

Classificazione di catene self-avoiding

Corso di Laboratory of
Computational Physics Mod. B



Indice

1. Introduzione e preprocessing dei dati
2. Modelli con matrice delle distanze
 - o Risultati
 - o Interpretazione: Heatmap
3. Modelli alternativi:
 - o Vettore con più canali
 - o Angoli diedri
4. Conclusioni e outlooks

Cosa sono le catene self-avoiding?

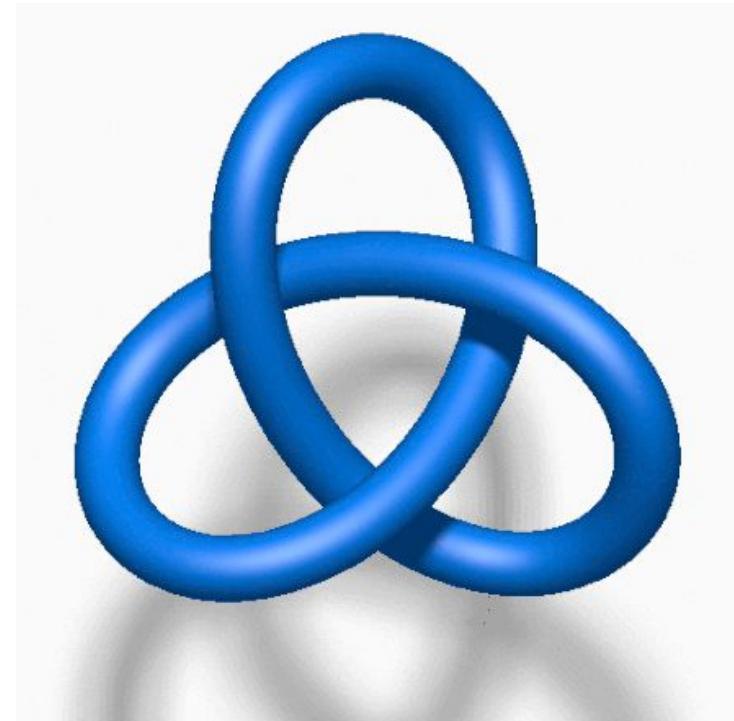
- Per catene self-avoiding si intendono catene chiuse generate attraverso algoritmi probabilistici che tengono conto della temperatura e della lunghezza di quest'ultime. Le catene sono simulate su di un reticolo cubico come random walk.

Qual è il nostro scopo?

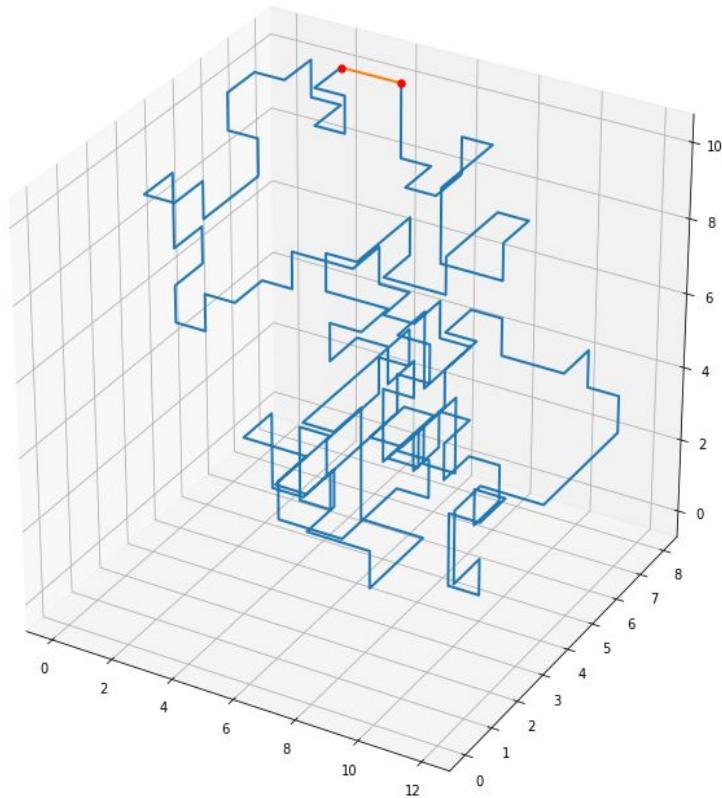
- Il nostro obiettivo è classificare le catene in nostro possesso in base al tipo di nodo presente in esse o eventualmente come “unknotted”. Per far questo abbiamo a disposizione due datasets iniziali: uno relativo alle catene da 200 passi e l'altro relativo a quelle da 400 passi.

Caratteristiche spaziali dei nodi

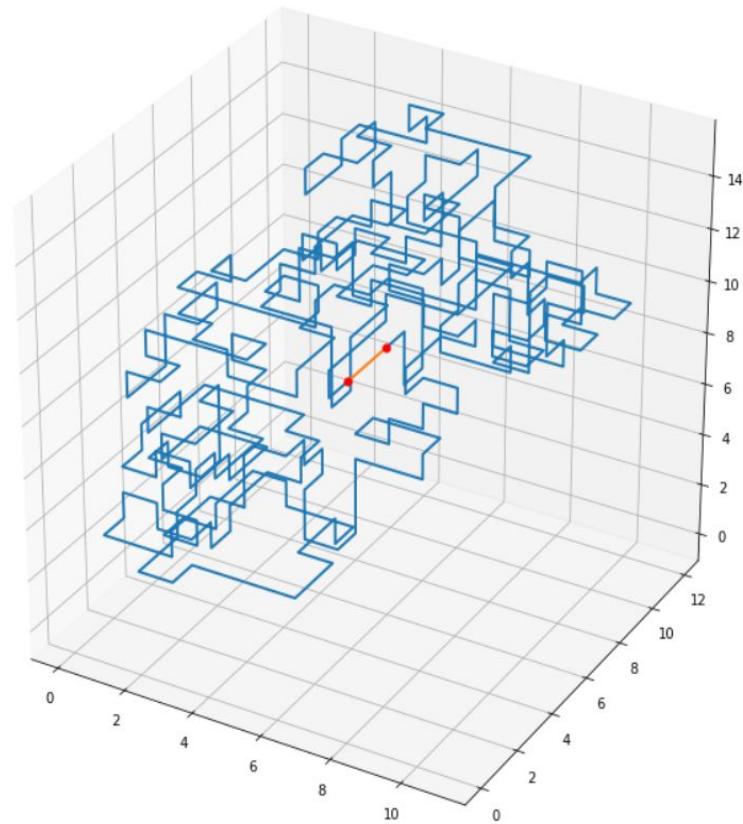
- I nodi sono invarianti per traslazioni, rotazioni, contrazioni/dilatazioni ma non per la chiralità: alcuni, infatti, sono diversi dalla propria immagine “specchiata”.
- I nodi nei 200 sono delocalizzati: tendono, cioè, ad occupare più porzioni della catena. Nei nodi dei 400, invece, c'è una maggiore localizzazione del complesso topologico.



Plot 3D di una catena di 200...



...e una di 400 passi



File nei dataset

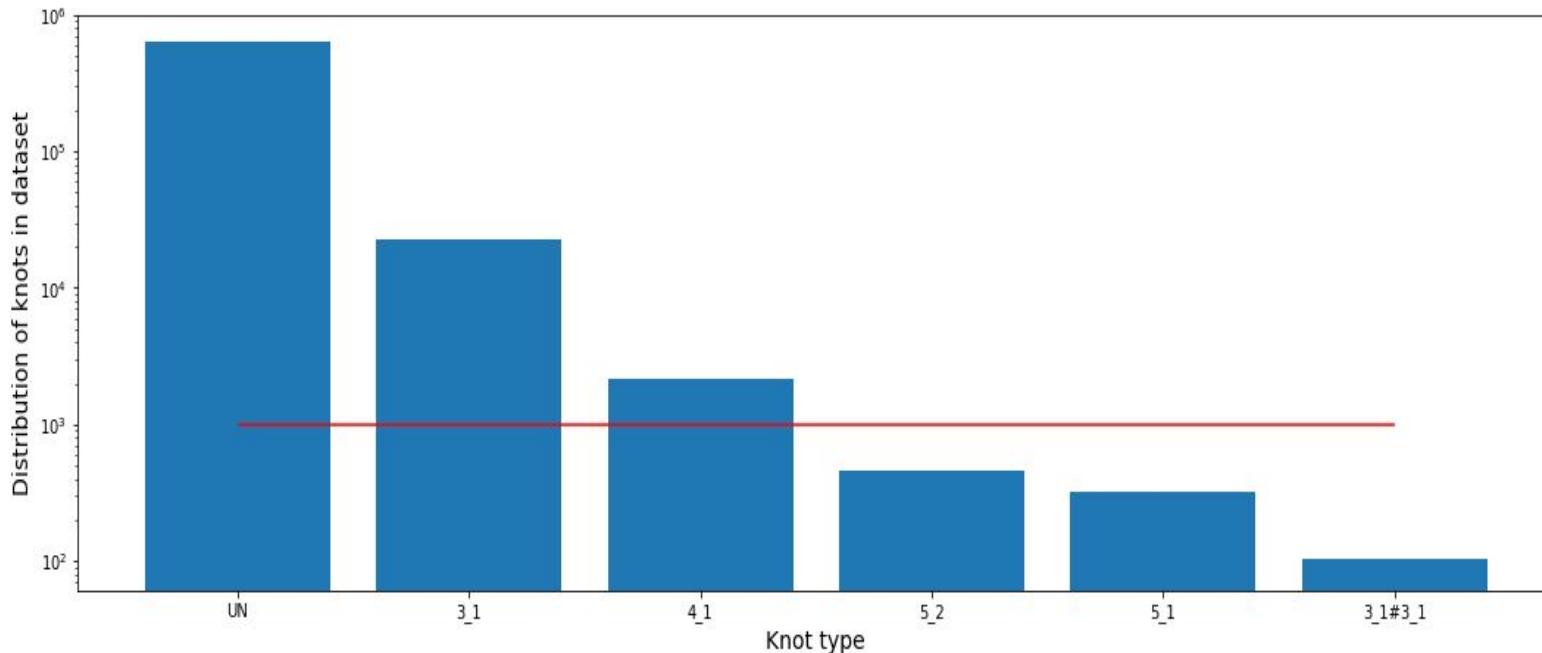
```
2.083930e-05
309
96
115512223434533310240110210204344535400555122113224355310540445332313115435544201320001350454045534
4533120501512110213133344420244540205115113344201131555432404505310211050042440443334311212151101515
4550122200213340202023421102432320151223154322421315434511210024204000445505332115045155154350132232
000150434043202235435500242202345553153321212332120154442045353135423515110043440200022044433544451
1.872129e-05
309
96
115512223434533310240110210204344535400555122113224355310540445332313115435544201320001350454045534
4533120501512110213133344420244540205115113344201131555432404505310211050042440443334311212151101515
4550122200213340202023421102432320151223154322421315434511210024204000445505332115045155154350132232
000150434043202235435500242202345553153321212332120154442045353135423515110043440200022044433544501
1.798558e-06
263
390
0202344231151333111020105512151544353240421132213334222244504334513510450135332043505344553151540535
510505422020211120005502010423312355334422354244421131001122101002223422342353355005355453345101233
451534212200024534353210201210222110451500544202321315401121335510000434055543150133550543500133350
44534542231013312110024012055312205044210124240001004343445101313454244042224023311354235444435051
```

1	UN	UN	-0.292929	2.723906	-0.028058	309	2.083930e-05	96
2	UNncr0	UN	-0.121212	2.588103	-0.024691	309	1.872129e-05	96
3	UNncr0	UN	-0.560045	2.176207	0.007856	263	1.798558e-06	390
4	UNncr0	UN	-0.503928	1.121212	0.001122	262	1.804203e-06	390
5	5_2	5_2	-3.796857	9.895623	0.037037	349	3.248603e-05	534
6	5_2	5_2	-3.933782	10.479237	0.072952	329	7.688062e-06	534
7	UNncr0	UN	-0.390572	3.171717	-0.008979	315	1.930488e-05	565
8	UN	UN	-0.007856	4.061728	0.001122	272	4.143635e-05	1589
9	UNncr0	UN	1.437710	3.399551	0.014590	352	7.868747e-06	1818
10	UNncr0	UN	-0.085297	1.301988	0.000000	305	2.811273e-05	2258
11	UNncr0	UN	0.333333	2.099888	-0.046016	290	1.158216e-05	2258
12	UN	UN	0.589226	3.597082	-0.023569	295	1.629124e-05	2258
13	3_1	3_1	-2.751964	5.618406	-0.017957	376	2.886520e-03	2453
14	3_1	3_1	-3.328844	4.383838	-0.006734	368	1.468261e-03	2453
15	UNncr0	UN	1.112233	3.258137	0.019080	346	3.817632e-05	2453
16	UN	UN	0.721661	4.762065	-0.050505	346	3.269201e-05	2453
17	3_1	3_1	2.887767	5.540965	-0.001122	342	1.194873e-03	2739
18	UNncr0	UN	0.866442	1.768799	0.000000	299	1.484924e-04	2949
19	UNncr0	UN	0.259259	3.276094	0.003367	302	2.465060e-04	2949
20	UNncr0	UN	0.010101	0.737374	-0.003367	298	8.971430e-05	2949
21	UN	UN	0.012346	1.848485	0.003367	301	2.978620e-04	2949

Dati in nostro possesso:

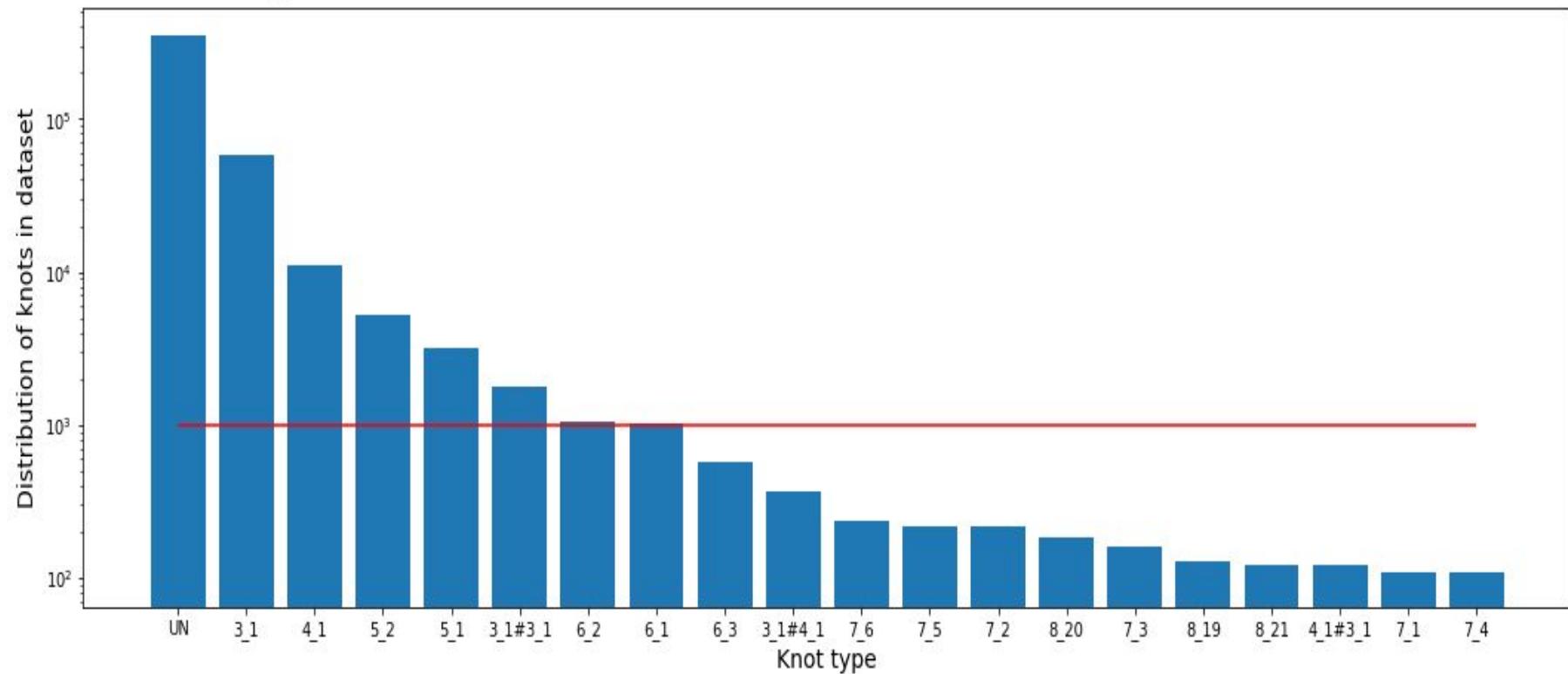
- Per i 200, 660000 sequenze e 22 tipi di nodi diversi
- Per i 400, 430000 sequenze e 188 tipi di nodi diversi.

Distribuzione dei nodi per i 200...



- Da notare un forte unbalance per le catene “unknotted” rispetto a tutte le altre categorie.

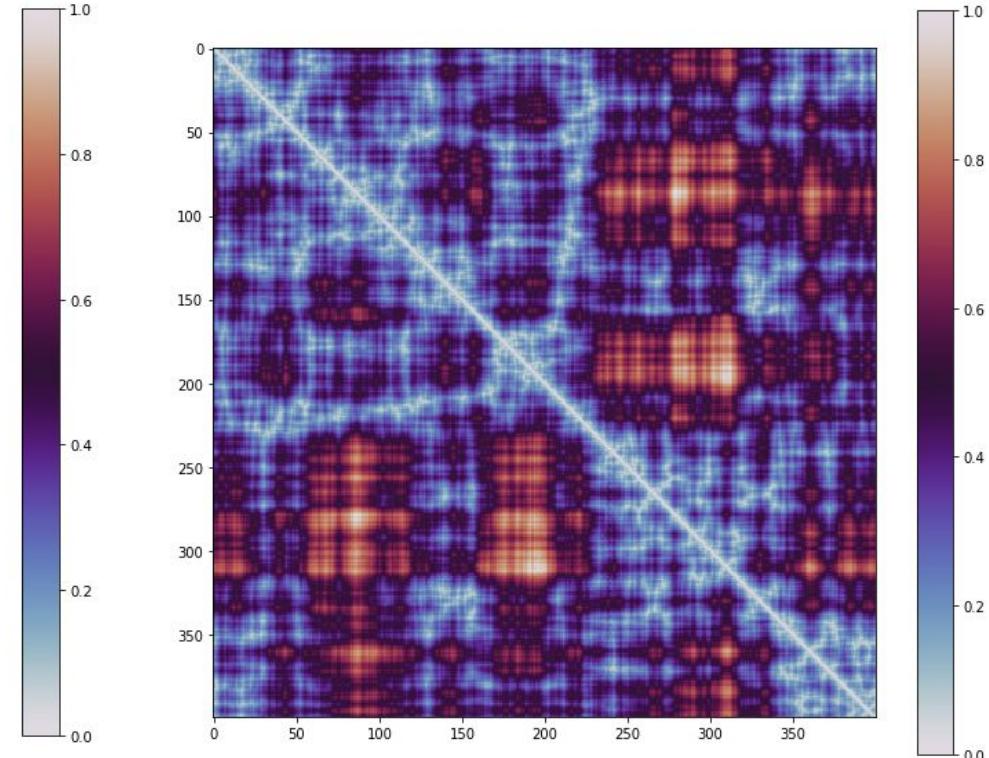
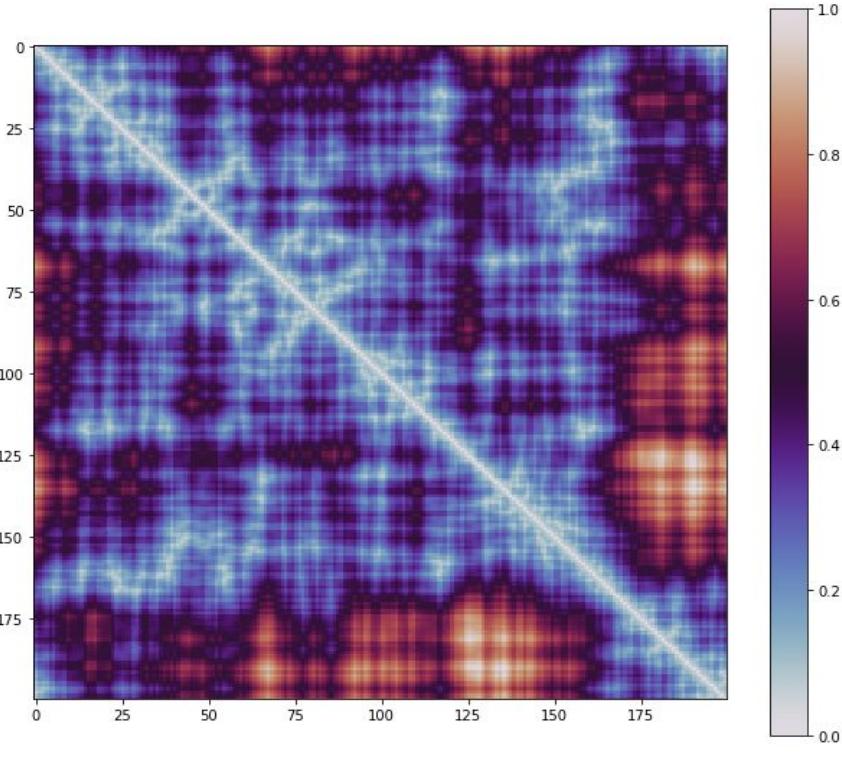
...e per i 400



Preprocessing

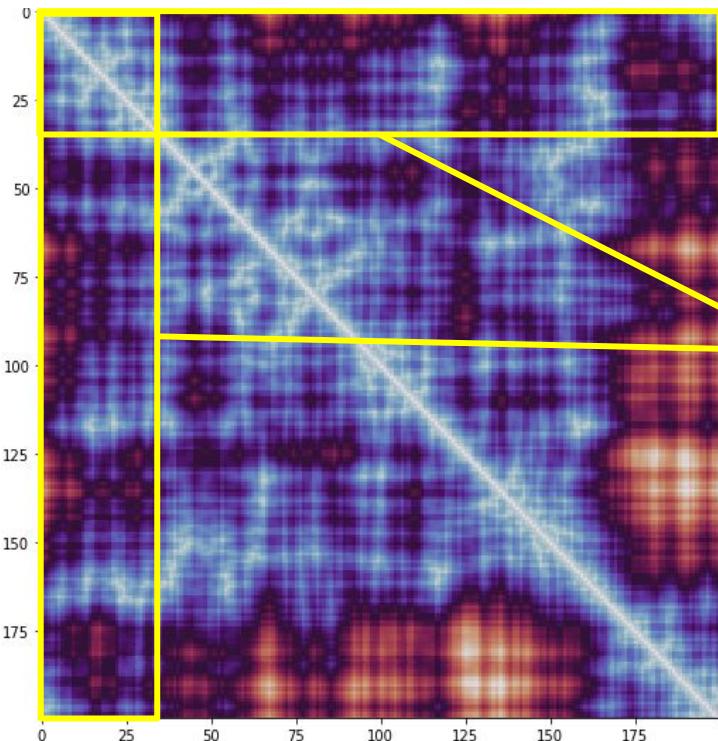
- Il primo passo è stato quello di parametrizzare l'input che poi, come vedremo, andrà “in pasto” alla Convolutional Neural Network (CNN)
- L'esigenza è stata quella di trovare un modo per esprimere la sequenza che identifica la catena in modo che essa risultasse invariante per rotazioni e traslazioni (comprendendo, quindi, i possibili augmenting dell'input iniziale)
- Per far ciò si è optato per una “matrice delle distanze” (MD): cioè una matrice di grandezza NxN in cui ogni entrata ha per riga il punto da cui si prendono le distanze per i successivi N-1 passi e per colonna tutti gli N passi della catena.

Esempi di MD per le catene da 200 e 400 passi

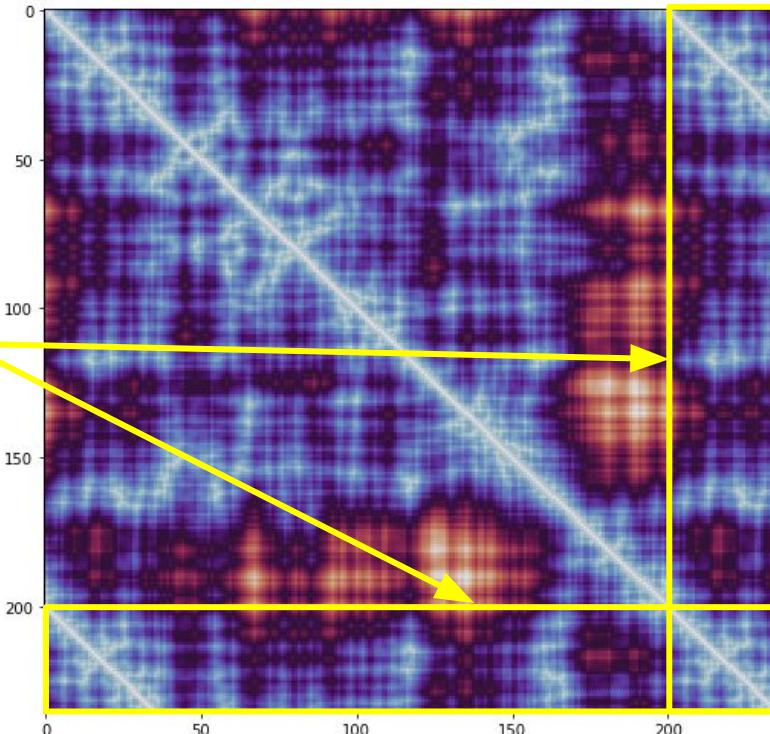


Il padding periodico

Pre...



...post padding periodico



Perchè la CNN?

- Vista la sua efficacia nella classificazione dei pattern, si è scelto di usare una CNN e di “allenarla” su di una parte delle sequenze dei datasets dopo la necessaria parametrizzazione, come detto, attraverso le MD.
- L'altra parte dei dataset è stata usata come *Validation set* per testare la “bontà” delle diverse architetture usate nel corso delle varie prove.

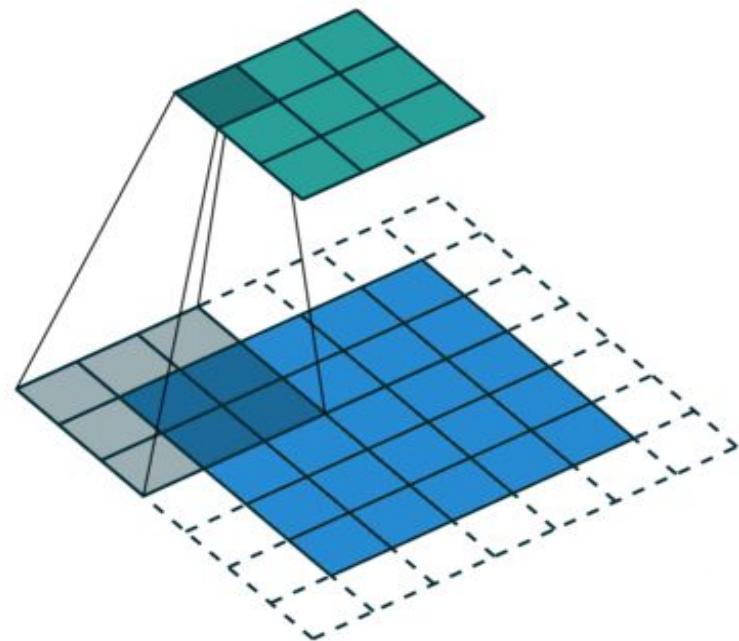
Cosa è una CNN e come funziona?

- La CNN è alla base delle principali applicazioni del deep learning e ha come scopo il riconoscimento di input, tipicamente delle immagini, attraverso analisi non-lineari dei pattern presenti.
- Tale riconoscimento avviene sulla scorta di input precedentemente usati nella fase di training: in questa fase, infatti, essa cerca di fissare correttamente i parametri interni all'architettura data. Quindi, una volta fissati questi parametri, la CNN li usa per catalogare al meglio i nuovi input provenienti dal validation set.
- Per il raggiungimento di questo scopo è anche importante definire in maniera opportuna certi **iperparametri** che concorrono al miglioramento dell'accuracy

Quali sono gli iperparametri?

Visto che lo spazio degli iperparametri è particolarmente esteso citeremo solo quelli su cui ci siamo concentrati maggiormente nel definire l'architettura:

- **Padding**: è spesso necessario quando il filtro va oltre l'activation map e tipicamente è l'aggiunta di zeri ai bordi dell'input
- **Stride**: indica di quanti pixel il filtro si sposta nella convoluzione con l'input
- **Kernel size**: grandezza dei filtri
- **Filter numbers**: numero di filtri usati in ogni convolutional layer



- **Pooling**: utile per ridurre le features da analizzare: in sostanza, una volta che vengono definiti **strides** e **kernel size**, il filtro prende in considerazione solo il pixel con il valore più alto (se si usa la forma Max) oppure fa una media dei vari pixel su cui si trova a passare (se si usa la forma Average)

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

Input

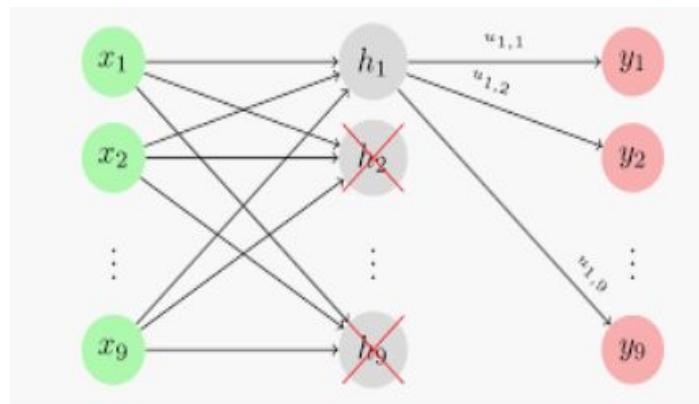
4	0
6	1

Max Pooling

0.25	-2.25
3	-3

Mean Pooling

- **Dropout:** percentuale dell'output dei "neuroni" che viene scartata durante il training per evitare overfitting



- **Activation function:** tipicamente non lineare, l'activation function è utile per imporre che tipo di output avrà ciascun neurone (softmax, ad esempio, normalizza output)

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}. \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = \max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^z)$	Multi-layer Neural Networks	

Catene da 200, classificazione binaria

L'architettura del modello

Obiettivo:

Distinguere i nodi (senza discriminare il tipo) dai non-nodi ('UN')

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 54, 54, 32)	8224
conv2d_1 (Conv2D)	(None, 27, 27, 48)	98352
conv2d_2 (Conv2D)	(None, 27, 27, 48)	36912
max_pooling2d (MaxPooling2D)	(None, 13, 13, 48)	0
conv2d_3 (Conv2D)	(None, 13, 13, 32)	24608
max_pooling2d_1 (MaxPooling2D)	(None, 6, 6, 32)	0
dropout (Dropout)	(None, 6, 6, 32)	0
flatten (Flatten)	(None, 1152)	0
dense (Dense)	(None, 186)	214458
dropout_1 (Dropout)	(None, 186)	0
dense_1 (Dense)	(None, 1)	187

Total params: 382,741

Trainable params: 382,741

Non-trainable params: 0

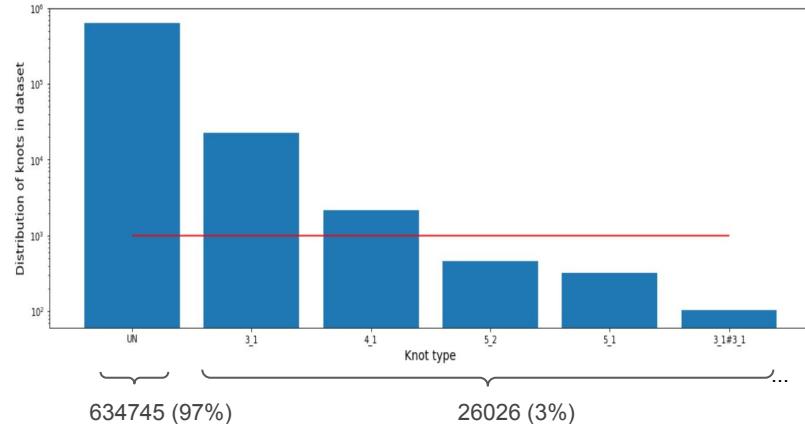
```
model = Sequential()
model.add( Conv2D(filters = 32,
                  kernel_size = 16,
                  activation = 'relu',
                  input_shape = (216, 216, 1),
                  padding = 'same',
                  strides = (4,4)
                 ) )
model.add( Conv2D(filters = 48,
                  kernel_size = 8,
                  activation = 'relu',
                  padding = 'same',
                  strides = (2,2)
                 ) )
model.add( Conv2D(filters = 48,
                  kernel_size = 4,
                  activation = 'relu',
                  padding = 'same',
                  ) )
model.add( MaxPooling2D(pool_size = (2,2),
                      strides = (2,2),
                      padding='valid'
                     ) )
model.add( Conv2D(filters = 32,
                  kernel_size = 4,
                  activation = 'relu',
                  padding = 'same',
                  ) )
model.add( MaxPooling2D(pool_size = (2,2),
                      padding='valid'
                     ) )
model.add( Dropout(.35) )
model.add( Flatten() )
model.add( Dense(186,
                 activation = 'relu',
                 ) )
model.add( Dropout(.75) )
model.add(Dense(n_class, activation='sigmoid'))

# Compiling the model
model.compile(loss = binary_crossentropy,
              optimizer = optimizers.Nadam(learning_rate = 0.0001),
              metrics = ['accuracy'])
```

Catene da 200, classificazione binaria

Il training

- Problema:
Con un dataset altamente sbilanciato è favorita la classe più rappresentata: Random Undersampling
- Dataset: circa 60.000 sequenze
 - 75% Training set
 - 15% Validation set
- Leggera predominanza (58%) di non nodi rispetto ai nodi (42%) che ottimizza l'accuracy del modello
- Il training è stato eseguito tramite due chiamate di `model.fit()`: OOM eseguendo il fit sull'intero dataset



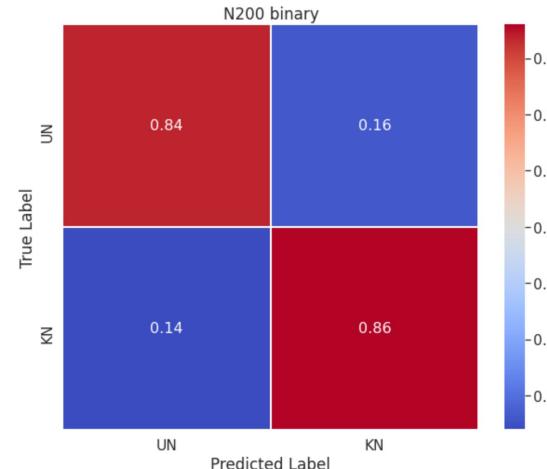
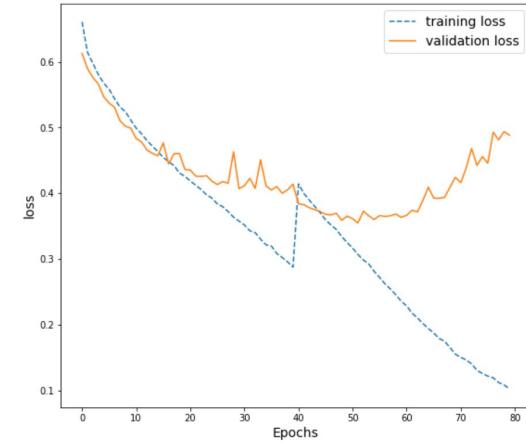
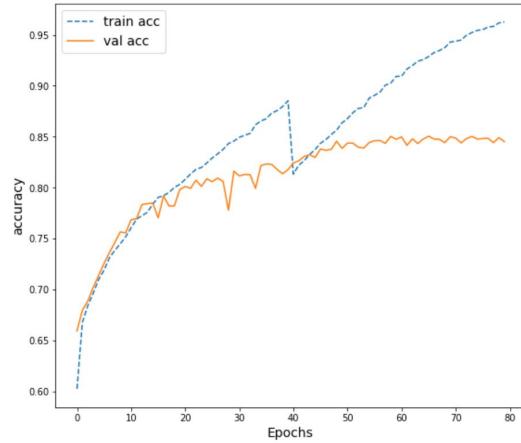
```
# Training
epochs = 40
step_update = 64
validation_split = 0.15

fit = model.fit(X, Y,
                 batch_size = step_update,
                 epochs = epochs,
                 validation_split = validation_split,
                 verbose = 1,
                 shuffle = True)
```

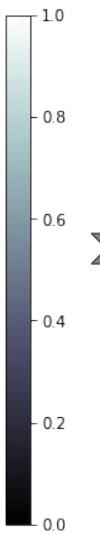
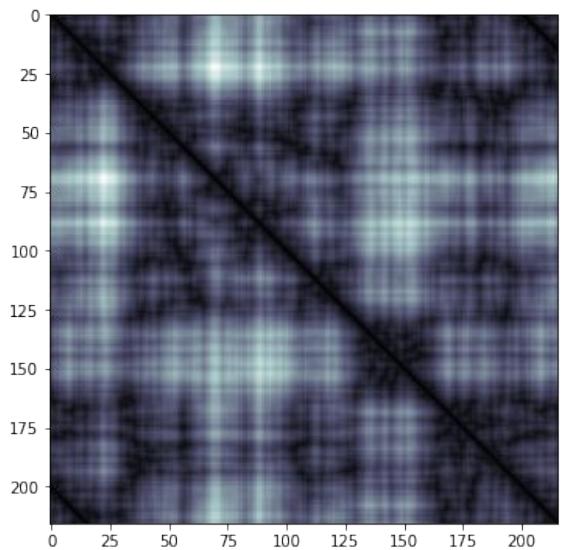
Catene da 200, classificazione binaria

Risultati del training

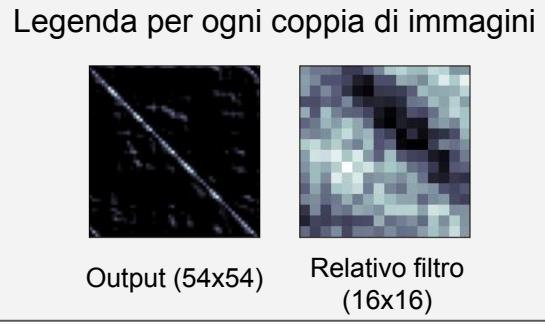
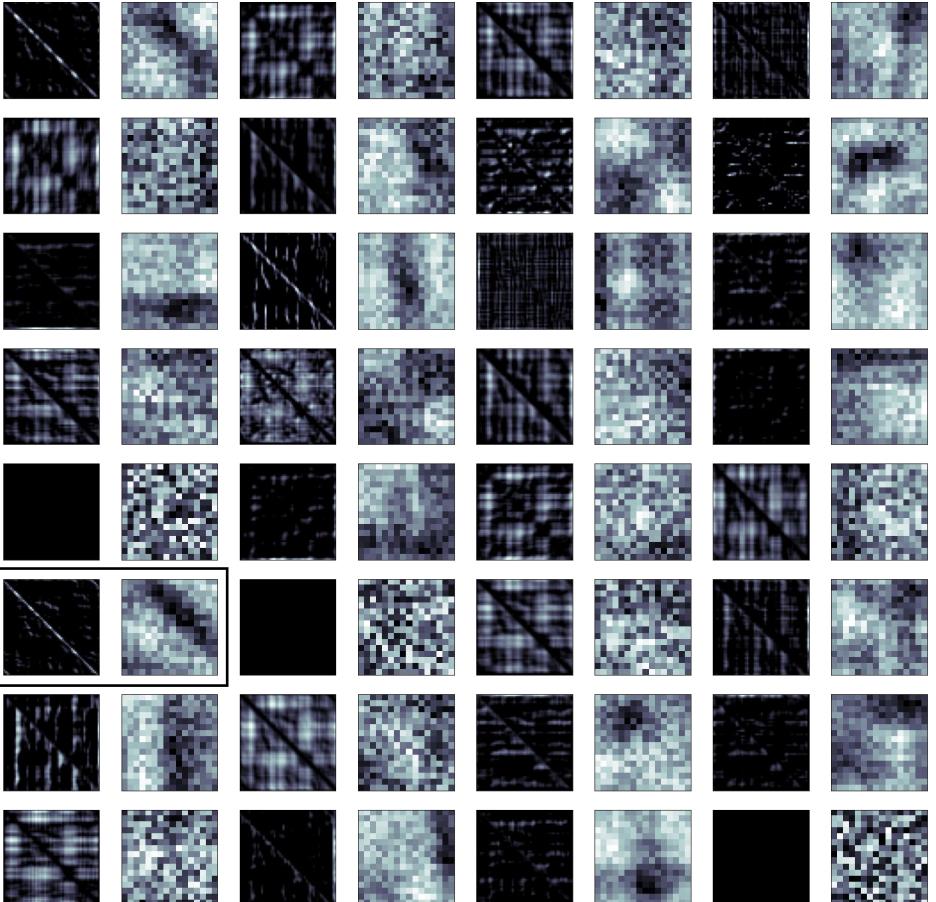
- Accuracy ≈ 0.85
- Nel grafico è mostrato un inizio di overfitting a puro scopo illustrativo: il training è bloccato prima
- Confusion matrix: il modello è in grado di riconoscere l'84% di non nodi e l'86% di nodi



Feature maps del primo Conv2D e visualizzazione dei rispettivi filtri

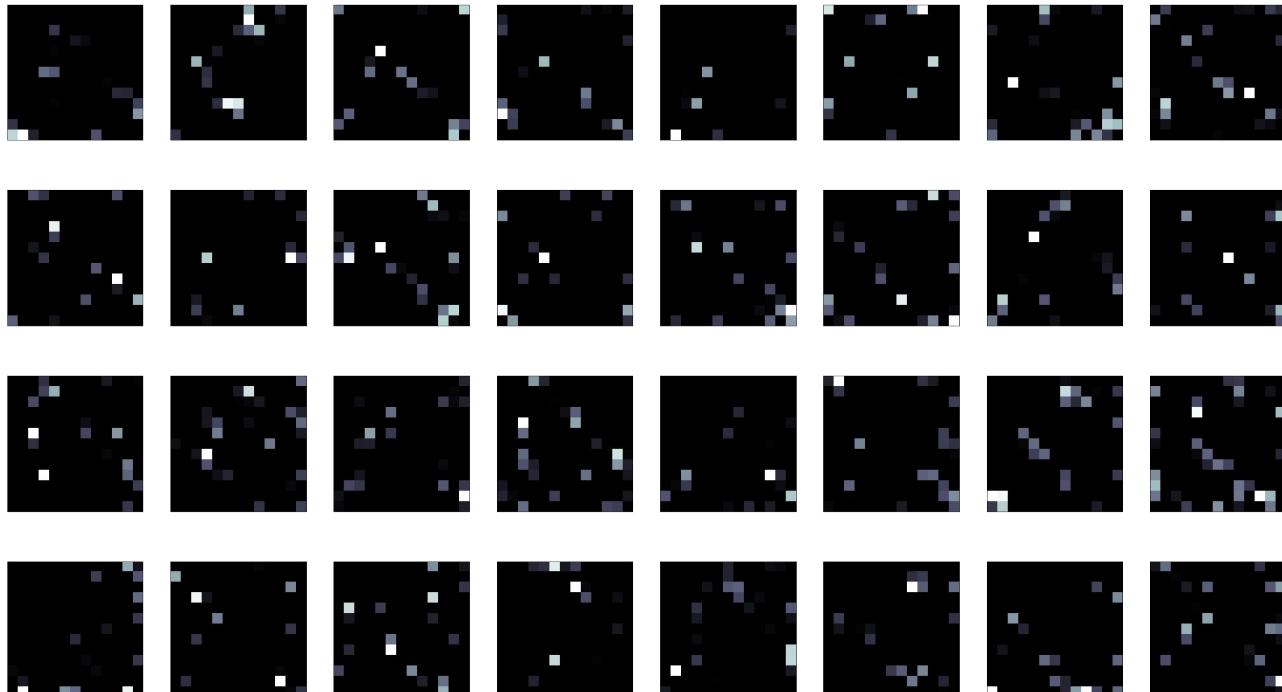


Primo Conv2D



Evoluzione delle feature maps dei Conv2D successivi

- Di seguito è riportato l'output del quarto convolutional layer
- Matrici sparse e regioni attivate sempre più localizzate
- Osservazioni qualitative valide sia per nodi che non nodi

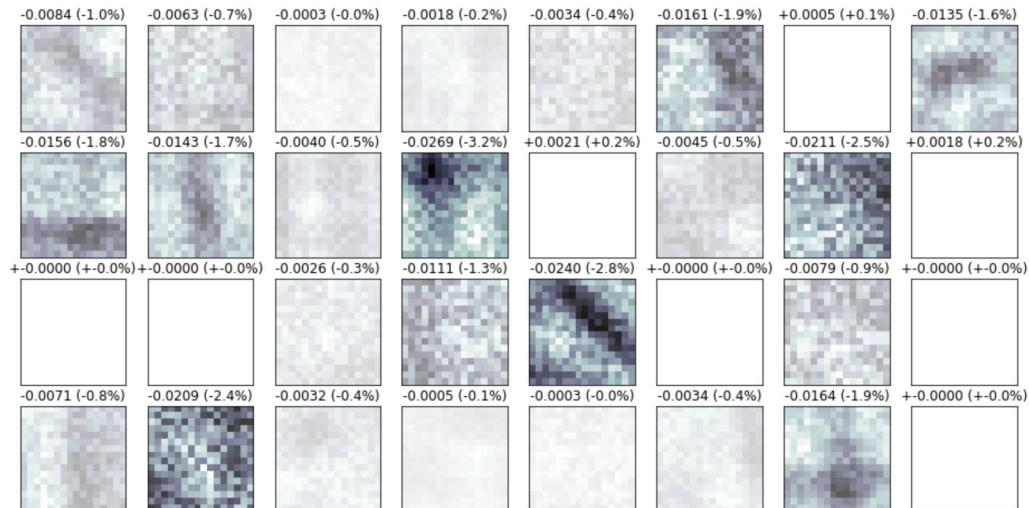
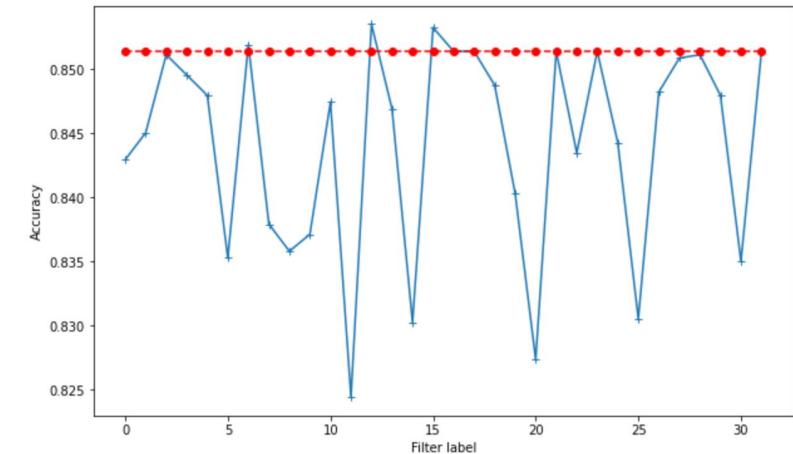


Studio dell'importanza dei filtri

Idea di base:

Attribuire più importanza, ai fini della classificazione, ai filtri del primo Conv2D che fanno diminuire di più l'accuracy se rimossi

- Vi sono filtri più rilevanti per l'accuracy rispetto ad altri
- Alcuni filtri non fanno diminuire l'accuracy se rimossi; altri la fanno aumentare di poco
- Si ha al massimo un drop del 3.2%



Catene da 200 - Classificazione multclasse

Architettura del modello

Obiettivo:

Classificare 5 classi: non nodi (UN), nodi 3_1, 4_1, 5_2 e 5_1

- Stessa architettura del modello precedente
- Generalizzazione a più classi sostituendo
 - Activation function Dense Layer:
Sigmoid —> Softmax
 - Loss function:
Binary cross entropy —> Categorical cross entropy

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 54, 54, 32)	8224
conv2d_1 (Conv2D)	(None, 27, 27, 48)	98352
conv2d_2 (Conv2D)	(None, 27, 27, 48)	36912
max_pooling2d (MaxPooling2D)	(None, 13, 13, 48)	0
conv2d_3 (Conv2D)	(None, 13, 13, 32)	24608
max_pooling2d_1 (MaxPooling2D)	(None, 6, 6, 32)	0
dropout (Dropout)	(None, 6, 6, 32)	0
flatten (Flatten)	(None, 1152)	0
dense (Dense)	(None, 186)	214458
dropout_1 (Dropout)	(None, 186)	0
dense_1 (Dense)	(None, 5)	935

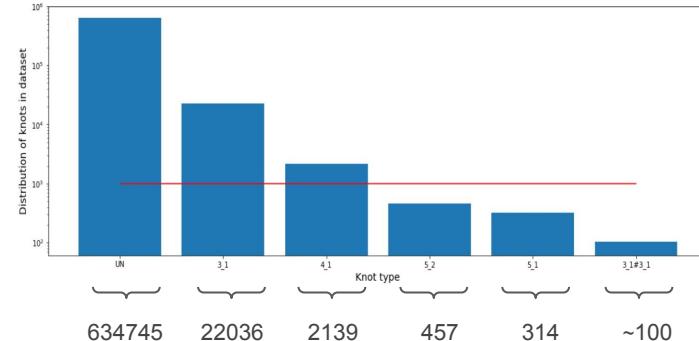
Total params: 383,489
Trainable params: 383,489
Non-trainable params: 0

Catene da 200 - Classificazione multclasse

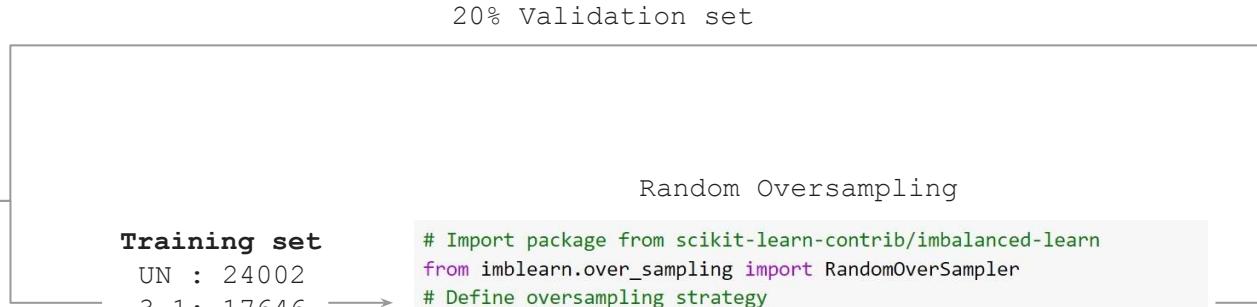
Random Oversampling

Bilanciamento del dataset:

- Random Undersampling della majority class
- Random Oversampling delle classi con meno samples



Dataset
UN : 30000
3_1: 22036
4_1: 2139
5_2: 457
5_1: 314
TOT: 54946



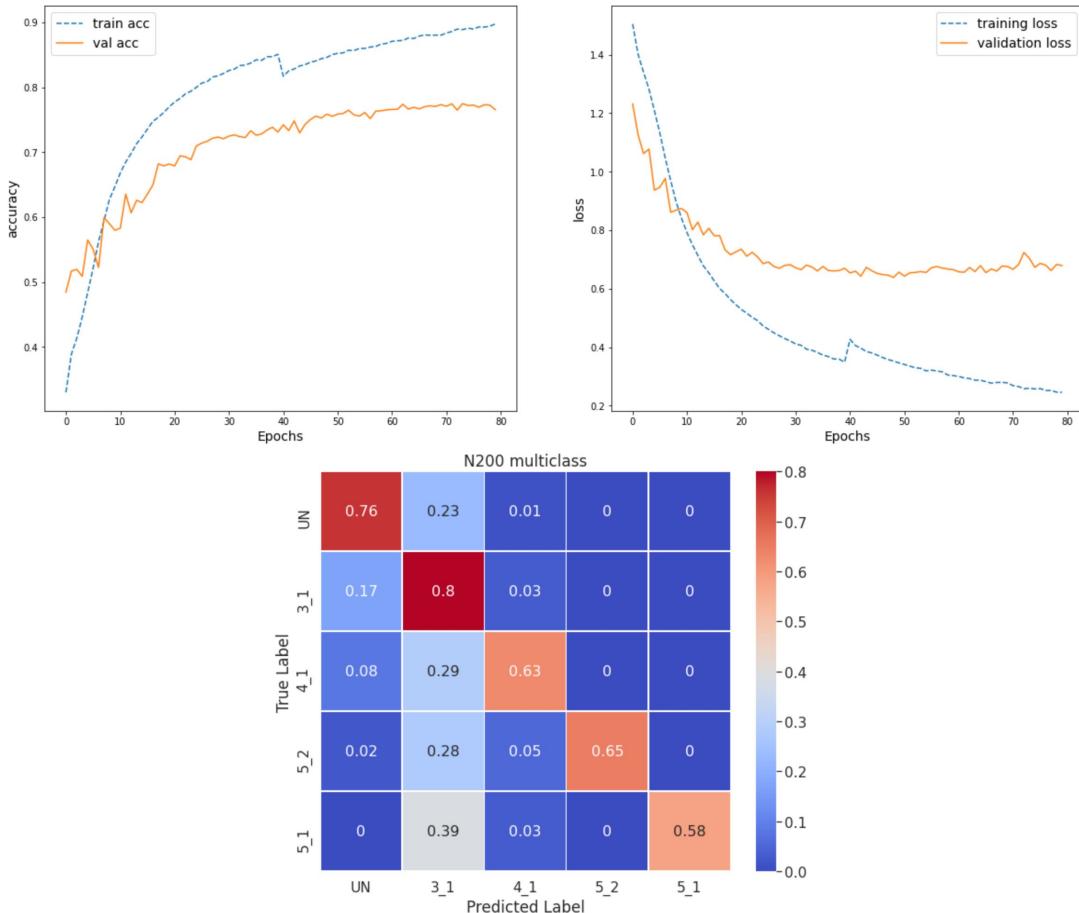
Validation set
UN : 5998
3_1: 4390
4_1: 439
5_2: 96
5_1: 67
TOT: 10990

Training set
UN : 24002
3_1: 17646
4_1: 17646
5_2: 8800
5_1: 8800
TOT: 76894

Catene da 200 - Classificazione multclasse

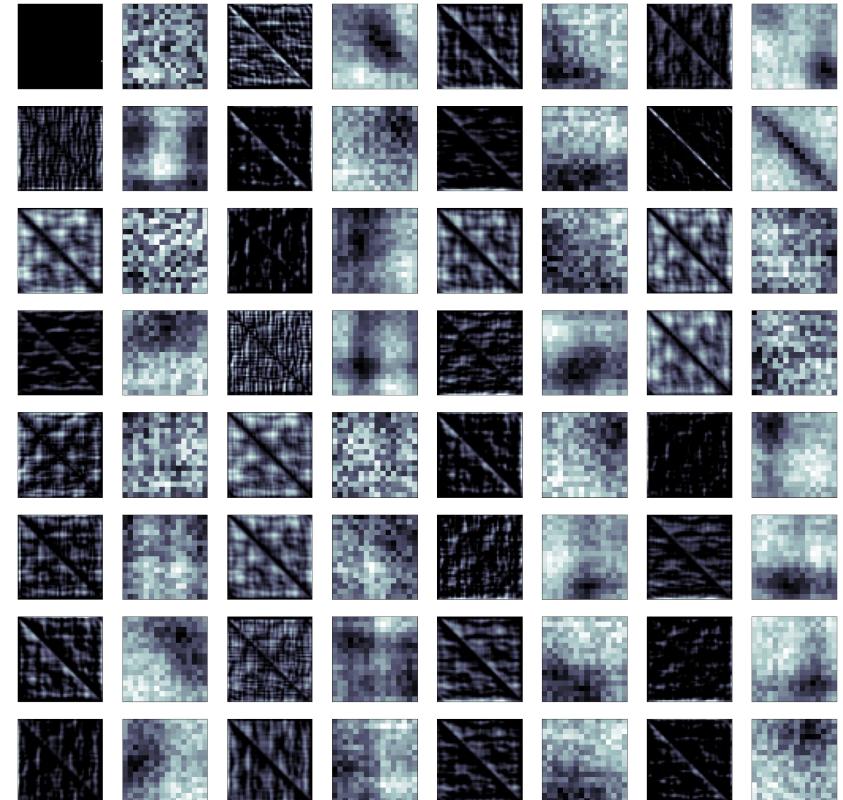
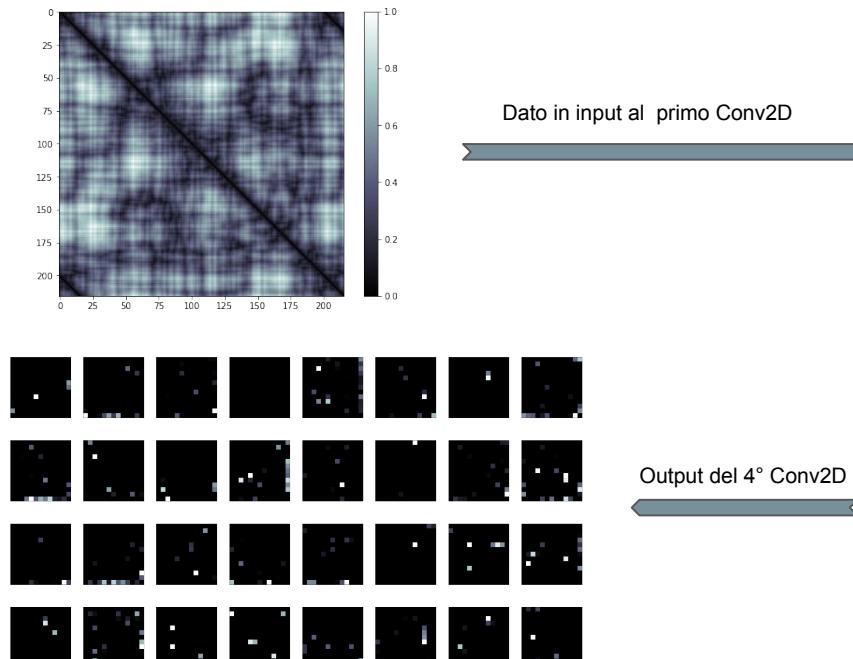
I risultati

- Accuracy ≈ 0.77
- Confusion matrix: il 76% dei non nodi e l'80% dei 3_1 sono classificati correttamente
- I nodi 4_1, 5_2 e 5_1 sono più spesso confusi come 3_1 che come UN.



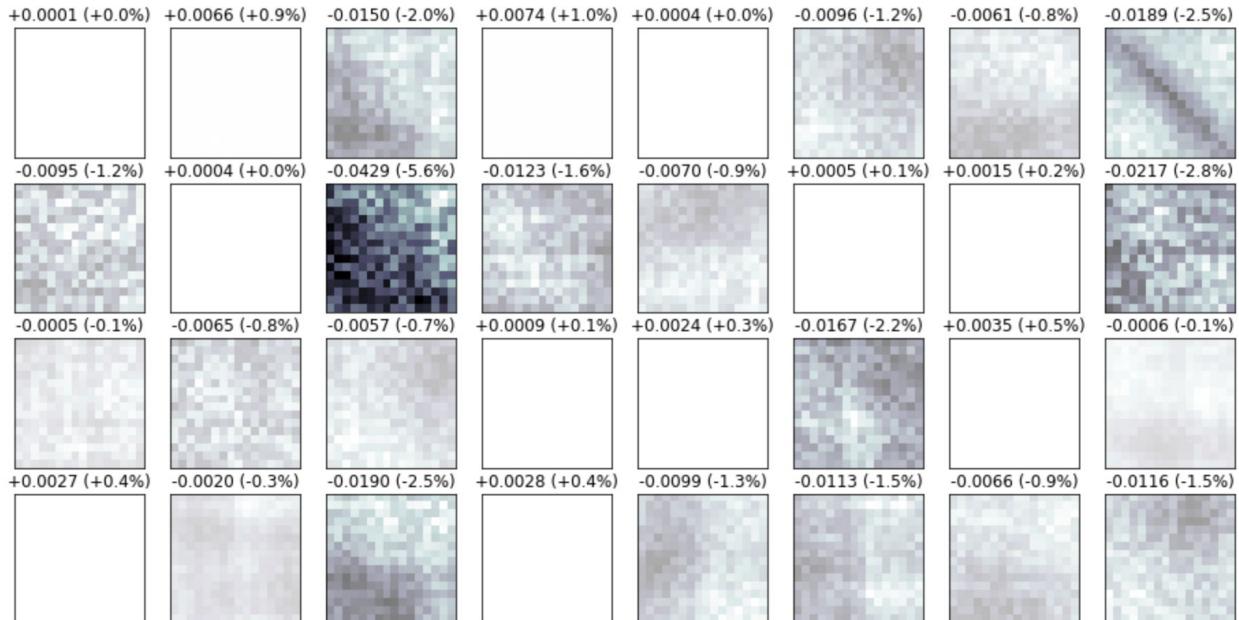
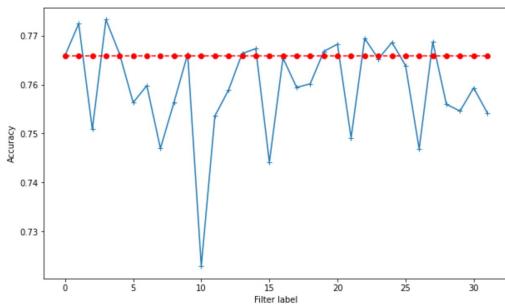
Feature maps e visualizzazione dei rispettivi filtri

- Stesse considerazioni qualitative della classificazione binaria
- Osservazioni qualitative valide per tutte le classi



Studio dell'importanza dei filtri

- Stesso approccio avuto per la classificazione binaria
- Conclusioni simili, con il massimo drop di accuracy pari al 5.6 %



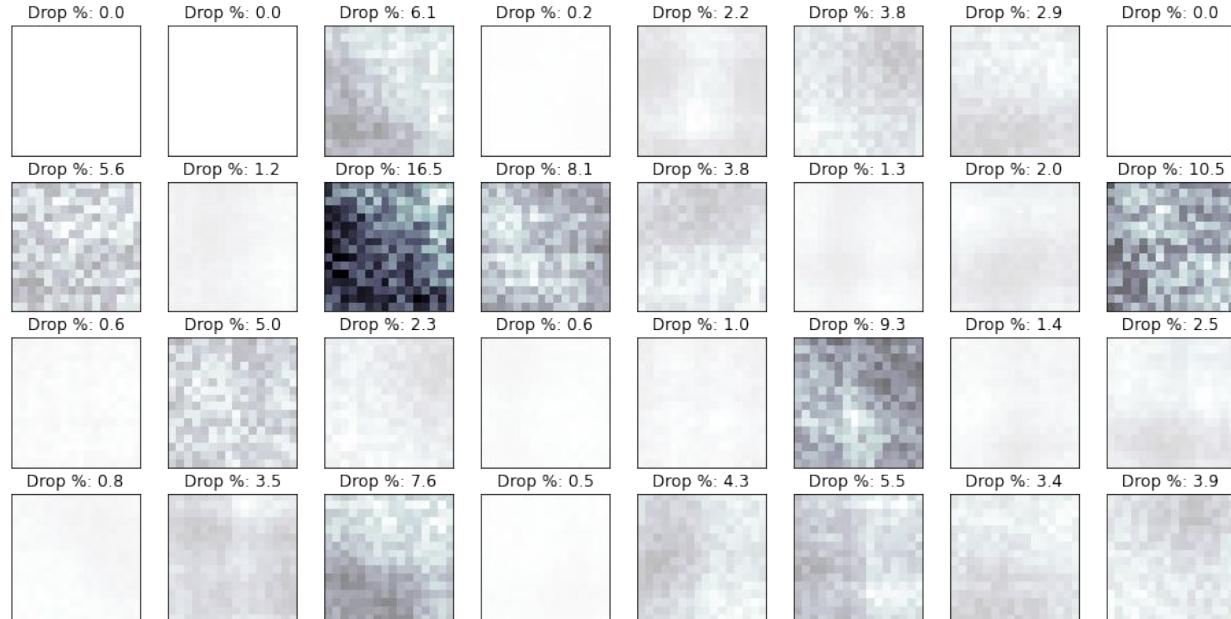
Studio dell'importanza dei filtri per ogni classe

Idea di base:

Si rimuove un filtro alla volta e si calcola l'accuracy *solo* per i dati del validation set appartenenti ad una true label che sono classificati con una predicted label a scelta

- Diversi filtri sembrano essere specializzati per caratterizzare le classi UN (drop fino a 16.5%) 3_1 (drop fino a 26.4%) 4_1 (drop fino al 4%)
- Comportamento peculiare per i 5_2 e i 5_1, da approfondire

True label: UN
Pred label: UN



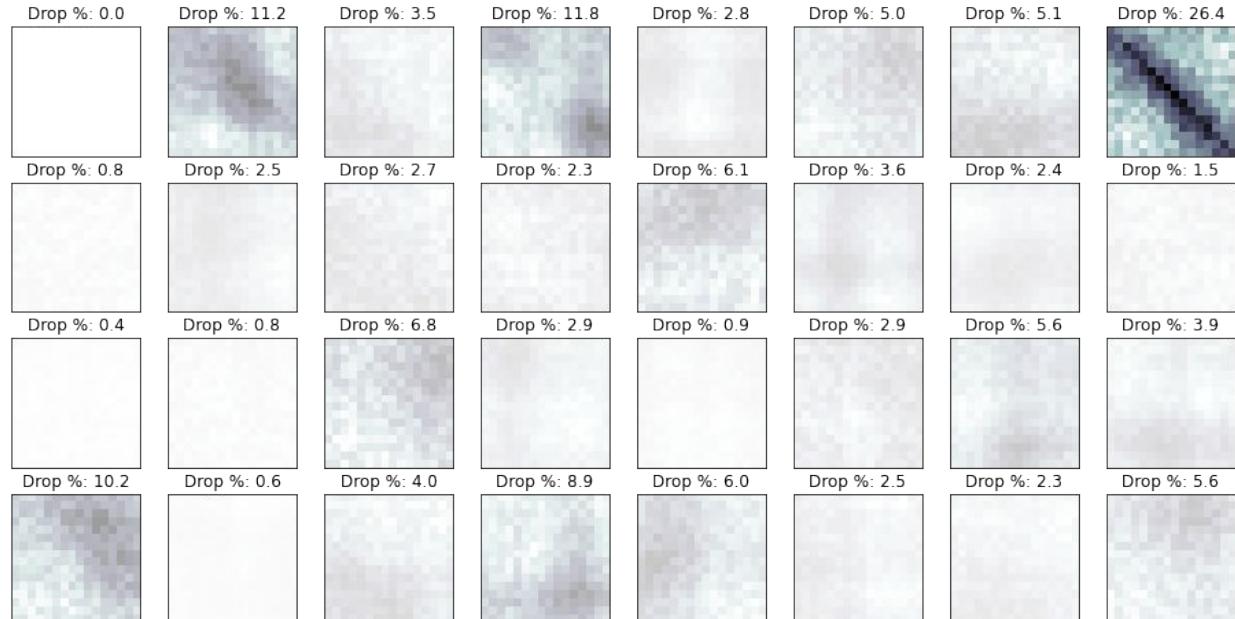
Studio dell'importanza dei filtri per ogni classe

Idea di base:

Si rimuove un filtro alla volta e si calcola l'accuracy solo per i dati del validation set appartenenti ad una true label che sono classificati con una predicted label a scelta

- Diversi filtri sembrano essere specializzati per caratterizzare le classi UN (drop fino a 16.5%)
3_1 (drop fino a 26.4%)
4_1 (drop fino al 4%)
- Comportamento peculiare per i 5_2 e i 5_1, da approfondire

True label: 3_1
Pred label: 3_1



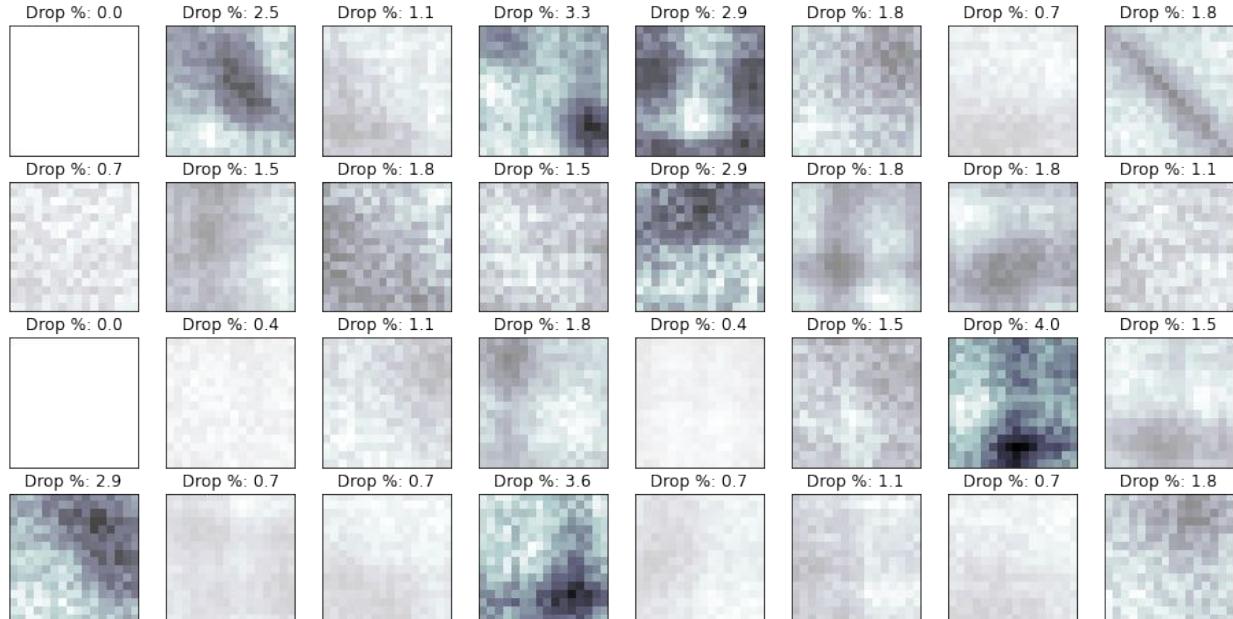
Studio dell'importanza dei filtri per ogni classe

Idea di base:

Si rimuove un filtro alla volta e si calcola l'accuracy solo per i dati del validation set appartenenti ad una true label che sono classificati con una predicted label a scelta

- Diversi filtri sembrano essere specializzati per caratterizzare le classi UN (drop fino a 16.5%)
3_1 (drop fino a 26.4%)
4_1 (drop fino al 4%)
- Comportamento peculiare per i 5_2 e i 5_1, da approfondire

True label: 4_1
Pred label: 4_1



Catene da 400 - Classificazione binaria

Architettura modello

Idea di base:

Utilizzare la stessa architettura del 200 binary

Layer (type)	Output Shape	Param #
<hr/>		
conv2d (Conv2D)	(None, 104, 104, 32)	8224
conv2d_1 (Conv2D)	(None, 52, 52, 48)	98352
conv2d_2 (Conv2D)	(None, 52, 52, 48)	36912
max_pooling2d (MaxPooling2D)	(None, 26, 26, 48)	0
conv2d_3 (Conv2D)	(None, 26, 26, 32)	24608
max_pooling2d_1 (MaxPooling2D)	(None, 13, 13, 32)	0
dropout (Dropout)	(None, 13, 13, 32)	0
flatten (Flatten)	(None, 5408)	0
dense (Dense)	(None, 186)	1006074
dropout_1 (Dropout)	(None, 186)	0
dense_1 (Dense)	(None, 1)	187
<hr/>		
Total params:	1,174,357	
Trainable params:	1,174,357	
Non-trainable params:	0	

None

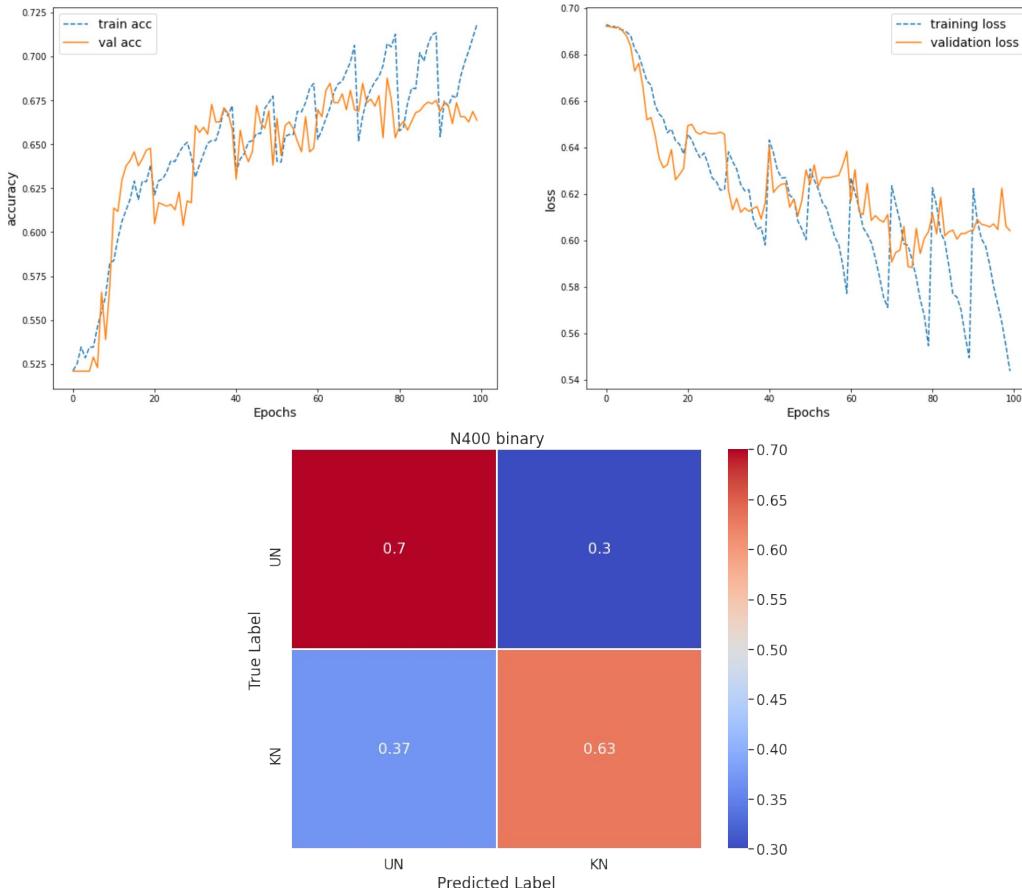
```
model = Sequential()
model.add( Conv2D(filters = 32,
                  kernel_size = 16,
                  activation = 'relu',
                  input_shape = (416, 416, 1),
                  padding = 'same',
                  strides = (4,4)
                 ) )
model.add( Conv2D(filters = 48,
                  kernel_size = 8,
                  activation = 'relu',
                  padding = 'same',
                  strides = (2,2)
                 ) )
model.add( Conv2D(filters = 48,
                  kernel_size = 4,
                  activation = 'relu',
                  padding = 'same',
                  ) )
model.add( MaxPooling2D(pool_size = (2,2),
                       strides = (2,2),
                       padding='valid'
                      ) )
model.add( Conv2D(filters = 32,
                  kernel_size = 4,
                  activation = 'relu',
                  padding = 'same',
                  ) )
model.add( MaxPooling2D(pool_size = (2,2),
                       padding='valid'
                      ) )
model.add( Dropout(.35) )
model.add( Flatten() )
model.add( Dense(186,
                 activation = 'relu',
                 ) )
model.add( Dropout(.75) )
model.add(Dense(n_class, activation='sigmoid'))


# Compiling model
model.compile(loss = binary_crossentropy,
              optimizer = optimizers.Nadam(learning_rate = 0.0001),
              metrics = ['accuracy'])
```

Catene da 400 - Classificazione binaria

Risultati del training

- Dataset:
66800 sequenze, 15% validation
- Accuracy ≈ 0.67
- Confusion matrix: il modello è in grado di riconoscere il 70% di non nodi e il 63% di nodi
- Difficoltà tecniche: OOM

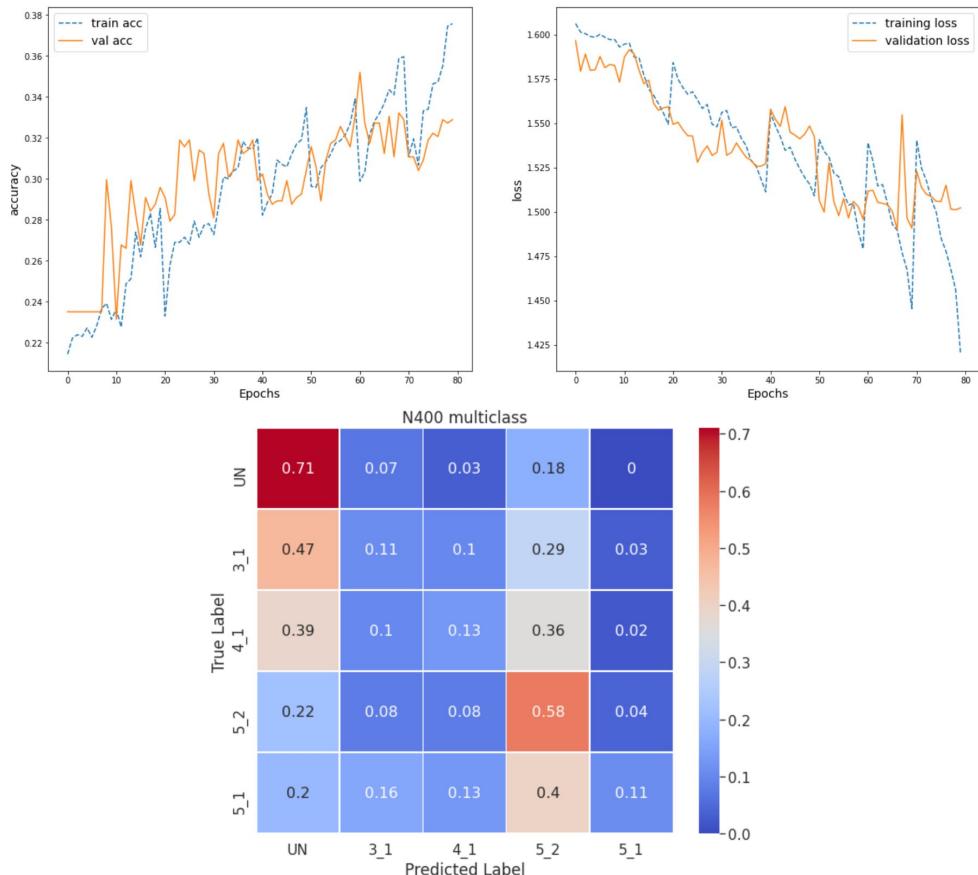


Catene da 400 - Classificazione multclasse

- Generalizzazione multclasse dell'architettura del modello binario
- Dataset:
24167 sequenze, 15% validation

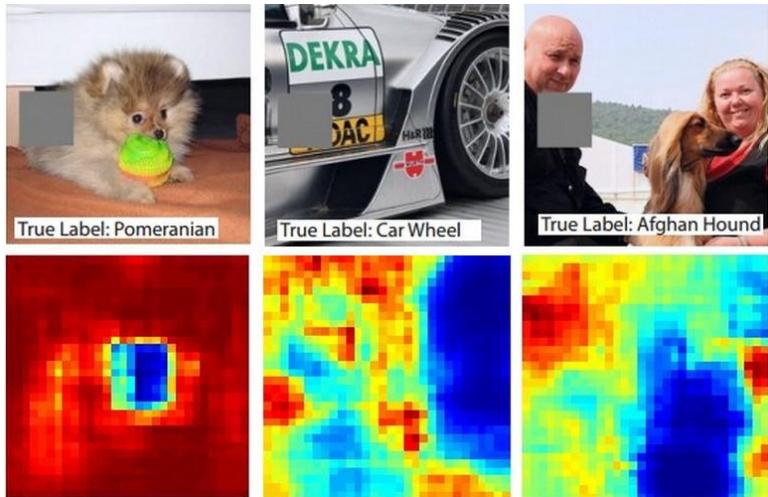
UN : 355034	UN : 5241
3_1: 57678	3_1: 5241
4_1: 11112	4_1: 5241
5_2: 5241	5_2: 5241
5_1: 3203	5_1: 3203

→



Heatmaps

Finestre quadrate



M. D. Zeiler, R. Fergus (2013)

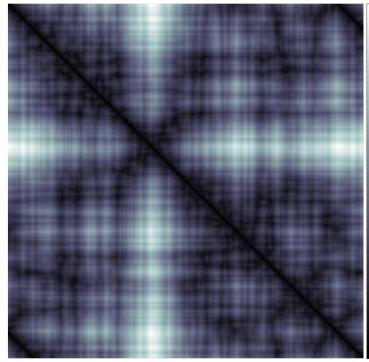
Image segmentation: metodo Felzenszwalb



P. F. Felzenszwalb, D. P. Huttenlocher (2004)

Heatmaps

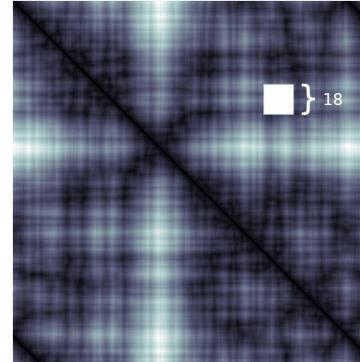
Originale



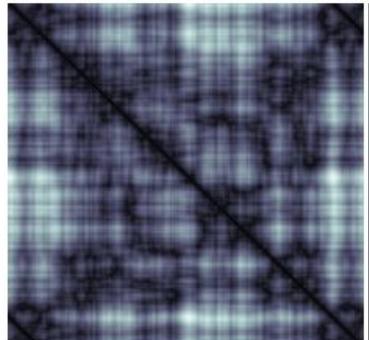
Finestra quadrata



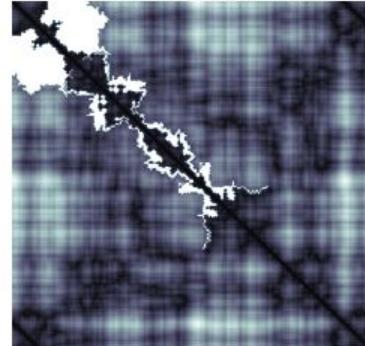
Ritagliato



18



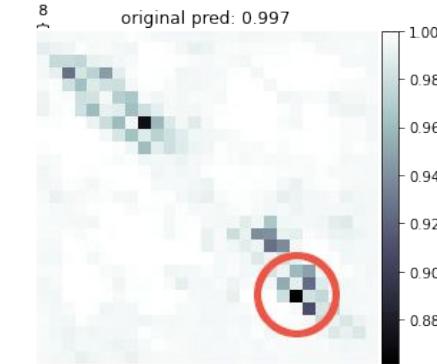
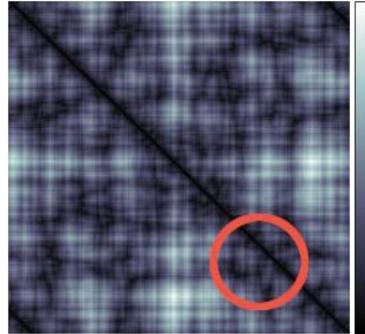
Segmentation



Catene da 200 - Classificazione binaria

Nodo

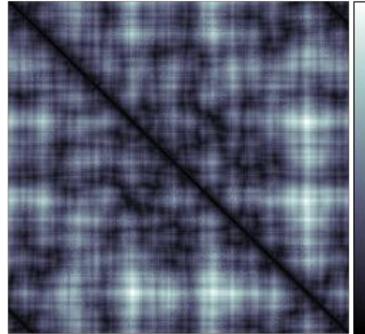
original pred: 0.997



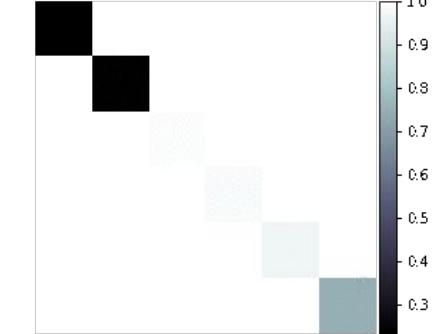
- Asimmetria
- Importante variazione della predizione rimuovendo piccole regioni
- Diagonale come discriminante tra nodo e non nodo

Nodo

original pred: 0.967

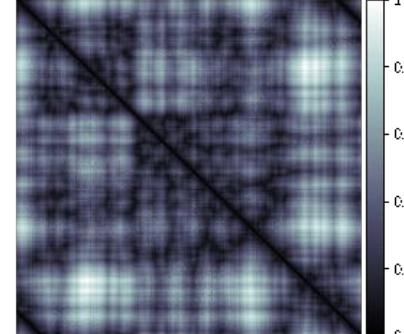


original pred: 0.967

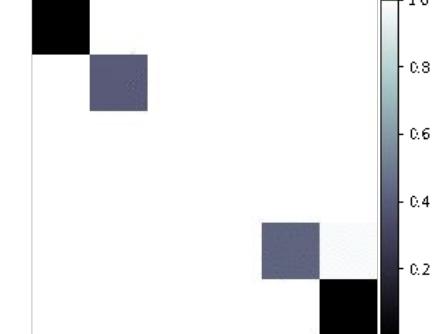


Non-nodo

original pred: 0.006

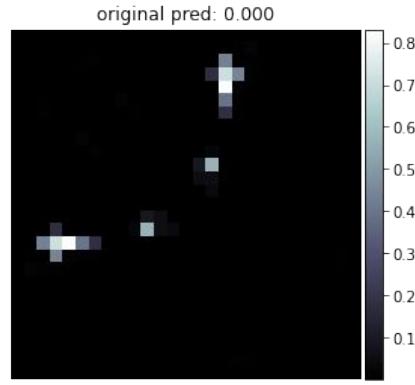
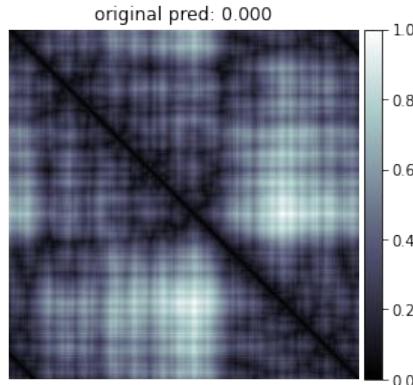


original pred: 0.006



Catene da 200 - Classificazione binaria

Asimmetria



Il metodo del *One pixel attack*

AllConv



SHIP
CAR(99.7%)

NiN



HORSE
FROG(99.9%)

VGG

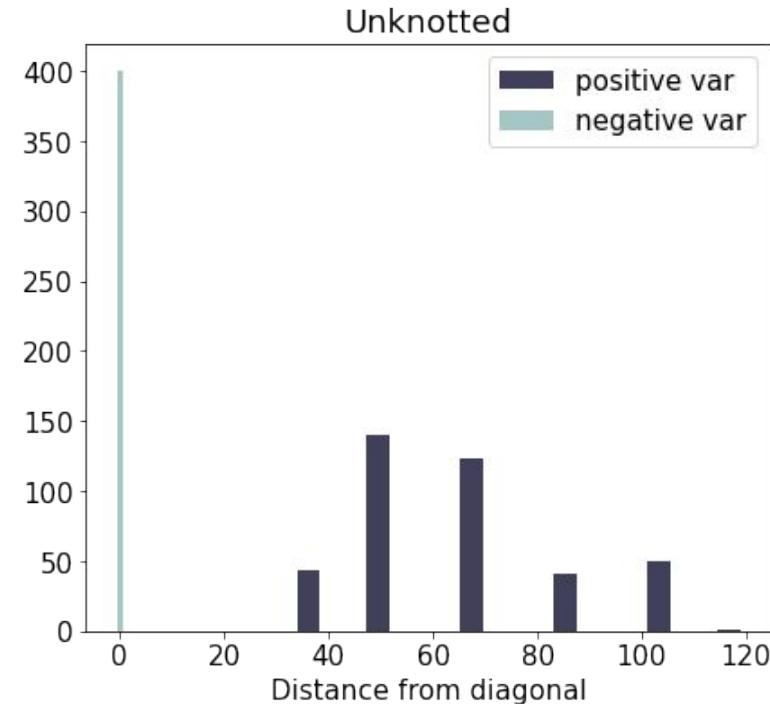
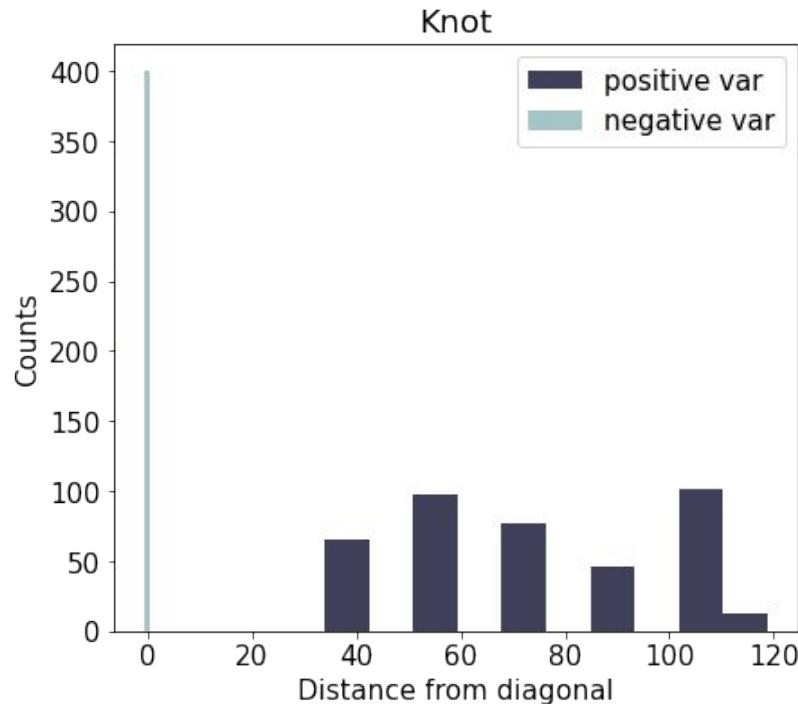


DEER
AIRPLANE(85.3%)

J. Su, D. V. Vargas and K. Sakurai (2019)

Catene da 200 - Classificazione binaria

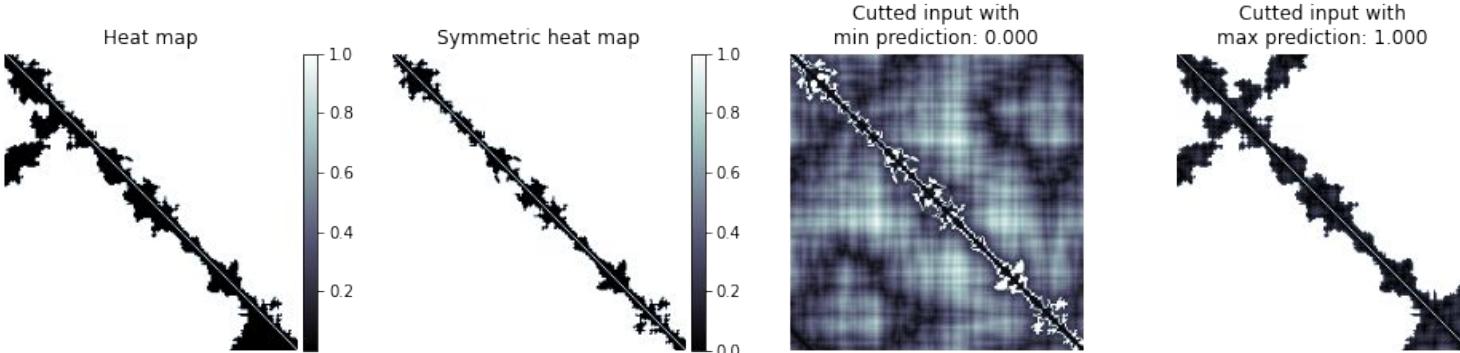
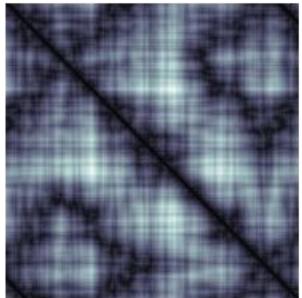
Distribuzione distanze dalla diagonale delle finestre con maggior variazione



Catene da 200 - Classificazione binaria

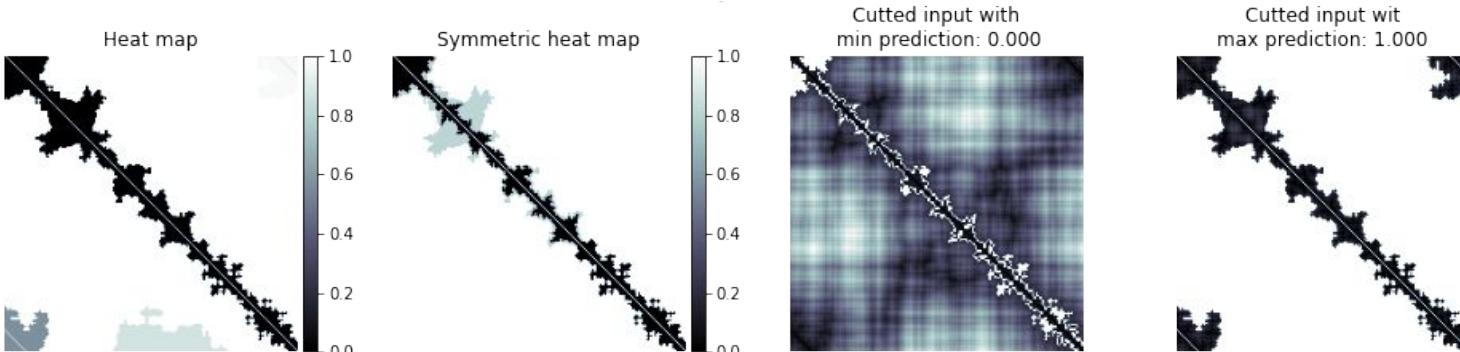
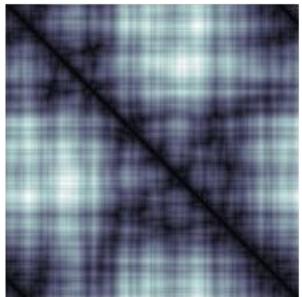
Nodo

original pred: 0.867



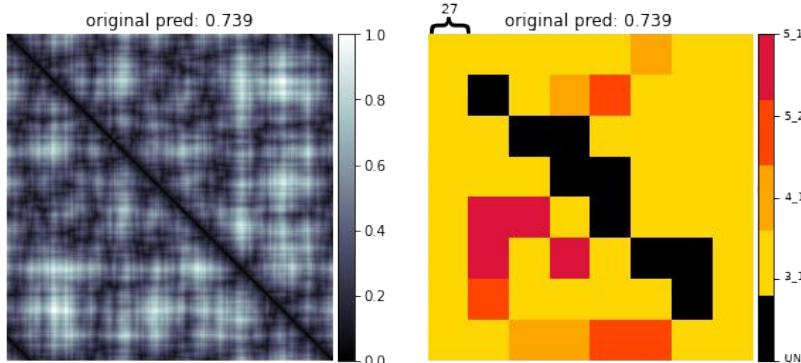
Non nodo

original pred: 0.000



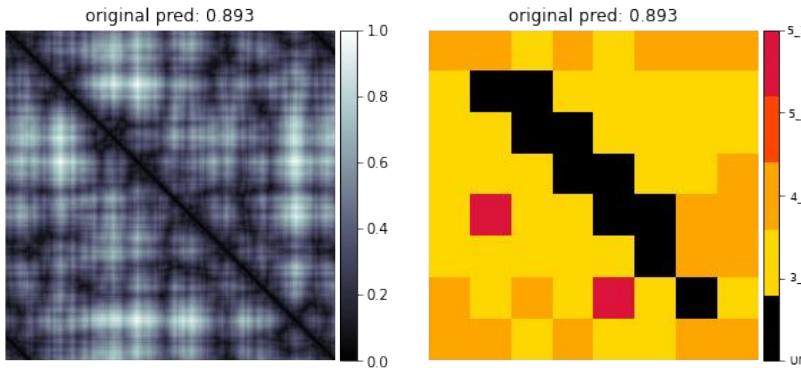
Catene da 200 - Classificazione multclasse

3_1 correttamente predetto

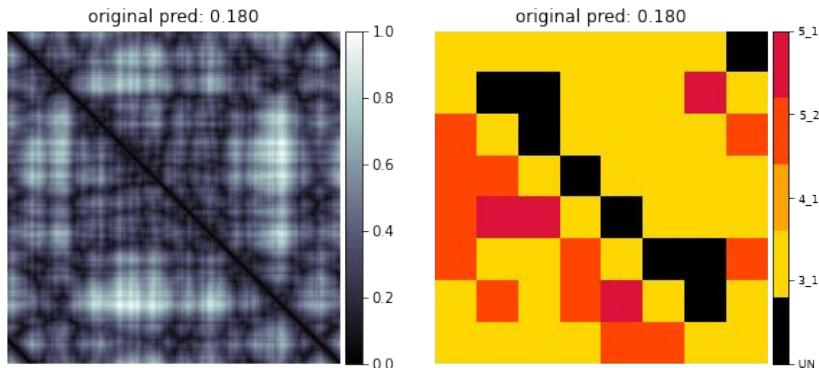


- Conferma di quanto visto per classificazione binaria
- Informazione relativa alla classe situata fuori dalla diagonale

4_1 correttamente predetto



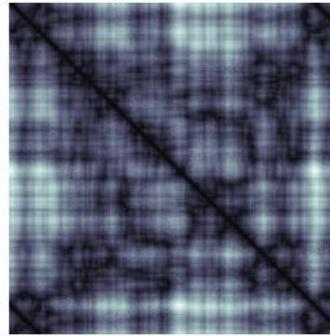
UN erroneamente predetto come 3_1



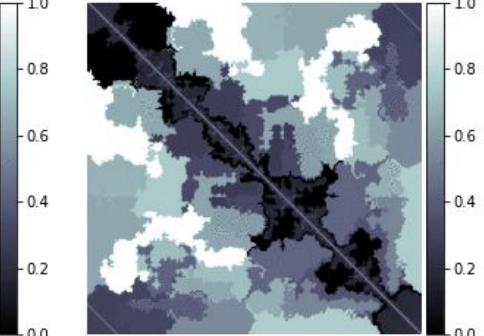
Catene da 400 - Classificazione binaria

Nodi

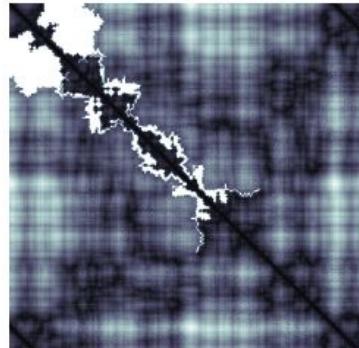
original pred: 0.386



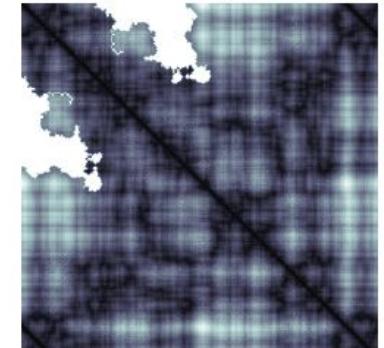
Heat map



Cutted input with min prediction: 0.177

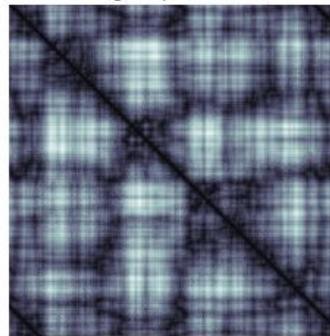


Cutted input with max prediction: 0.782



Non nodi

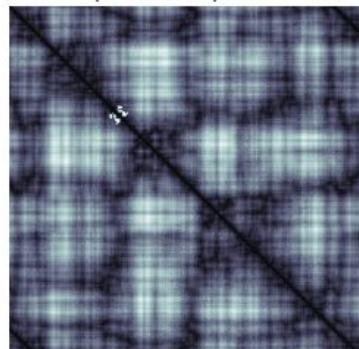
original pred: 0.248



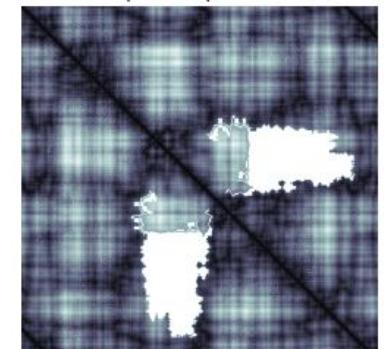
Heat map



Cutted input with min prediction: 0.154



Cutted input max prediction: 0.834

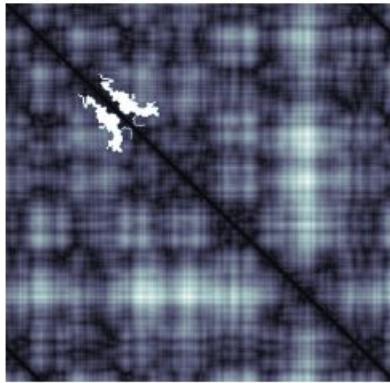


Catene da 400 - Classificazione binaria

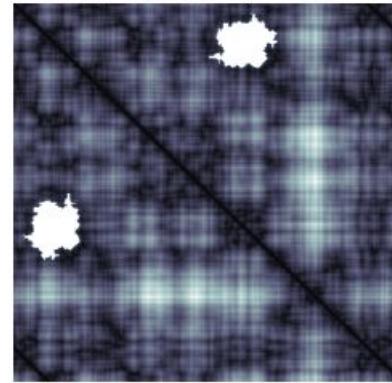
400

Nodo Localizzato

Cutted input with min prediction: 0.361



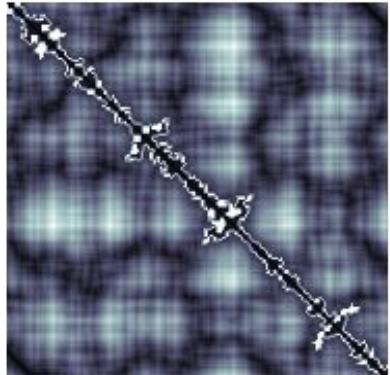
Cutted input with max prediction: 0.814



200

Nodo delocalizzato

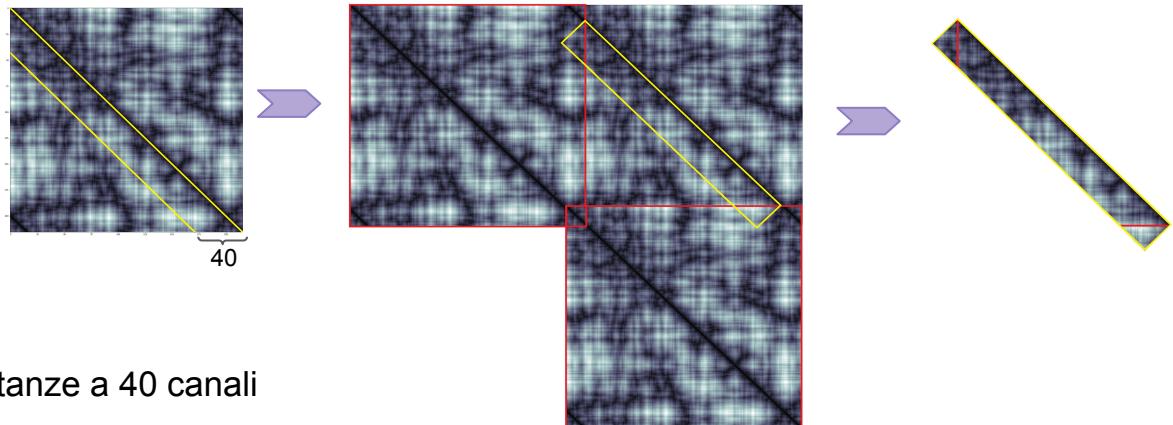
Cutted input with
min prediction: 0.000



Cutted input with
max prediction: 1.000



Estrazione degli elementi
più influenti dalla matrice
delle distanze:

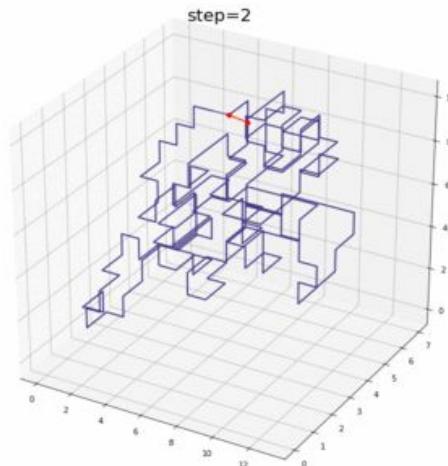


Nuovo input: vettore delle distanze a 40 canali

```
[[[1.      0.618  0.547  ...  0.673  0.6855 0.689 ] step = 2
 [1.      1.      0.6763 ... 0.662   0.6855 0.678 ] step = 3
 [0.      0.618  0.6763 ... 0.5703  0.6855 0.689 ] step = 4
 ...
 [0.      0.618  0.4004 ... 0.3604  0.3142 0.4111] step = 40
 [0.      0.618  0.547  ... 0.4663  0.4497 0.4111] step = 41
 [0.      0.618  0.547  ... 0.4663  0.5083 0.4702]]] step = 42
```

200

padding
periodico

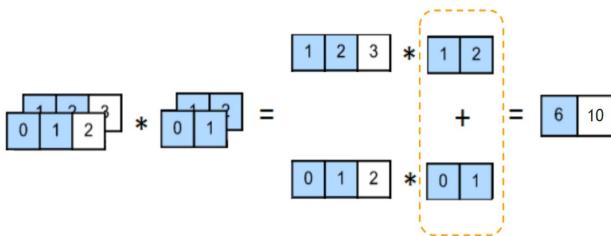


Prima classificazione con una CNN

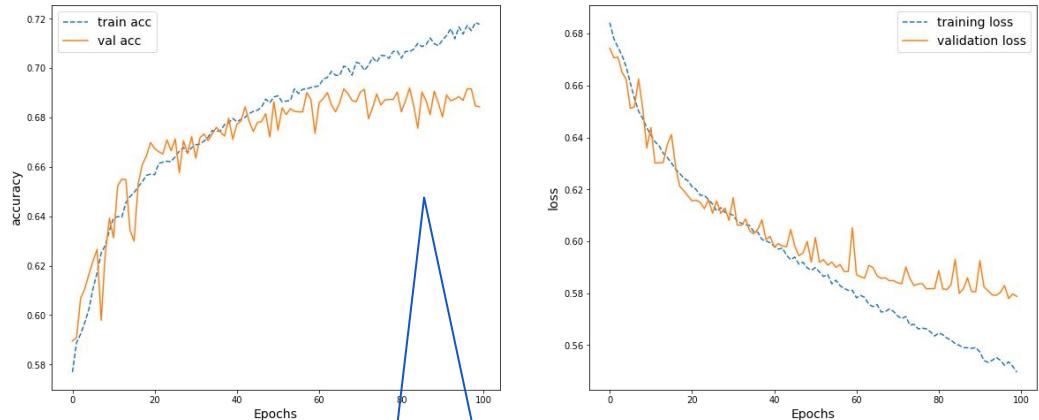
Architettura del modello:

```
Model: "sequential_1"
Layer (type)      Output Shape       Param #
===== =====
conv1d (Conv1D)   (None, 100, 64)    163904
max_pooling1d (MaxPooling1D) (None, 50, 64) 0
conv1d_1 (Conv1D)  (None, 35, 128)   131200
max_pooling1d_1 (MaxPooling1 (None, 17, 128) 0
conv1d_2 (Conv1D)  (None, 10, 128)   131200
dropout_10 (Dropout) (None, 21, 128) 0
flatten_5 (Flatten) (None, 2688)    0
dense_10 (Dense)   (None, 186)      500154
dropout_11 (Dropout) (None, 186)    0
dense_11 (Dense)   (None, 1)        187
=====
Total params: 426,433
Trainable params: 426,433
Non-trainable params: 0
```

Convoluzione multichannel:



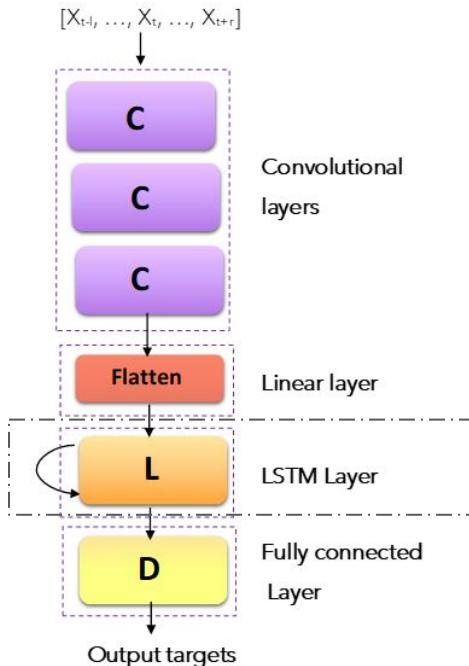
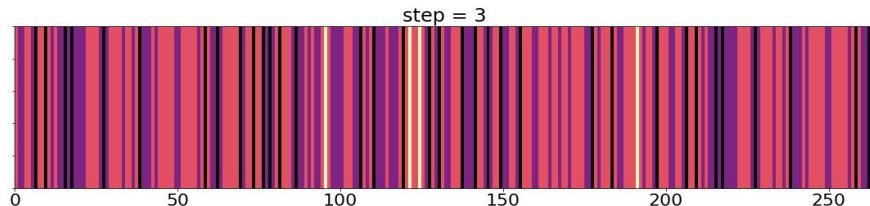
Risultati:



- Decrescita consistente
dell'accuracy

CNN + LSTM

Input interpretabili come sequenze di vettori:



Composto da n blocchi indipendenti con diversa inizializzazione ma uguale struttura:

Input gate:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

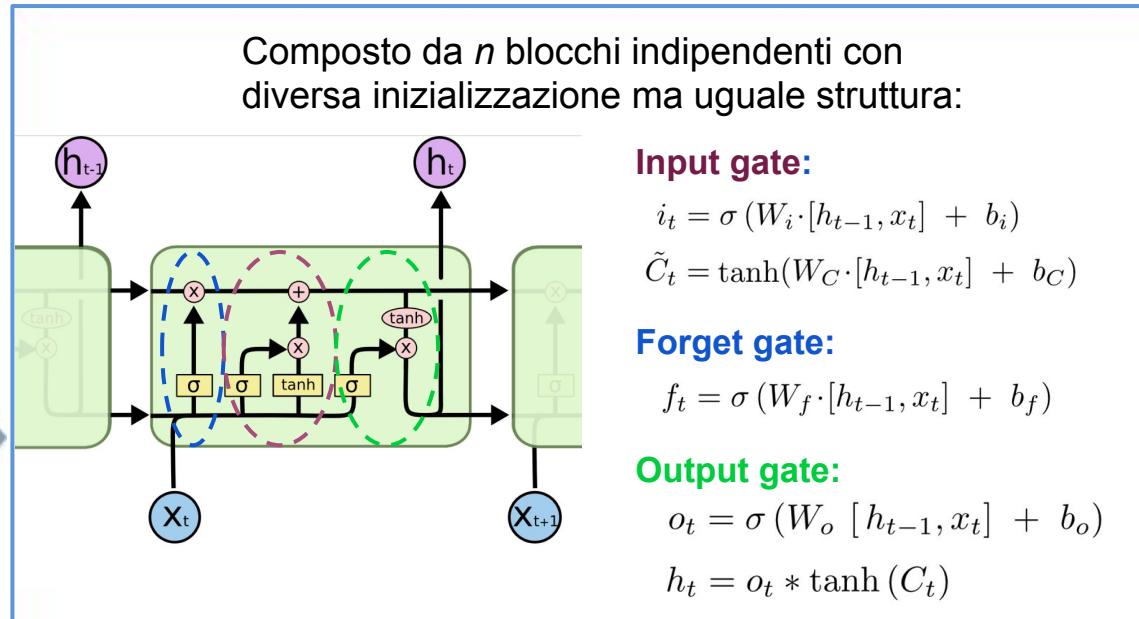
Forget gate:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

Output gate:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$



Architettura del modello

Model: "sequential_7"

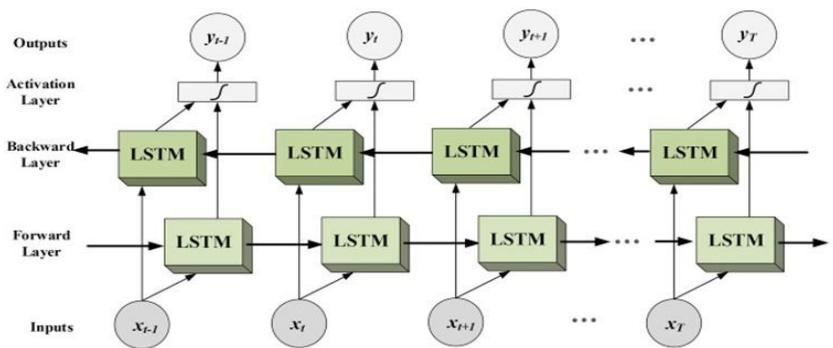
Layer (type)	Output Shape	Param #
time_distributed_9 (TimeDist)	(None, 40, 201, 64)	704
Conv1D		
time_distributed_10 (TimeDis)	(None, 40, 100, 64)	0
MaxPool1D		
time_distributed_11 (TimeDis)	(None, 40, 97, 128)	32896
Conv1D		
time_distributed_12 (TimeDis)	(None, 40, 48, 128)	0
MaxPool1D		
time_distributed_13 (TimeDis)	(None, 40, 45, 128)	65664
Conv1D		
dropout_2 (Dropout)	(None, 40, 45, 128)	0
time_distributed_14 (TimeDis)	(None, 40, 5760)	0
Flatten		
bidirectional (Bidirectional)	(None, 1000)	25044000
dropout_3 (Dropout)	(None, 1000)	0
dense_2 (Dense)	(None, 1)	1001
dense_3 (Dense)	(None, 1)	2
Total params:	25,144,267	
Trainable params:	25,144,267	
Non-trainable params:	0	

TimeDistributed Wrapper applicato ai Convolutional, Flatten e Pooling layer

Applicati separatamente ad ogni canale

Bidirectional wrapper applicato al livello LSTM

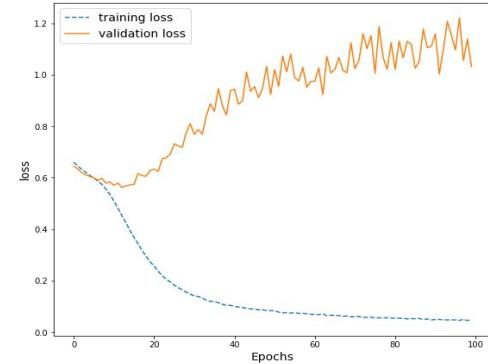
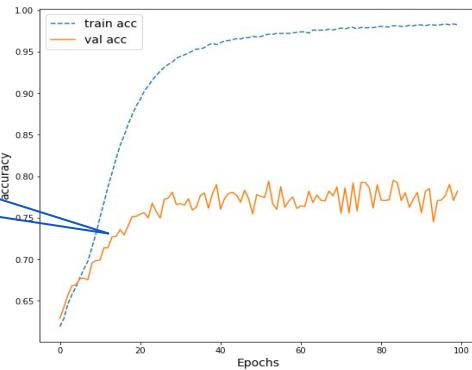
Invarianza per inversione dell'ordine dei canali



Risultati della classificazione binaria

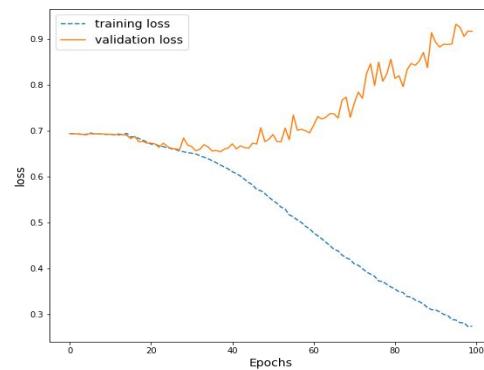
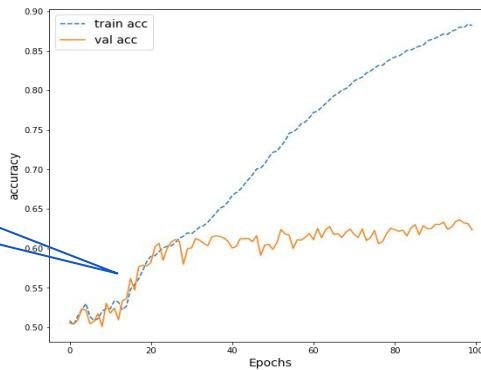
Polimeri a 200 passi:

- Validation accuracy finale: 78.22%
- Overfit dopo 30 epoch



Polimeri a 400 passi:

- Validation accuracy finale: 64.2%
- Overfit dopo 20 epoch



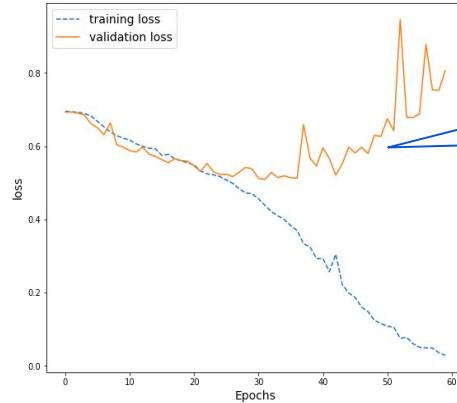
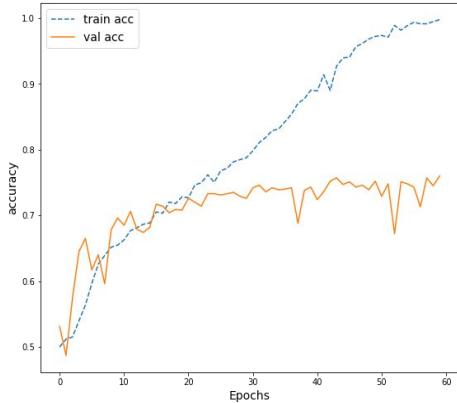
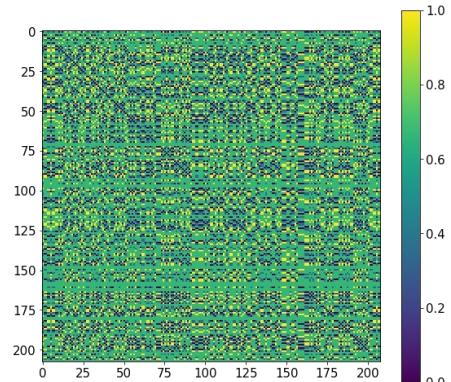
Nuovi tipi di input: riparametrizzazione delle sequenze con gli angoli diedri

1. Matrice di $n \times n$ angoli diedri generalizzati

Utilizzata come secondo canale
alla matrice delle distanze e
allenata con la stessa CNN

elemento
della matrice

$$\vec{\alpha}_l = \vec{v}_l \times \vec{v}_{l-1}$$
$$\theta_{ij} = \vec{\alpha}_l \cdot \vec{v}_{j+1}$$

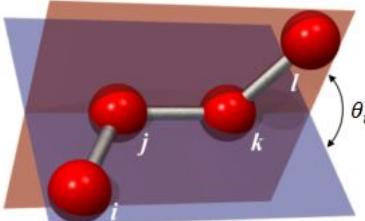


- Overfit dopo 40 epoch
- Diminuzione della validation accuracy

2. Sequenza di n angoli diedri propri

$$\theta_i = \theta_{ii}$$

angolo diedro generalizzato



4 possibili valori:

[-3.14159265 -1.57079633 0. 1.57079633]

standardizzazione

[-0.08010042 -0.03054413 0.01901216 0.06856845]

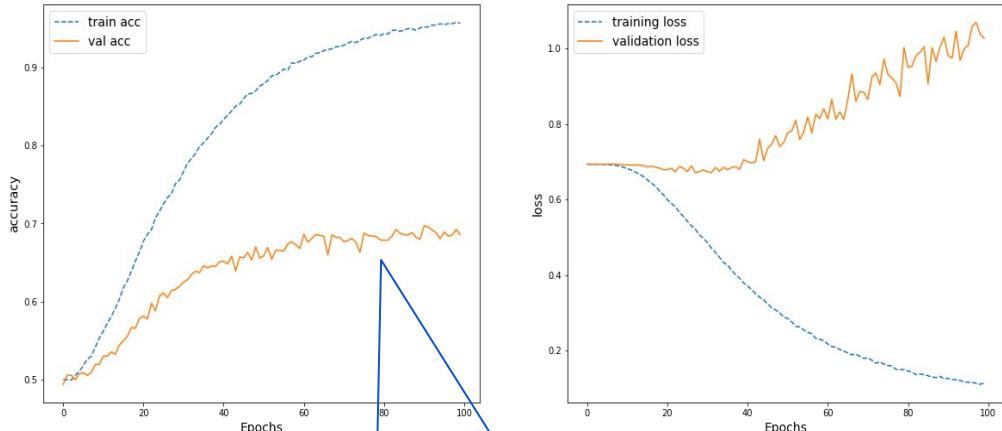
Architettura del modello:

CNN + LSTM

Model: "sequential"

Layer (type)	Output Shape	Param #
<hr/>		
conv1d (Conv1D)	(None, 101, 128)	2688
conv1d_1 (Conv1D)	(None, 82, 64)	163904
dropout (Dropout)	(None, 82, 64)	0
bidirectional (Bidirectional)	(None, 800)	1488000
dropout_1 (Dropout)	(None, 800)	0
dense (Dense)	(None, 1)	801
<hr/>		
Total params:	1,655,393	
Trainable params:	1,655,393	

Risultati:



- Validation accuracy finale: 69.2%
- Overfit dopo 40 epoch

Conclusioni e possibili applicazioni

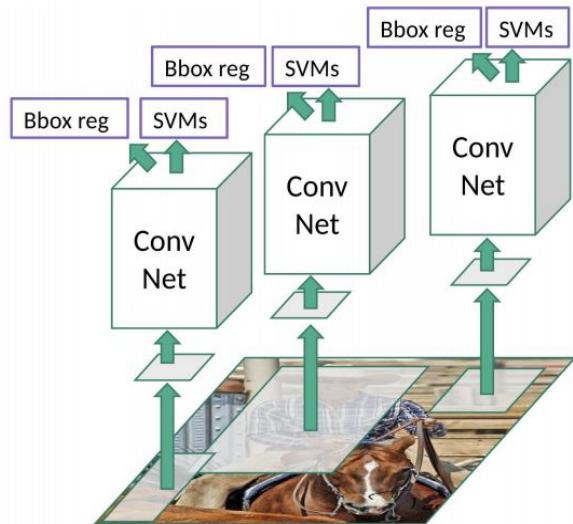
Tipi di input	Lunghezza dei polimeri	Tipi di classificatore	Accuracy
Matrice delle distanze	200	Binary	0.85
		Multiclass	0.77
	400	Binary	0.67
		Multiclass	0.33
Vettore delle distanze a 40 canali	200	Binary	0.78
	400		0.64
Vettore di angoli diedri	200		0.69

- Modelli utilizzabili per una classificazione preliminare di un dataset di catene chiuse

Outlooks

Object detection

R-CNN



YOLO

