFLOW PROGETTO

- ▶ studio EEG e crisi epilettiche
- ▶ studio teorico CNN
- ▶ studio tool di sviluppo
- ▶ creazione di un dataset di allenamento bilanciato
- ▶ creazione e allenamento rete per le predizioni
- ▶ costruzione di architettura web backend e frontend

OBIETTIVI



Il programma realizzato ha tre scopi principali:

- ▶ Offrire una visione grafica di file .edf contenenti tracciati elettroencefalografici
- ▶ Ridurre la fase di individuazione di crisi epilettiche all'interno di un tracciato EEG
- ▶ Offrire un ambiente di allenamento di reti neurali personalizzate per l'individuazione di crisi epilettiche

OBIETTIVI



Il programma realizzato ha tre scopi principali:

- ▶ Offrire una visione grafica di file .edf contenenti tracciati elettroencefalografici
- ▶ Ridurre la fase di individuazione di crisi epilettiche all'interno di un tracciato EEG
- ▶ Offrire un ambiente di allenamento di reti neurali personalizzate per l'individuazione di crisi epilettiche

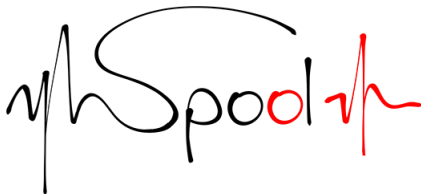
OBIETTIVI



Il programma realizzato ha tre scopi principali:

- ▶ Offrire una visione grafica di file .edf contenenti tracciati elettroencefalografici
- ▶ Ridurre la fase di individuazione di crisi epilettiche all'interno di un tracciato EEG
- ▶ Offrire un ambiente di allenamento di reti neurali personalizzate per l'individuazione di crisi epilettiche

OBIETTIVI



Il programma realizzato ha tre scopi principali:

- ▶ Offrire una visione grafica di file .edf contenenti tracciati elettroencefalografici
- ▶ Ridurre la fase di individuazione di crisi epilettiche all'interno di un tracciato EEG
- ▶ Offrire un ambiente di allenamento di reti neurali personalizzate per l'individuazione di crisi epilettiche

SVILUPPI FUTURI

- ▶ Deployment su server remoto
- ▶ Maggiore personalizzazione delle reti
- ▶ Ottimizzazione e diversificazione delle reti pre-allenate offerte

SVILUPPI FUTURI

- ▶ Deployment su server remoto
- ▶ Maggiore personalizzazione delle reti
- ▶ Ottimizzazione e diversificazione delle reti pre-allenate offerte

SVILUPPI FUTURI

- ▶ Deployment su server remoto
- ▶ Maggiore personalizzazione delle reti
- ▶ Ottimizzazione e diversificazione delle reti pre-allenate offerte

BACKEND

django, pytorch, pyedflib

FRONTEND

angular, ionic , highchart

TOOL ANALISI

blablabla

TOOL PREDIZIONE

blablabla

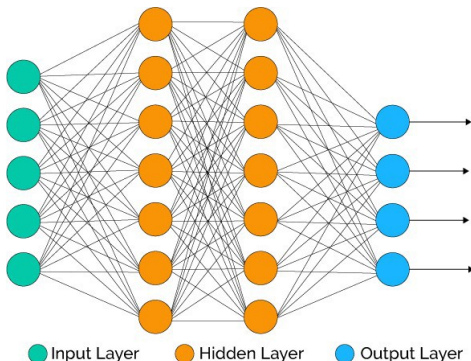
TOOL TRAINING

blablabla

EPILESSIA

bla bla

RETI NEURALI



- ▶ Modelli computazionali basati su neuroni
- ▶ Ispirate al funzionamento biologico del cervello
- ▶ Applicate nella risoluzione di problemi ingegneristici, informatici, di simulazione...

RETI NEURALI: TRAINING

Tipologie di allenamento:

- ▶ Supervised Learning
- ▶ Unsupervised Learning
- ▶ Semi-supervised Learning
- ▶ Apprendimento per rinforzo

Passi allenamento supervisionato:

- ▶ Inizializzazione
- ▶ Feed-forward
- ▶ Loss function → Calcolo errore
- ▶ Differentiation → Diminuzione loss
- ▶ Backpropagation e weights update

Ripetuti per ogni
epoca
Validation e Test

RETI NEURALI: TRAINING

Tipologie di allenamento:

- ▶ Supervised Learning
- ▶ Unsupervised Learning
- ▶ Semi-supervised Learning
- ▶ Apprendimento per rinforzo

Passi allenamento supervisionato:

- ▶ Inizializzazione
- ▶ Feed-forward
- ▶ Loss function → Calcolo errore
- ▶ Differentiation → Diminuzione loss
- ▶ Backpropagation e weights update

Ripetuti per ogni
epoca
Validation e Test

RETI NEURALI: TRAINING

Tipologie di allenamento:

- ▶ Supervised Learning
- ▶ Unsupervised Learning
- ▶ Semi-supervised Learning
- ▶ Apprendimento per rinforzo

Passi allenamento supervisionato:

- ▶ Inizializzazione
- ▶ Feed-forward
- ▶ Loss function → Calcolo errore
- ▶ Differentiation → Diminuzione loss
- ▶ Backpropagation e weights update

Ripetuti per ogni
epoca

Validation e Test

RETI NEURALI: TRAINING

Tipologie di allenamento:

- ▶ Supervised Learning
- ▶ Unsupervised Learning
- ▶ Semi-supervised Learning
- ▶ Apprendimento per rinforzo

Passi allenamento supervisionato:

- ▶ Inizializzazione
- ▶ Feed-forward
- ▶ Loss function → Calcolo errore
- ▶ Differentiation → Diminuzione loss
- ▶ Backpropagation e weights update

Ripetuti per ogni
epoca
Validation e Test

RETI NEURALI: TRAINING

Tipologie di allenamento:

- ▶ Supervised Learning
- ▶ Unsupervised Learning
- ▶ Semi-supervised Learning
- ▶ Apprendimento per rinforzo

Passi allenamento supervisionato:

- ▶ Inizializzazione
- ▶ Feed-forward
- ▶ Loss function → Calcolo errore
- ▶ Differentiation → Diminuzione loss
- ▶ Backpropagation e weights update

Ripetuti per ogni
epoca
Validation e Test

RETI NEURALI: TRAINING

Tipologie di allenamento:

- ▶ Supervised Learning
- ▶ Unsupervised Learning
- ▶ Semi-supervised Learning
- ▶ Apprendimento per rinforzo

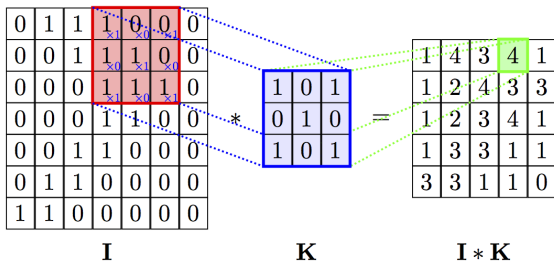
Passi allenamento supervisionato:

- ▶ Inizializzazione
- ▶ Feed-forward
- ▶ Loss function → Calcolo errore
- ▶ Differentiation → Diminuzione loss
- ▶ Backpropagation e weights update

Ripetuti per ogni
epoca
Validation e Test

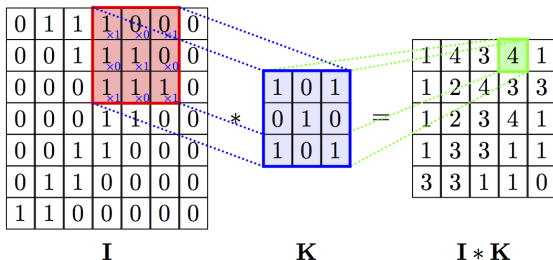
RETI NEURALI CONVOLUZIONALI

- ▶ Adatte ad input di grandi dimensioni
- ▶ Riconoscimento di *pattern* nell'architettura
- ▶ Applicazione di *kernel* per individuazione di features nei dati



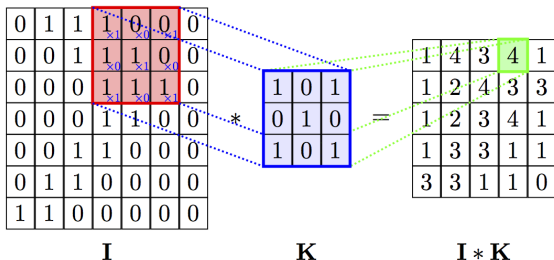
RETI NEURALI CONVOLUZIONALI

- ▶ Adatte ad input di grandi dimensioni
- ▶ Riconoscimento di *pattern* nell'architettura
- ▶ Applicazione di *kernel* per individuazione di features nei dati



RETI NEURALI CONVOLUZIONALI

- ▶ Adatte ad input di grandi dimensioni
- ▶ Riconoscimento di *pattern* nell'architettura
- ▶ Applicazione di *kernel* per individuazione di features nei dati



RETI NEURALI CONVOLUZIONALI

Parametri layer convoluzionale:

- ▶ Input (I)
- ▶ Profondità (K)
- ▶ Stride (S)
- ▶ Zero-Padding (P)

Dimensione del volume di output:

$$O = \frac{(I - K - 2P)}{S} + 1 \quad (1)$$

RETI NEURALI CONVOLUZIONALI

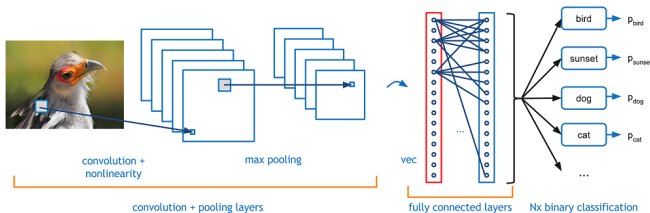
Parametri layer convoluzionale:

- ▶ Input (I)
- ▶ Profondità (K)
- ▶ Stride (S)
- ▶ Zero-Padding (P)

Dimensione del volume di output:

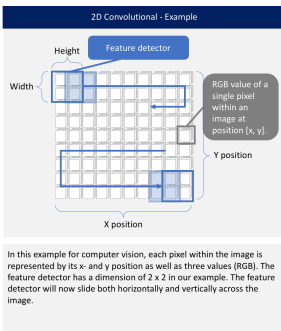
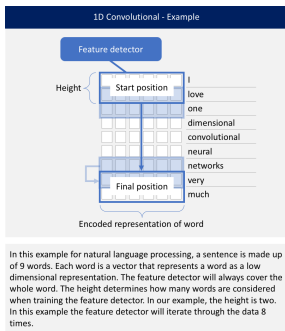
$$O = \frac{(I - K - 2P)}{S} + 1 \quad (1)$$

RETI NEURALI CONVOLUZIONALI



- ▶ Convolutional layers
- ▶ Pooling Layers (max, average...)
- ▶ Fully Connected Layers

CONVOLUZIONI 1D



Conv 1D (segnali, NLP...)

- ▶ Input bidimensionale (C, W)
- ▶ Il kernel scorre lungo la dimensione W

Conv 2D (immagini)

- ▶ Input tridimensionale (C, H, W)
- ▶ Il kernel scorre lungo le dimensioni H e W

MODELLO PROPOSTO

```
<bound method Module.state_dict of ConvNet(
  (layer1): Sequential(
    (0): Conv1d(23, 10, kernel_size=(10,), stride=(5,), padding=(9,))
    (1): ReLU()
    (2): MaxPool1d(kernel_size=15, stride=10, padding=0, dilation=1, ceil_mode=False)
  )
  (layer2): Sequential(
    (0): Conv1d(10, 5, kernel_size=(5,), stride=(4,), padding=(4,))
    (1): ReLU()
    (2): MaxPool1d(kernel_size=10, stride=5, padding=0, dilation=1, ceil_mode=False)
  )
  (drop_out): Dropout(p=0.5)
  (fc1): Linear(in_features=35, out_features=15, bias=True)
  (fc3): Linear(in_features=15, out_features=2, bias=True)
  (soft): Softmax())
)>
```

► Sequential:

- Convolution
- ReLU activation
- MaxPool

► Dropout

- Fully-connected
- Softmax

) >

► Dropout

MODELLO PROPOSTO

```
<bound method Module.state_dict of ConvNet(
  (layer1): Sequential(
    (0): Conv1d(23, 10, kernel_size=(10,), stride=(5,), padding=(9,))
    (1): ReLU()
    (2): MaxPool1d(kernel_size=15, stride=10, padding=0, dilation=1, ceil_mode=False)
  )
  (layer2): Sequential(
    (0): Conv1d(10, 5, kernel_size=(5,), stride=(4,), padding=(4,))
    (1): ReLU()
    (2): MaxPool1d(kernel_size=10, stride=5, padding=0, dilation=1, ceil_mode=False)
  )
  (drop_out): Dropout(p=0.5)
  (fc1): Linear(in_features=35, out_features=15, bias=True)
  (fc3): Linear(in_features=15, out_features=2, bias=True)
  (soft): Softmax()
)>
```

► Sequential:

► Convolution

► ReLU activation

► MaxPool

► Dropout

► Fully-connected

► Softmax

MODELLO PROPOSTO

```
<bound method Module.state_dict of ConvNet(
  (layer1): Sequential(
    (0): Conv1d(23, 10, kernel_size=(10,), stride=(5,), padding=(9,))
    (1): ReLU()
    (2): MaxPool1d(kernel_size=15, stride=10, padding=0, dilation=1, ceil_mode=False)
  )
  (layer2): Sequential(
    (0): Conv1d(10, 5, kernel_size=(5,), stride=(4,), padding=(4,))
    (1): ReLU()
    (2): MaxPool1d(kernel_size=10, stride=5, padding=0, dilation=1, ceil_mode=False)
  )
  (drop_out): Dropout(p=0.5)
  (fc1): Linear(in_features=35, out_features=15, bias=True)
  (fc3): Linear(in_features=15, out_features=2, bias=True)
  (soft): Softmax()
)>
```

► Sequential:

- Convolution
- ReLU activation
- MaxPool

► Dropout

► Fully-connected

► Softmax

MODELLO PROPOSTO

```
<bound method Module.state_dict of ConvNet(
  (layer1): Sequential(
    (0): Conv1d(23, 10, kernel_size=(10,), stride=(5,), padding=(9,))
    (1): ReLU()
    (2): MaxPool1d(kernel_size=15, stride=10, padding=0, dilation=1, ceil_mode=False)
  )
  (layer2): Sequential(
    (0): Conv1d(10, 5, kernel_size=(5,), stride=(4,), padding=(4,))
    (1): ReLU()
    (2): MaxPool1d(kernel_size=10, stride=5, padding=0, dilation=1, ceil_mode=False)
  )
  (drop_out): Dropout(p=0.5)
  (fc1): Linear(in_features=35, out_features=15, bias=True)
  (fc3): Linear(in_features=15, out_features=2, bias=True)
  (soft): Softmax()
)>
```

► Sequential:

- Convolution
- ReLU activation
- MaxPool

► Dropout

► Fully-connected

► Softmax

MODELLO PROPOSTO

```
<bound method Module.state_dict of ConvNet(
  (layer1): Sequential(
    (0): Conv1d(23, 10, kernel_size=(10,), stride=(5,), padding=(9,))
    (1): ReLU()
    (2): MaxPool1d(kernel_size=15, stride=10, padding=0, dilation=1, ceil_mode=False)
  )
  (layer2): Sequential(
    (0): Conv1d(10, 5, kernel_size=(5,), stride=(4,), padding=(4,))
    (1): ReLU()
    (2): MaxPool1d(kernel_size=10, stride=5, padding=0, dilation=1, ceil_mode=False)
  )
  (drop_out): Dropout(p=0.5)
  (fc1): Linear(in_features=35, out_features=15, bias=True)
  (fc3): Linear(in_features=15, out_features=2, bias=True)
  (soft): Softmax()
)>
```

► Sequential:

- Convolution
- ReLU activation
- MaxPool

► Dropout

- Fully-connected
- Softmax

γ

MODELLO PROPOSTO

```
<bound method Module.state_dict of ConvNet(
  (layer1): Sequential(
    (0): Conv1d(23, 10, kernel_size=(10,), stride=(5,), padding=(9,))
    (1): ReLU()
    (2): MaxPool1d(kernel_size=15, stride=10, padding=0, dilation=1, ceil_mode=False)
  )
  (layer2): Sequential(
    (0): Conv1d(10, 5, kernel_size=(5,), stride=(4,), padding=(4,))
    (1): ReLU()
    (2): MaxPool1d(kernel_size=10, stride=5, padding=0, dilation=1, ceil_mode=False)
  )
  (drop_out): Dropout(p=0.5)
  (fc1): Linear(in_features=35, out_features=15, bias=True)
  (fc3): Linear(in_features=15, out_features=2, bias=True)
  (soft): Softmax()
)>
```

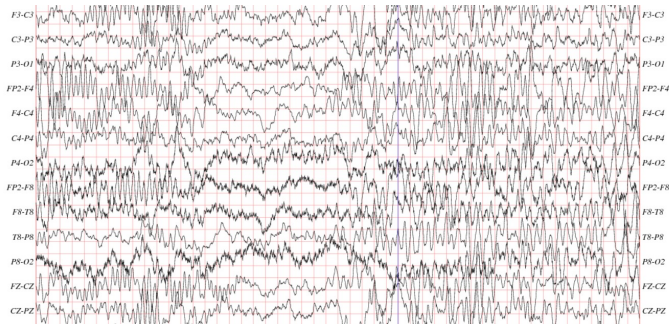
► Sequential:

- Convolution
- ReLU activation
- MaxPool

► Dropout

- Fully-connected
- Softmax

DATASET



- ▶ Children's Hospital Boston database (22 pazienti totali)
- ▶ Pazienti 1,2,3,5,7 e 8
- ▶ File con almeno un evento di crisi
- ▶ Frequenza di campionamento 256Hz
- ▶ 23 canali di campionamento

