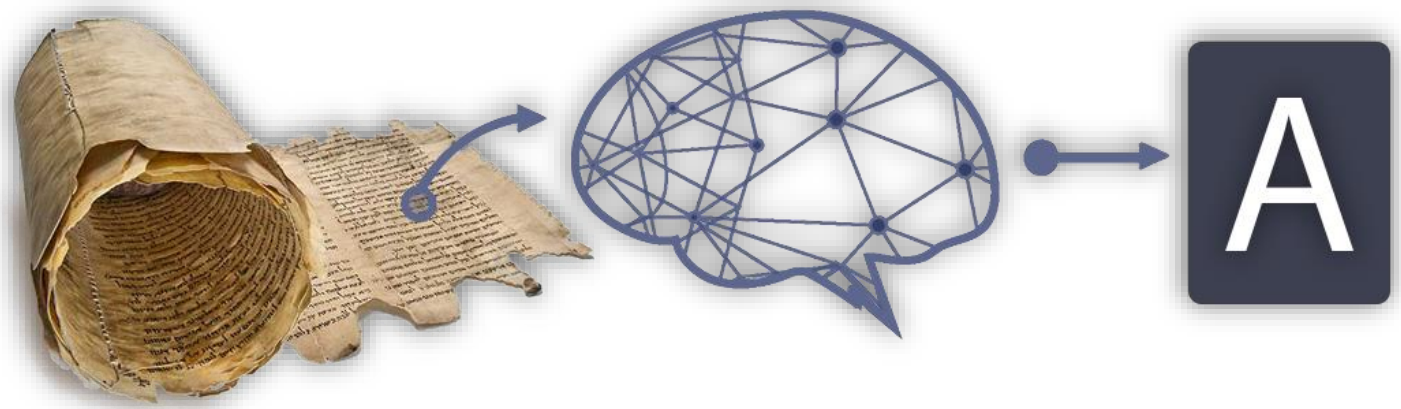
In Codice Ratio

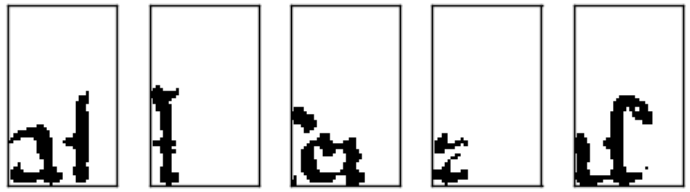
OCR with Convolutional Neural Network



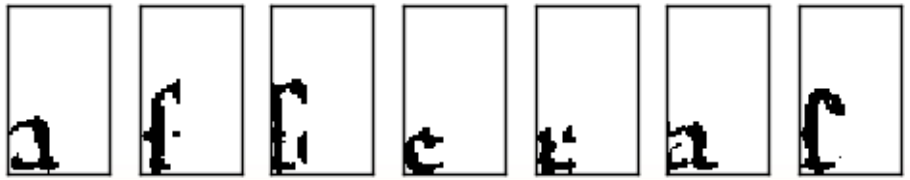
Gaetano Bonofiglio, Veronica Iovinella, Andrea Salvoni

Il problema

- **In Codice Ratio** (ICR) è un progetto curato dall'*Università degli Studi di Roma Tre* in collaborazione con l'*Archivio Segreto dello Stato del Vaticano*. Tale progetto ha lo scopo di digitalizzare i documenti e i testi antichi contenuti nell'Archivio.
- Il problema che abbiamo affrontato è solo una parte di ICR, ed è quello di classificare le in scrittura carolina al fine di riconoscerle.

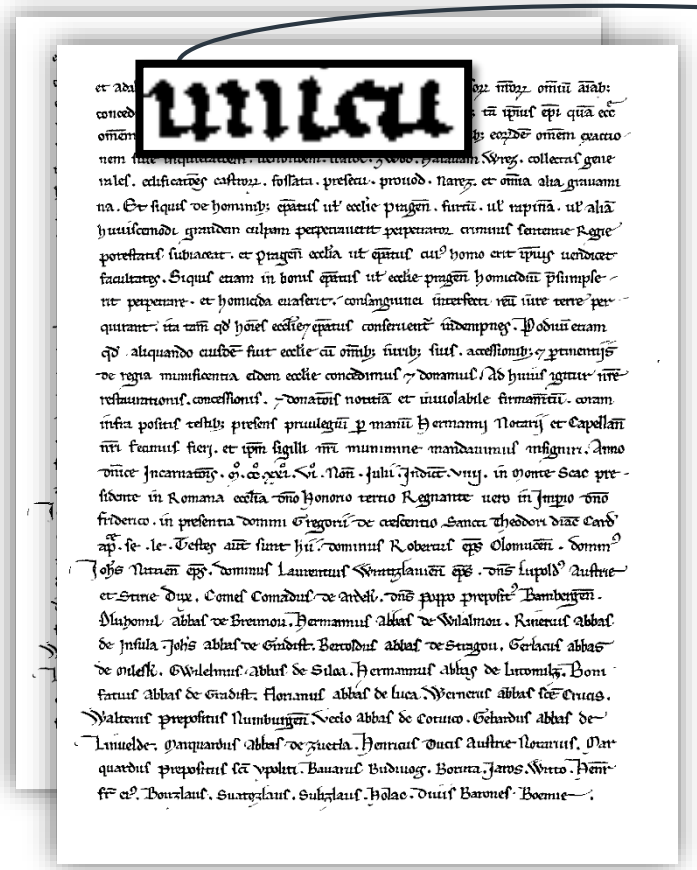


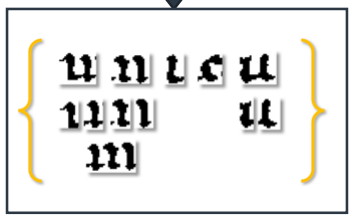
Niente



«asseras»

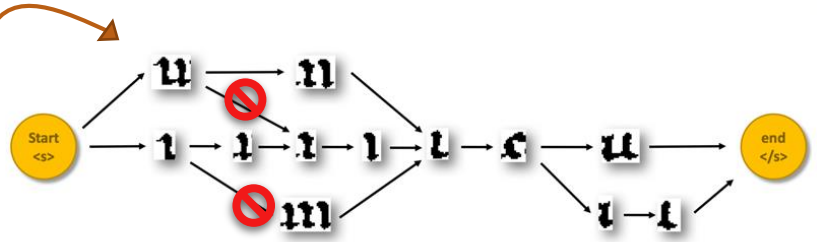
In Codice Ratio





Neural Network OCR

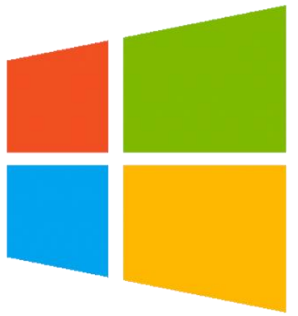
[unicu, unicii, uiicu, …]

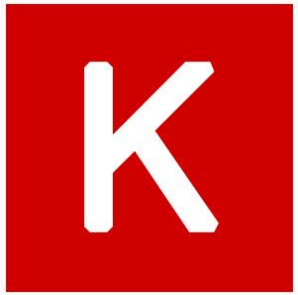


Language Model

«unicu»

L’ambiente



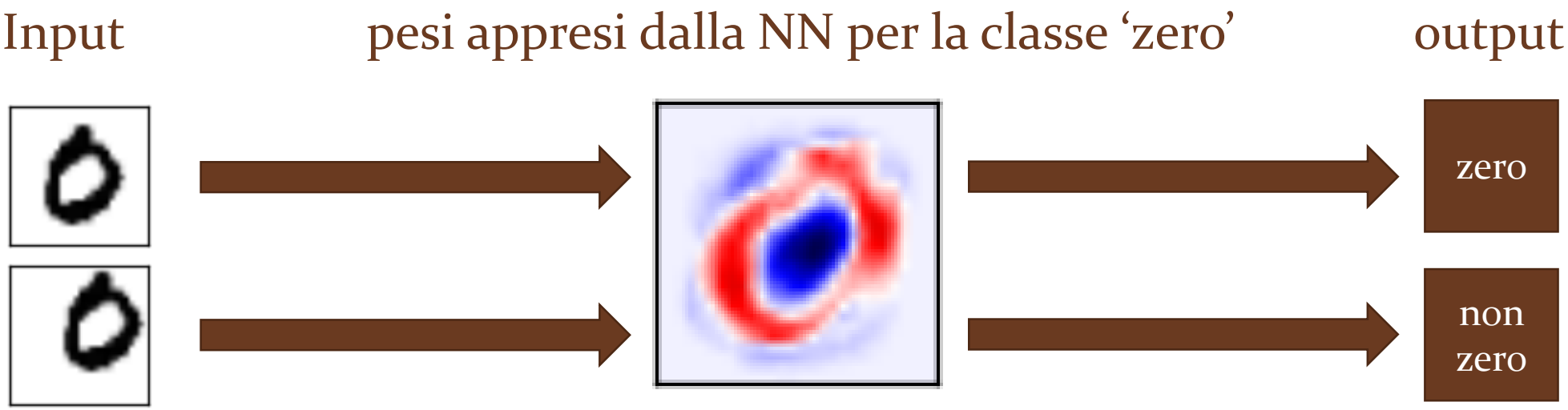


TensorFlow

0.12

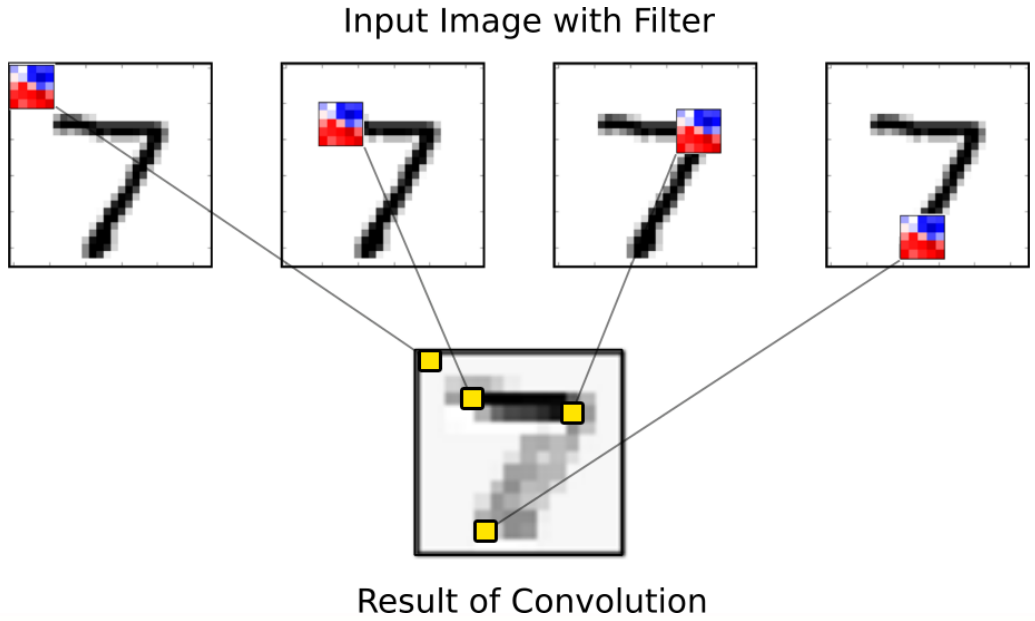
Perché «convolutional»?

- Linear model (senza convoluzione)

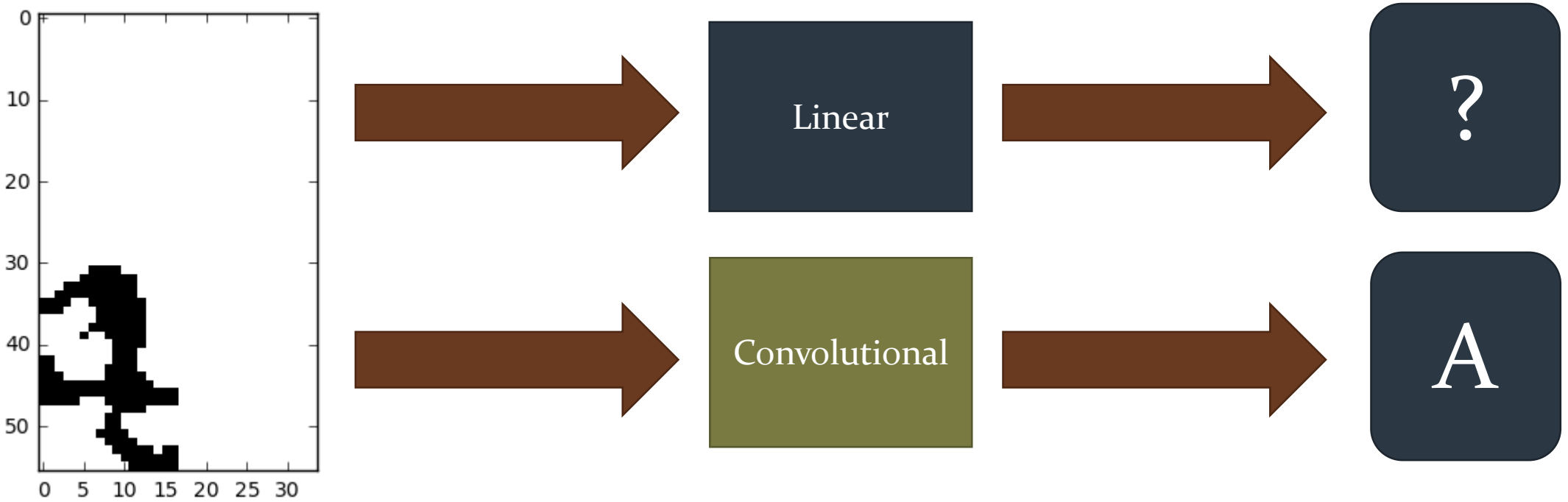


Perché «convolutional»?

- Convoluzione: la rete neurale allena sempre dei pesi, ma stavolta la loro finalità è l'estrazione delle «feature» dell'immagine mediante convoluzione. Una volta estratte le feature servono a capire il contenuto dell'immagine.



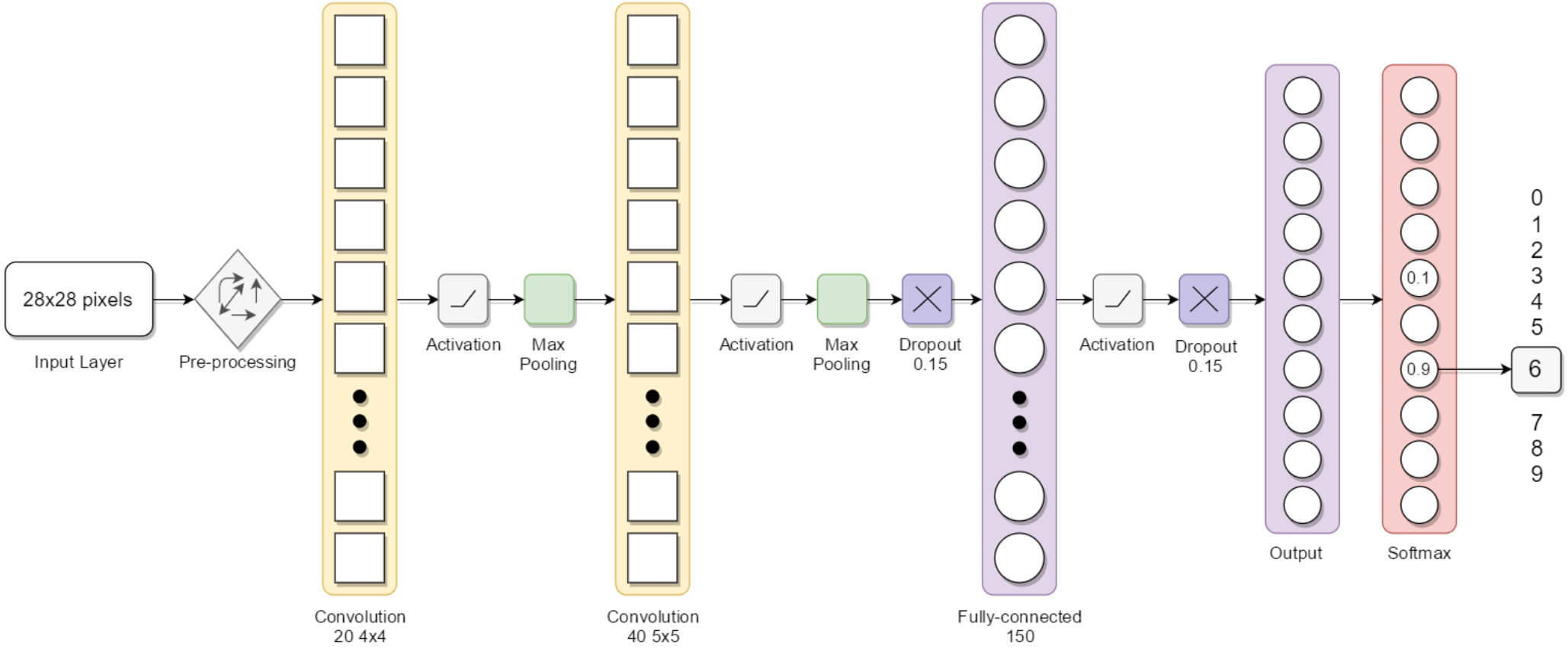
Convolutional vs Linear



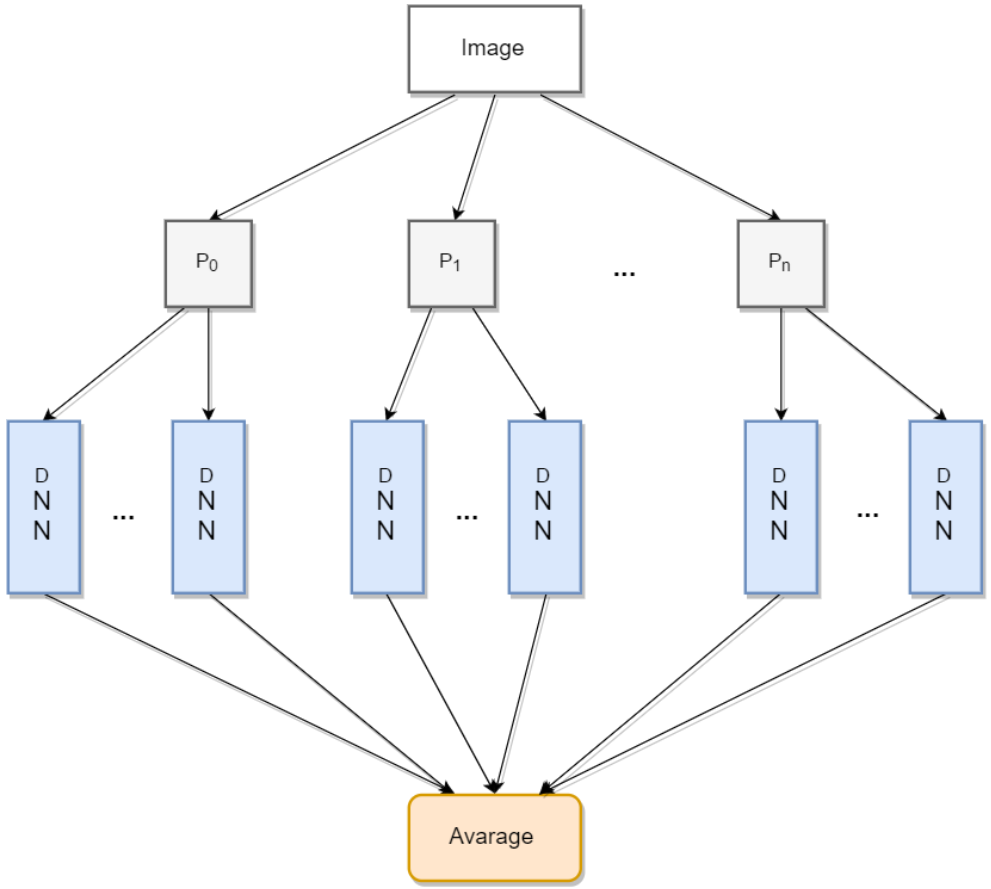
L’architettura

- Per progettare l’architettura abbiamo utilizzato un noto dataset di test: MNIST.
- Su questo dataset l’errore umano è dello 0.2%, mentre il record del computer è un errore dello 0.21% raggiunto nel 2013 con una rete a 35 colonne.
- Abbiamo applicato in scala ridotta le stesse tecniche, integrandole ad altre tecniche moderne, migliorando il risultato su 5 colonne ottenuto nel 2013, con un errore complessivo dello 0.4%.

L'architettura usata per MNIST: singola colonna



L’architettura: multi colonna (ensemble)



Reti realizzate per ICR

- 22 reti binarie, una per ogni lettera attualmente etichettata.
  - Output (per la «a»): «a», «altra lettera o non una lettera»
  - Legacy dalla prima progettazione di ICR
- Rete OCR a 22 classi
  - Output: «a», «b», «c», …
- Rete segmentatrice binaria
  - Output: «lettera», «non una lettera»

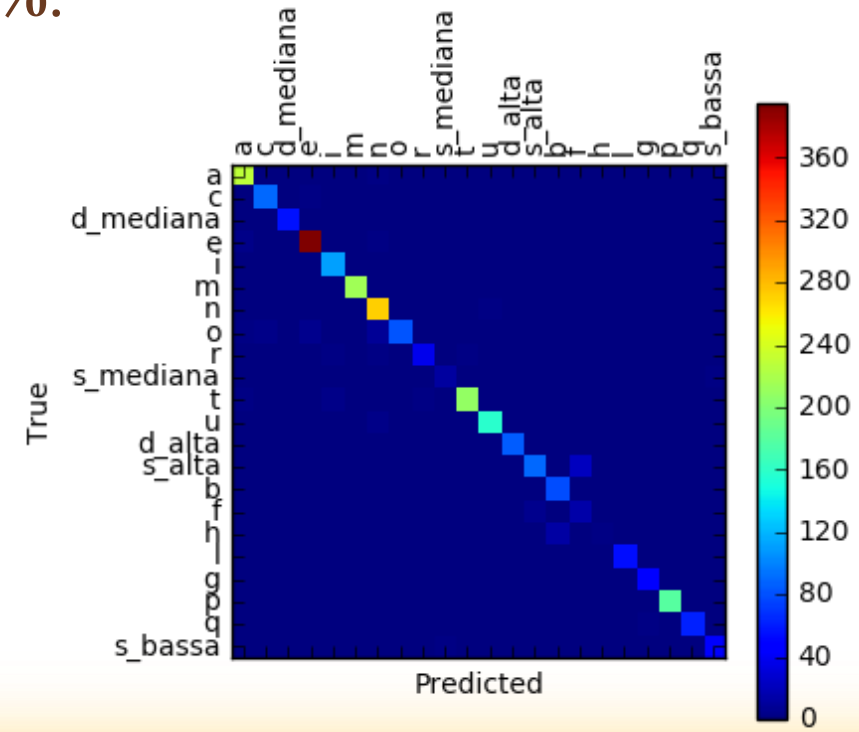
Risultati dell'addestramento: 22 reti binarie

- Quasi ogni lettera ha raggiunto un errore tollerabile inferiore al 5% eccetto:

Carattere	Ratio pos:neg 1:1	Ratio pos:neg 1:2	Esito
i	17,9%	17,5%	-
m	9,4%	10.3%	+
n	11,2%	12.9%	+
u	15.5%	15.8%	+
s_mediana	3.3%	6.6%	+
h	19,4%	13.8%	-
f	5%	7.5%	+
media	**7.2%**	**7.1%**	-

Risultati dell'addestramento: OCR a 22 classi

- In generale la rete OCR si è comportata meglio, dal momento che ha un solo task e può utilizzare le feature estratte dalle altre lettere per distinguere quella in esame «per esclusione». L'errore sul test set è un tollerabile **4,6%**.



Risultati dell'addestramento: rete segmentatrice

- Anche la rete segmentatrice si è dimostrata buona per il task in esame, con un errore sul test set del 6,2% equamente diviso tra falsi positivi e falsi negativi.
- Gli errori principali sono dovuti a similitudine di tagli sbagliati di «m», «n», «u» ed «r» con tagli corretti della «i», oltre che con tagli errati della «m» con tagli corretti della «n».
- Ciò comporta che alcune «i» ed «n» potrebbero dare dei falsi negativi per via dell'overfitting, soprattutto per le «i» risultanti da tagli di legature.

Pipeline per classificazione

Pipeline 1

- 22 reti binarie

Pipeline 2

- Rete segmentatrice
- Rete OCR

Pipeline 3

- 22 reti binarie (solo per segmentare)
- Rete OCR

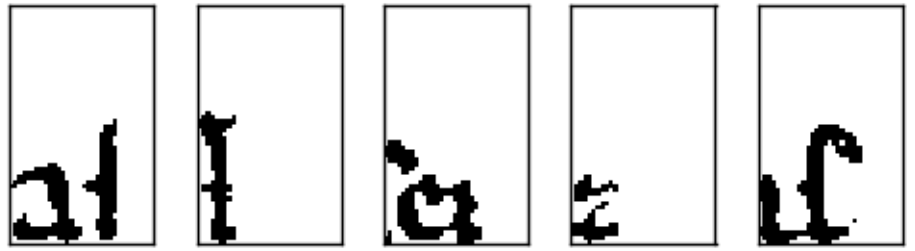
Pipeline 4

- Rete segmentatrice
- 22 reti binarie

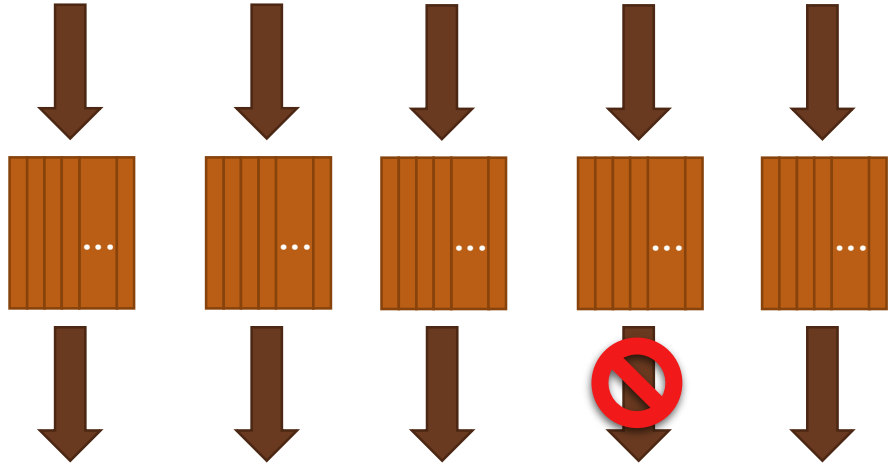
Abbiamo studiato il comportamento di 4 pipeline sulle parole di una pagina, tagliate in tutti i modi possibili (nei minimi locali).

Esempio di pipeline (pipeline 1)

Input (tagli errati)



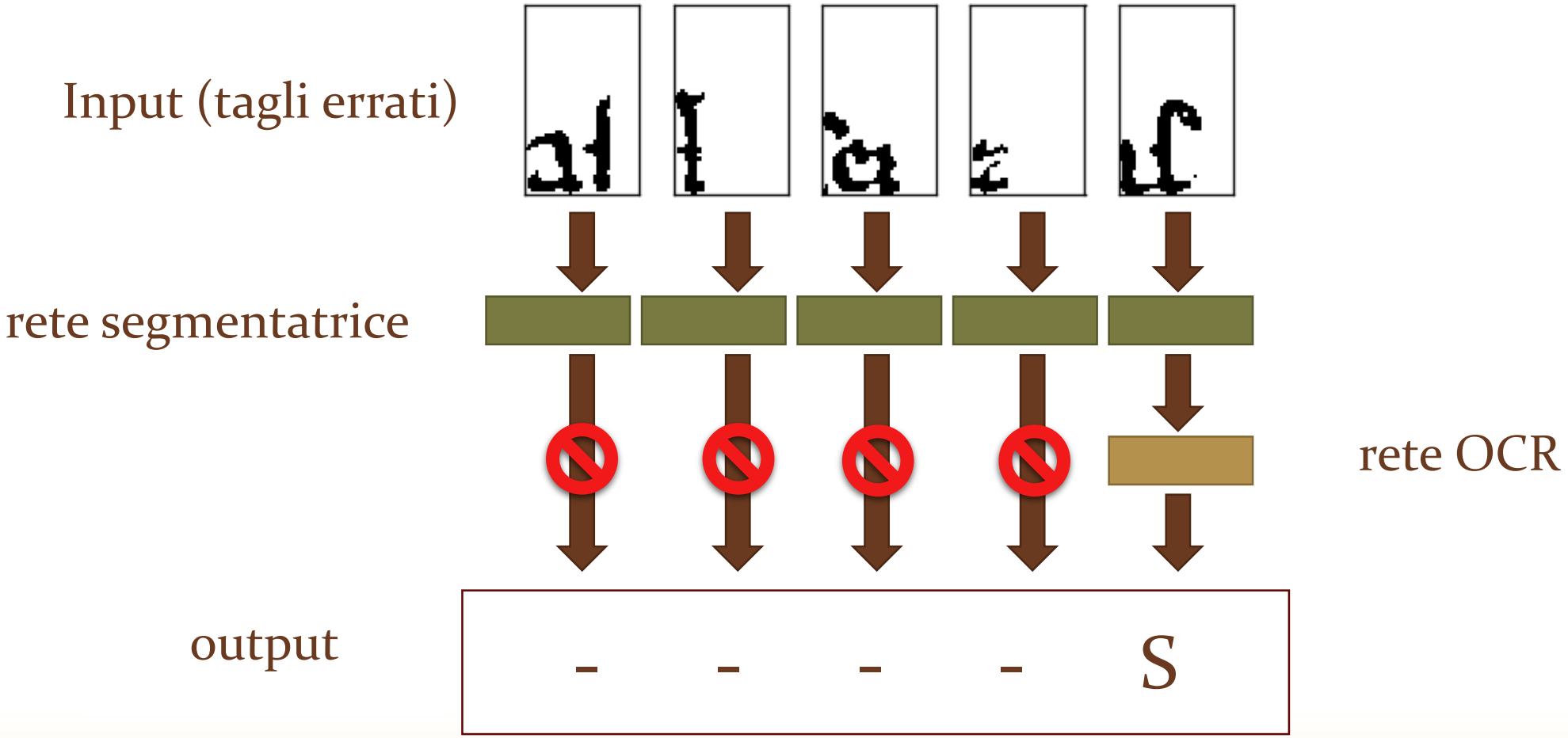
22 reti binarie



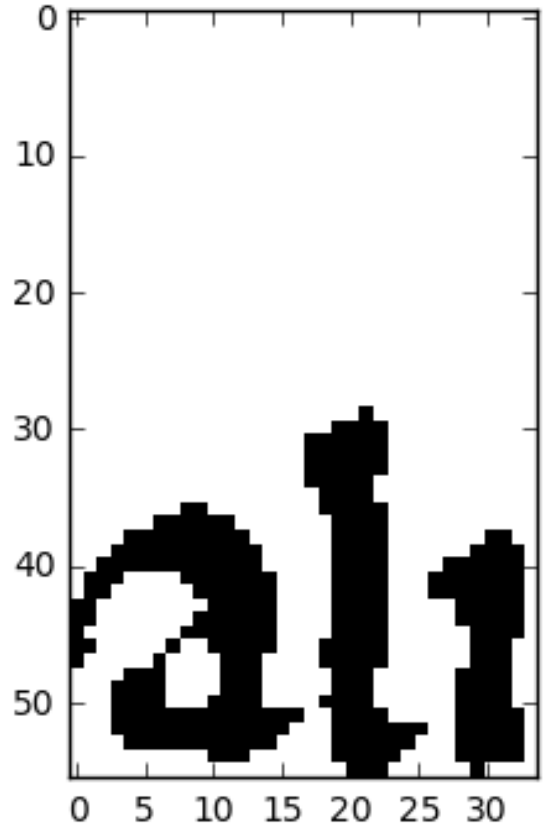
output

S	L	S	-	S

Esempio di pipeline (pipeline 2)



Esempio di output delle 4 pipeline (taglio errato)



Pipeline 1

- «s_mediana» 93%

Pipeline 2

- Bad cut 100%

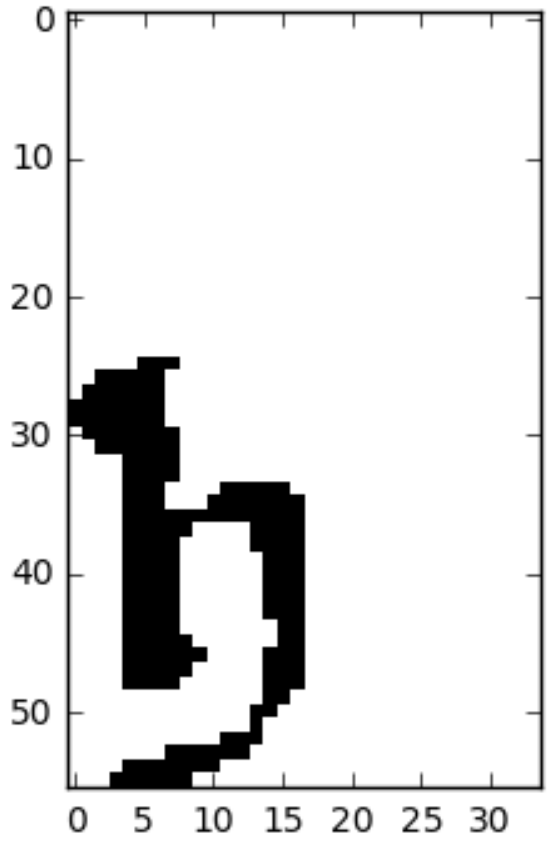
Pipeline 3

- Good cut 93% + «b» 50%

Pipeline 4

- Bad cut 100%

Esempio di output delle 4 pipeline (taglio corretto)



Pipeline 1

- «h» 98%

Pipeline 2

- «h» 99%

Pipeline 3

- Good cut 98% + «h» 99%

Pipeline 4

- «h» 98%

Risultati

- Abbiamo testato le 4 pipeline su un foglio con 13500 tagli di lettere.
- In generale le pipeline 1 e 3 classificano allo stesso modo i tagli con molti falsi positivi, con la pipeline 3 ad avere il miglior ranking.
- Le pipeline 2 e 4 hanno pochissimi falsi positivi ma sporadici falsi negativi, e fra loro la pipeline 2 ha il miglior ranking.
- In generale la pipeline 2 si è rivelata essere la più efficace e la più efficiente, da 10 a 50 volte più veloce a classificare delle pipeline 1 e 3.
- In caso di rari falsi negativi può intervenire il language model o si può utilizzare la pipeline 3 qualora la 2 non desse risultati per avere più tolleranza.

Grazie per l'attenzione

https://github.com/Kidel/In-Codice-Ratio-OCR-with-CNN