Report Progetto

Michele Cafagna, Ruggiero Santo
mikele.c@outlook.it, ruggiero.santo@gmail.com

Team: Pest-e-Fasoul

ML 654AA 2017/2018
Date: 25*/01/2018*
Tipo di progetto: **A**

ABSTRACT

In questo progetto è stata implementata una rete neurale personalizzabile, ed è stata testata confrontando gli effetti di hyperparametri, funzioni di attivazione e inizializzazione. Ci si concentrerà maggiormente sul confronto tra diversi ottimizzatori implementati e calibrati utilizzando Grid-Search, Random-Search e Cross-Validation. Lo scopo è quindi quello di proporre la migliore soluzione verificando e confrontando alcune delle tecniche presenti in letteratura.

1. INTRODUZIONE

Nella seguente relazione, verranno mostrati i risultati relativi alle sperimentazioni condotte sfruttando una rete neurale con un solo hidden layer. Lo scopo della sperimentazione è stabilire l'effettivo ruolo di funzioni di attivazione, funzioni di inizializzazione, ottimizzatori, in combinazione con i comuni hyperparametri. Per effettuare questi confronti si farà uso di Grid-Search e Random-Serch in combinazione con Cross-Validation, in modo da ridurre la varianza generata dal dataset e rendere confrontabili i risultati.

2. METODO

L'intero progetto è stato implementato in Matlab e non sono state utilizzate librerie o tools esterni. La rete è stata progettata tenendo conto di due fattori fondamentali: modularità e generalità; a questo scopo si sono sfruttate caratterisitche Object Orientend e funzionali del linguaggio per implementare la rete e i singoli livelli separatamente. Ogni livello è responsabile delle proprie unità, e della loro inizializzazione, in oltre tiene traccia del suo gradiente, della sua matrice dei pesi, del suo input e del suo output gestendo la propria fase di forward e il bias; queste features permetto di mantenere l'indipendenza funzionale dalla rete in cui sono impiegate e il possibile riutilizzo in altre reti. La rete è in grado di gestire le fasi di training, validation e test separatamente e contemporaneamente, in base ai parametri passatigli; sfrutta il potente sistema di gestione delle matrici fornito da Matlab, per effettuare tutte le operazioni in forma matriciale velocizzando di fatto l'intera fase di training che è anche la più computazionalmente costosa. Queste scelte tecniche, hanno permesso di rendere l'intero progetto modulare e con un certo grado di generalità; puntando a produrre una sorta di libreria o più limitatamente un insieme di tools, espandibili con moduli. Sono state implementate anche *utilities*, per effettuare operazioni di preprocessing, calcolo delle metriche, plotting, manipolazione del dataset, e model selection (Random-Search, Grid-Search e Cross-Validation).

Si sono svolte sperimentazioni, sugli effetti delle funzioni di inizializzazione dei pesi, delle funzioni di attivazione (per la lista completa si consulti l'**Appendice C**), e diversi ottimizzatori, in combinazione con i classici hyperparametri. Per la strategia di scelta e validazione del modello si è deciso di sfruttare una ricerca casuale, per poter esplorare lo spazio dei parametri e poter restringere il raggio di ricerca, successivamente è stata utilizzata la Grid-Search per avere una maggiore e più accurata ricerca all'interno dello spazio di possibilità, in ognuna di queste fasi si è fatto di uso di Cross-Validation, per ridurre al minimo la varianza prodotta dal dataset e favorire una migliore valutazione del modello. Sapendo che i dati provengono da sensori, abbiamo supposto che sia presente del rumore e perciò abbiamo ritenuto adatto l'utilizzo del Cross-Validation, inoltre non sapendo la

natura e l'oggetto della misurazione effettuata abbiamo ritenuto opportuno normalizzare i dati. Nella denormalizzazione dei dati di output, si sono utilizzati i range dei target fornitici nel traning set, assumendo che anche i dati del test set abbiano la stessa distribuzione.

3.SPERIMANTAZIONE

Come già anticipato, sono state svolte sperimentazioni con diversi ottimizzatori, nello specifico oltre al classico SGD, sono stati implementati AdaGrad e AdaDelta, quest'ultimo si propone essere una versione migliorata del primo. Sommariamente possiamo dire che AdaGrad adatta il learning rate ai tipi di feature processate, informalmente fornendo un basso learning rate a feature frequenti e un più alto learning rate a features meno frequenti [1] accumulando i vecchi gradienti. AdaDelta, invece usa lo stesso principio di AdaGrad, ma utilizzando una finestra di accumulazione dei gradienti, (media mobile), in modo da fornire un learninig rate adattato localmente, evitando un indefinito decadimento del learning rate, come accade per AdaGrad [2]. In AdaDelta, vengono introdotti due nuovi parametri ρ, una costante di decadimento, utilizzata nel calcolo della media mobile dei gradienti ed ε, anch'essa costante (in genere molto piccola), per mantenere la stabilità numerica in caso di gradienti prossimi allo 0. Nella nostra implementazione è stata tenuta fissa a 10^{-8} come implementato nella versione di Keras [3]. Queste tecniche si propongono utili nel riconoscimento vocale e delle immagini, risolvendo il problema della sparsità dei dati. Proprio per quest'ultima caratteristica si è deciso di confrontarli con SGD, considerandoli anche per l'impiego nel modello finale.

3.1 RISULTATI DEI MONK'S

Di seguito verranno mostrati i modelli utilizzati per i Monk's Tasks e i relativi risultati, inoltre saranno menzionati solo le funzioni di attivazione e di inizializzazione dei pesi dell'hidden layer, ma sono da intendersi le stesse per l'output layer, per questi task è stato utilizzato unicamente SGD come tecnica di ottimizzazione.

MONK1						
UNITA'	F. ATT.	F. INIZIALIZZAZIONE	η	α	λ	DIM. BATCH
3	Sigmoid	Distribuzione Normale [-0.5, 0.5]	0.7	0.8	0.0	25

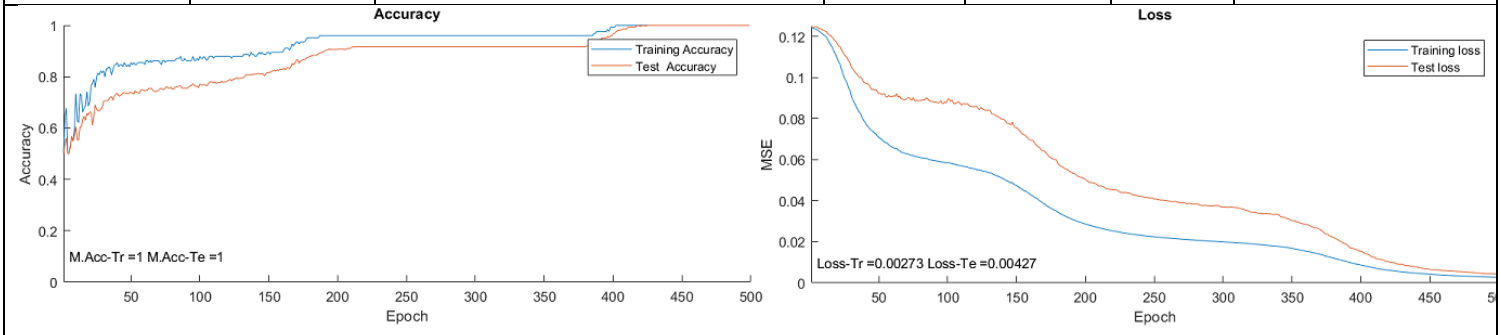


Tabella 1 Modello MONK1

MONK2						
UNITA'	F. ATT.	F. INIZIALIZZAZIONE	η	α	λ	DIM. BATCH
2	Sigmoid	Distribuzione Normale [-0.5, 0.5]	0.7	0.84	0	25

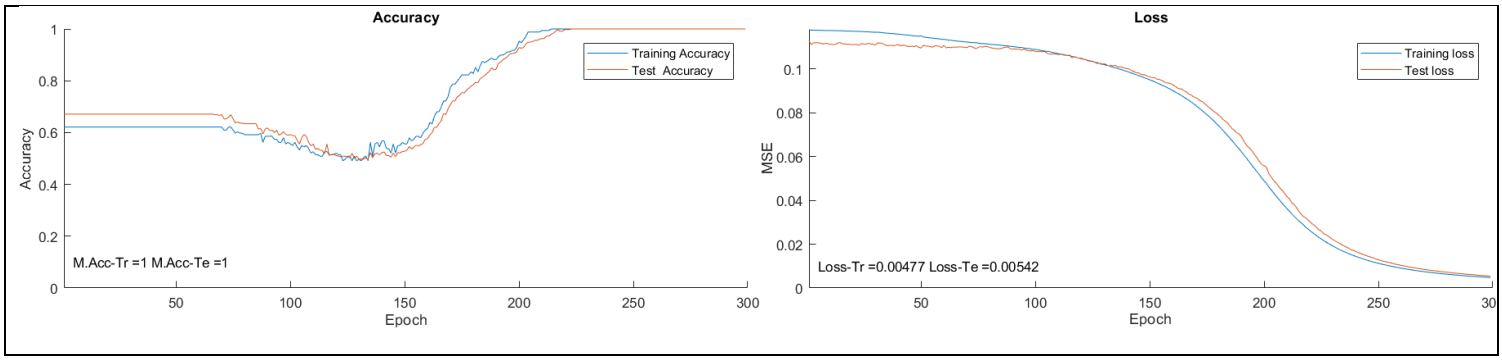


Tabella 2 Modello MONK2

MONK3						
UNITA'	F. ATT.	F. INIZIALIZZAZIONE	η	α	λ	DIM. BATCH
2	Sigmoid	Distribuzione Normale [-0.5, 0.5]	0.5	0.8	0.0001	25

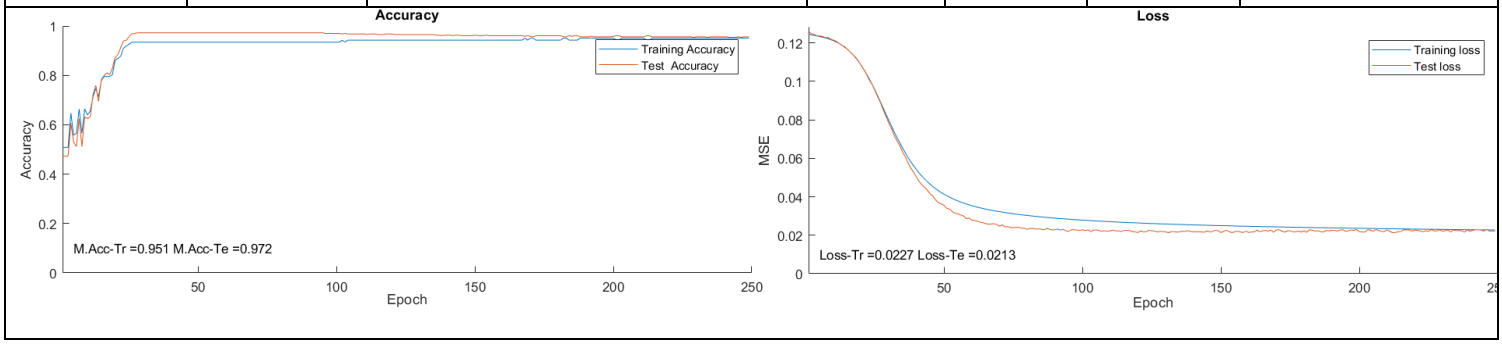


Tabella 3 Modello MONK3

TASK	MSE (TR/TS)	ACCURACY (TR/TS)
MONK 1	.00273 / .00427	100 % / 100 %
MONK 2	.00477 / .00542	100 % / 100 %
MONK3	.0227 / .0213	95 % / 97 %

Tabella 4 Risultati dei MONKS

3.2 CUP RESULTS

SELEZIONE DEL MODELLO

Creazione del Testset e Preprocessing

Per prima cosa si sono estratti casualmente il 20% dei record dal dataset, per poter essere utilizzati come testset nella valutazione finale del modello, i dati sono stati normalizzati utilizzando il *Feature scaling* anche noto come *min-max*, che trasforma i dati in un range che va da 0 a 1. Naturalmente tutti gli output sono stati denormalizzati quindi i grafici dell'accuratezza (MEE) sono in scala originale**.**

Fase di Screening

È stata svolta una fase di screening, verificando qualitativamente l'andamento dei grafici, al variare degli hyperparametri. A seguito di questa fase si è visto che con SGD:

- L'apprendimento non presenta particolari oscillazioni o instabilità,
- Il momentum, sembra influenzare positivamente la convergenza,
- La regolarizzazione sembra non intaccare i grafici.

In AdaGrad:

- Si necessita di un basso learning rate iniziale (< 0.1)
- Il momentum influenza positivamente la convergenza
- Si ottengono convergenze migliori in minibatch, piùttosto che in batch

Mentre in AdaDelta:

- Si ottiene una buona convergenza con un learning rate ancora più basso rispetto all'AdaGrad (<0.01)
- Ci sono molte oscillazioni, compensabili in parte con momentum e in parte con regolarizzazione

Da questa prima fase di screening, si è visto che generalmente AdaGrad e AdaDelta ottengono livelli di accuratezza leggermente migliori di SGD, mentre per quanto riguarda gli altri hyperparametri, si sono ottenute solo informazioni sommarie, se non altro si è notata una certa stabilità nella convergenza; questi e altri aspetti verrano approfonditi in seguito.

Cross-Validation e scelta del parametro 'k'

Vista la diversità di parametri che si andrà a valutare, si è ritenuto necessario utilizzare un metodo di validazione che rendesse confrontabili i risultati a prescindere dal training set e dal validation set utilizzati, a questo scopo si è scelto di utilizzare il Cross-Validation. Per la scelta del numero di partizioni, si è deciso di valutare la variazione delle prestazioni della rete con la stessa configurazione di parametri[1], al variare di k.

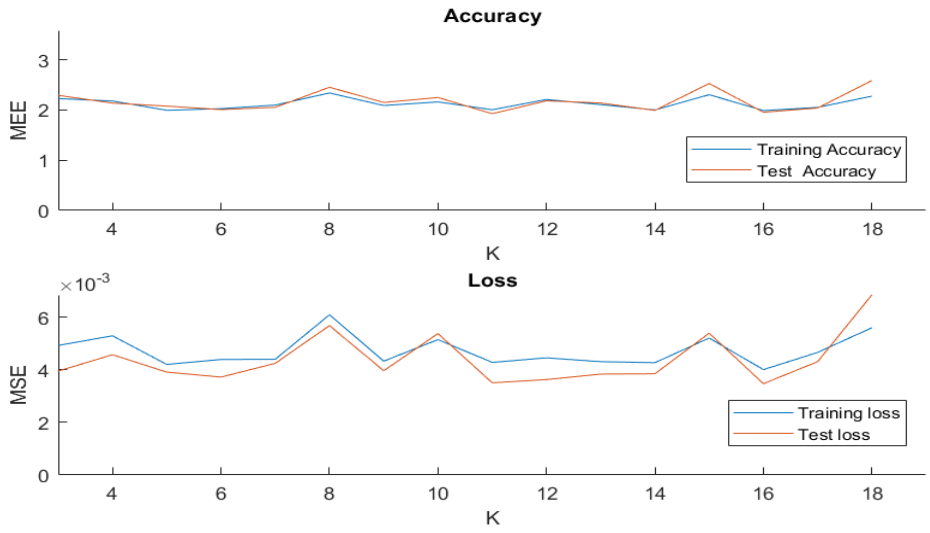


Figura 1 Accuracy e Loss del modello al variare di k da 3 a 20.

L'andamento è oscillatorio e non presenta particolari minimi, il miglior valore si ottiene per k = 11, tuttavia considerando l'andamento totale di entrambi i grafici e il guadagno in termini di accuratezza e loss, si avrebbe un drastico aumento del tempo di esecuzione, perciò si è deciso di considerare le ipotesi 3 e 5 partizioni. Dopo analisi più approfondire, si è visto che l'effettivo guadagno medio per 5 partizioni rispetto a 3, è minimo ($< 10^{-3}$) e il surplus di tempo necessario è $\simeq 2$ s, quindi si è optato per $k = 3$.

[1] 5 unità, SGD, Sigmoid, η: 0.1, α: 0.8, λ: 0, epoche: 1000, minibatch: 500

Random-Search

Dopo la fase esplorativa sono stati definiti i parametri per effettuare la ricerca casuale con Cross-Validation. Qui di seguito sono indicati i range utilizzati con un massimo di 1000 epoche in quanto sperimentalmente si è visto che entro tale limite si ottiene un buon grado di accuratezza del modello:

OTTIMIZZATORE	SGD	AdaGrad	AdaDelta
UNITA'	[2-10] step: 2	[2-8] step: 2	[2-8] step: 2
F. ATTIVAZIONE	Logistic, Tanh, SoftPlus	#	#
F. INIZIALIZZAZIONE	Random, Normal, Fanin	#	#
η	[0.01 - 1] step: 0.1	[0.01 – 0.2] step: 0.01	[0.001 - 0.01] step: 0.001
α	[0 - 0.9] step: 0.1	[0.3- 0.9] step: 0.1	[0.3- 0.9] step: 0.1
λ	$[10^{-10}\text{-}10^{-2}]$ step: 10^2	$[10^{-14}\text{-}10^{-6}]$ step: 10^2	$[10^{-14}\text{-}10^{-6}]$ step: 10^2
DIM. BATCH [2]	[0-500] step:100	#	#
ρ	No	No	[0.7 - 1] step:0.1

Tabella 5 : range dei parametri utilizzati nel Random-Search Cross-Validation con SGD, AdaGrad, AdaDelta

Sono state così generate combinazioni casuali di parametri, con i quali è stato eseguito il training con 3-fold Cross-Validation. Il 3-Cross-Validation, restituisce 3 modelli, perciò si è considerata la media dei 3 modelli e la si è confrontata con gli altri setting generati casualmente, la procedura restituisce di volta in volta 10 modelli. Per SGD la ricerca ha prodotto modelli con comportamenti differenti; si notano oscillazioni con momentum, learning rate elevati, pur raggiungendo ottime accuratezze e banalmente un basso learning rate genera una convergenza molto lenta, di seguito verranno riportate le migliori configurazioni trovate. Abbiamo notato che generalmente con l'utilizzo di AdaGrad, risultati contrastanti e generalmente più scarsi rispetto a SGD e AdaDelta, oltre ad un un'apparente aumento di oscillazioni con la TanH in combinazione con AdaGrad. Stranamente con parametri simili si ottengono comportamenti differenti, probabilmente, questo aspetto sarà chiarito della successiva Grid-Search. Di seguito sono mostrati i migliori modelli trovati con SGD e AdaGrad:

SGD								
UNITA'	F. ATT.	F. INIZ	η	α	λ	DIM. BATCH	(MSE) TR/VS	(MEE) TR/VS
4	TanH	Fanin	0.81	0.3	10^{-10}	300	.0026/.0026	1.46/1.49
8	SoftPlus	Fanin	0.01	0.7	10^{-4}	200	.0032/.0030	1.79/1.78
8	SoftPlus	Normal	0.21	0.5	10^{-8}	400	.0032/.0030	1.79/1.85
ADAGRAD								
8	Logistic	Random	0.16	0.3	10^{-8}	100	.0037/.0034	1.95/1.90
8	Logistic	Normal	0.01	0.7	10^{-10}	400	.0084/.0082	2.77/2.8
4	TanH	Normal	0.15	0.3	10^{-8}	400	.0032/.0030	2.27/1.57

Tabella 6 Migliori modelli Random-Search con SGD e AdaGrad

[2] Nell'implementazione, un *batch-size* = 0 corrisponde a training di tipo *batch* quindi con tutto il training set, mentre con dimensione 1 o *n* si intende rispettivamente *online* o *minibatch*

Molti risultati con AdaDelta soffrono di forti oscillazioni; i test sembrano evidenziare una certa sensibilità da parte di AdaDelta al momentum e alla regolarizzazione (esempi nell'**appendice A**). Di seguito le migliori configurazioni ottenute con AdaDelta:

UNITA'	F. ATT.	F. INIZ	η	α	λ	ρ	DIM. BATCH	(MSE) TR/VS	(MEE) TR/VS
8	Logistic	Normal	0.003	0.8	10^{-6}	0.9	200	.0026/.0026	1.43/1.45
2	Logistic	Fanin	0.008	0.7	10^{-2}	0.9	200	.0029/.0031	1.60/1.68
8	SoftPlus	Normal	0.007	0.9	10^{-4}	0.9	400	.0032/.0029	1.65/1.66

Tabella 7 risultati Random-Search AdaDelta

Grid-Search

In base ai risultati ottenuti i range sono stati ristretti ed è stata eseguita la Grid-Search partendo dai seguenti parametri:

OTTIMIZZATORE	UNITA'	F. ATT.	F. INIZ	η	α	λ	DIM. BATCH
SGD	[4-8] step: 3	SoftPlus, TanH	Normal, Fanin	[0.01-0.8] step: 0.4	[0.3-0.7] step: 0.2	[10^{-8}-10^{-4}] step: 10^2	[200-400] step:200
AdaGrad	[4-8] step: 3	Logistic, TanH	Normal, Random	[0.01-0.16] step: 0.04	[0.3-0.7] step: 0.1	[10^{-8}-10^{-4}] step: 10^2	[100-400] step:200
AdaDelta	[2-8] step: 3	Logistic, SoftPlus	Normal, Fanin	[0.001-0.008] step: 0.002	[0.7-0.9] step: 0.1	[10^{-2}-10^{-6}] step: 10^2	[200-400] step:100

Tabella 8 Parametri utilizzati per la Grid-Search. A seguito dei risultati ottenuti con AdaDelta ρ avrà valore costante 0.9

Rispetto alla ricerca casuale, gli step sono stati ampliati in modo da testare un minor numero di combinazioni. In SGD sono stati generati 128 esempi, si è visto che l'inizializzazione fanin, rispetto a quella con distribuzione normale a parità di configurazione genera più oscillazioni (esempio in **appendice A**), lo stesso fenomeno si è visto con la TanH rispetto alla SoftPlus (esempio in **appendice A.1**) tuttavia la TanH raggiunge accuratezza maggiori, perciò è stata mantenuta in griglia. Come già visto nella fase di screening, il momentum e il la regolarizzazione influenzano la convergenza, entrambi crescendo generano oscillazioni, ma sembra che la regolarizzazione generi una maggiore instabilità (un esempio in **Tabella 10**), a questo proposito sono state generate più griglie per ottenere un miglior affinamento dei parametri, ma non verranno mostrate per questioni di spazio. AdaGrad ottiene buone prestazioni con learning rate più bassi di SGD e le convergenze sono più lineari e morbide senza la gobba iniziale, probabilmente indice di un minimo locale, presente invece con SGD (esempio in **appendice B**). AdaDelta si distingue ottenendo risultati decisamente superiori a SGD e AdaGrad, inoltre sembra oscillare maggiormente all'aumentare del numero di neuroni, con particolari valori di learning rate, tutta via questo fenomeno non è lineare, una dimostrazione è mostrata in **Tabella 11** dove si evince un andamento non lineare del fenomeno. Non considerando le piccole oscillazioni, quasi in tutte le combinazioni testate, la convergenza assume un buon andamento si evince che AdaDelta dipende molto dal learning rate (in genere nell'ordine di 10^{-3}), all'aumentare di η si ottengono accuratezza raggiunte in nessun caso dagli altri due ottimizzatori, tuttavia, diventa terribilmente instabile, si è deciso di preferire la stabilità della convergenza all'accuratezza. Al termine dello studio si sono ottenuti i migliori modelli per ogni ottimizzatore, che verranno valutati nella prossima fase

OTTIMIZZATORE	UNITA'	F. ATT.	F. INIZ	η	α	λ	DIM. BATCH	(MSE) TR/VS	(MEE) TR/VS
SGD	4	Tanh	Normal	0.4	0.2	10^{-8}	300	.00286/.00274	1.53/1.51
AdaGrad	8	Logistic	Random	0.06	0.7	10^{-8}	100	.00319/.00280	1.67/1.68
AdaDelta	8	Logistic	Normal	0.0015	0.85	10^{-6}	100	.00263/.00259	1.43/1.48

Tabella 9 Migliori modelli individuati dalla Grid-Search. I grafici dell'errore e accuratezza dei tre modelli sono in appendice per questione di spazio

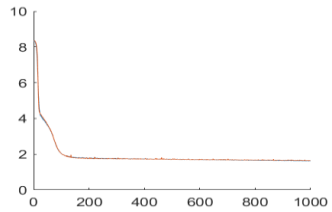
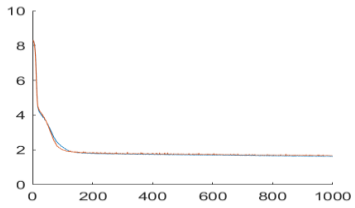
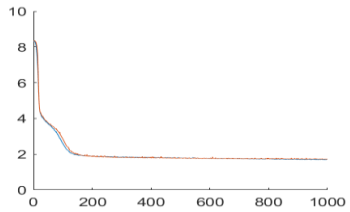
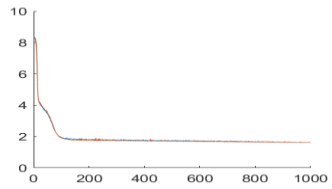
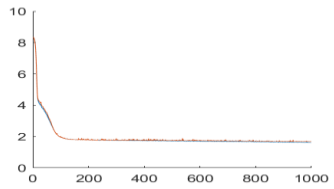
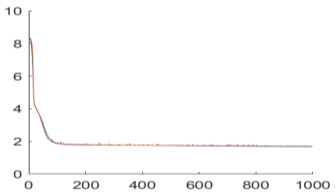
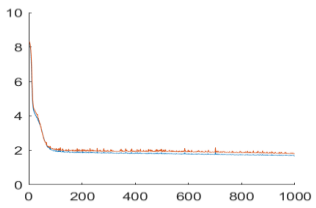
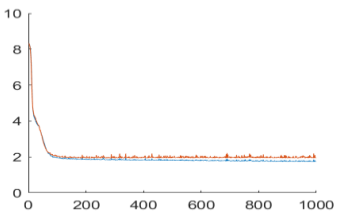
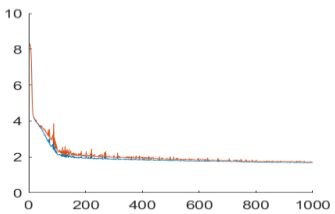
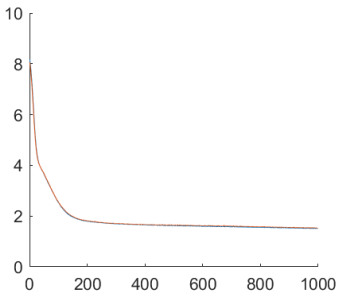
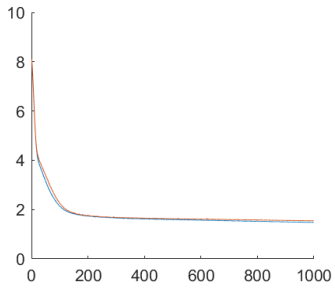
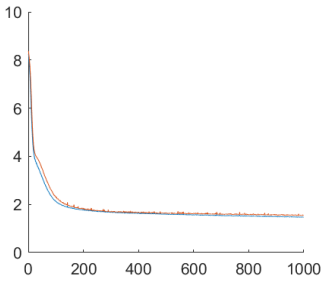
SGD	$\lambda = 10^{-10}$	$\lambda = 10^{-8}$	$\lambda = 10^{-4}$
$\alpha = 0.3$			
$\alpha = 0.4$			
$\alpha = 0.5$			

Tabella 10 Griglia dell'accuratezza (MEE) di un modello con SGD,η =0.4, 5 unità, mini-batch = 100, in cui è visibile come il momentum e la regolarizzazione influiscano la stabilità della convergenza. La legenda è stata eliminata per migliorare la leggibilità, le curve rosse indicano il validation set mentre in blu il training.

AdaDelta	unità = 6	unità = 8	unità = 10
$\eta = 10^{-3}$			

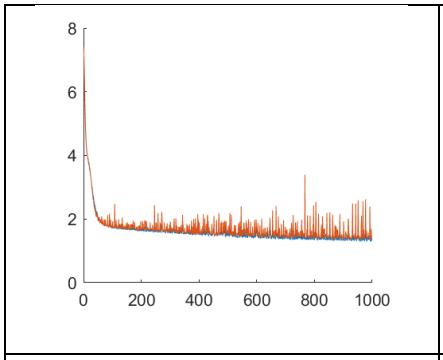
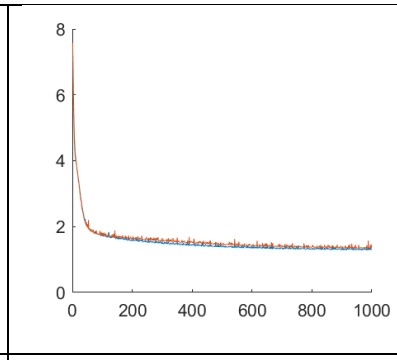
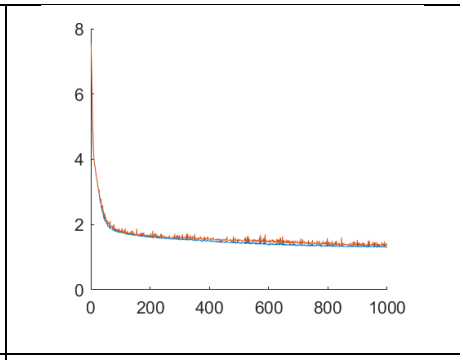
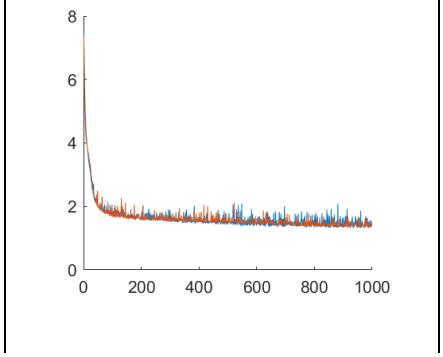
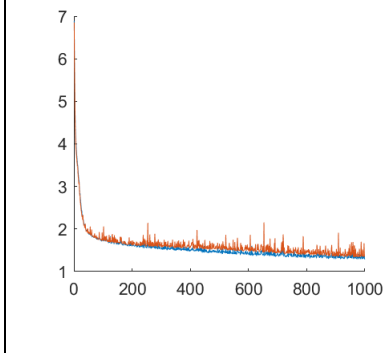
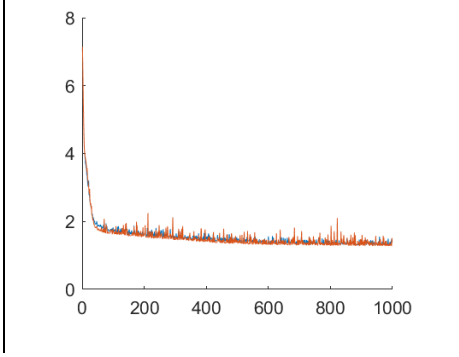
$\eta = 4*10^{-3}$			
$\eta = 8*10^{-3}$			

Tabella 11 Griglia dell'accuratezza in cui si può notare il comportamento anomalo della curva di learning con AdaDelta al variare il numero di neuroni e il learning rate. È visibile come nella prima riga le oscillazioni diminuiscano e nella seconda e nella terza aumentino. La legenda è stata eliminata per migliorare la leggibilità, le curve rosse indicano il validation set mentre in blu il training.

STIMA DEL RISCHIO

Lo scopo dell'esperimento è confrontare i tre ottimizzatori, quindi verranno valutati tutti utilizzando le stesse modalità: ogni modello verrà addestrato con tutto il training set (senza validazione), utilizzando le migliori configurazioni individuate nella fase precedente, e testato con il test set estratto all'inizio del processo, che ricordiamo essere il 20% del dataset totale a nostra disposizione.

MODELLO (OTTIMIZZATORE)	(MSE) TR/TS	(MEE) TR/TS
SGD	.00269 / .00340	1.45 / 1.58
AdaGrad	.00301 / .00371	1.60 / 1.67
AdaDelta	.00251 / .00333	1.34 / 1.46

Tabella 12 Risultati sul trainin e test con i SGD, AdaGrad, AdaDelta

I risultati sul test set sono stati buoni, SGD e AdaDelta hanno avuto un comportamento analogo a quello ottenuto durante la validazione, con accuratezze addirittura maggiori, questo può essere una conferma del fatto che si è riusciti ad evitare l'overfitting, confermando i risultati della validazione incrociata. Per quanto riguarda AdaGrad, invece i risultati son stati leggermente peggiori di quelli ottenuti durante la validazione, non in termini di accuratezza raggiunta (legermente maggiore), ma in ternini di andamento della curva. Si sono avute oscillazioni; discretamente maggiori a quelle ottenute in validazione. Probabilmente il Cross-Validation in questo senso ha mascherato il reale andamento della curva mediando fra tre modelli, smussandola, creando l'illusione che fosse lineare. Questa possibilità non era stata presa in considerazione nella scelta della tecnica di validazione, tuttavia, a giudicare dai risultati ottenuti con gli altri due modelli, possiamo affermare se non altro che il test ha confermato quanto ottenuto in validazione.

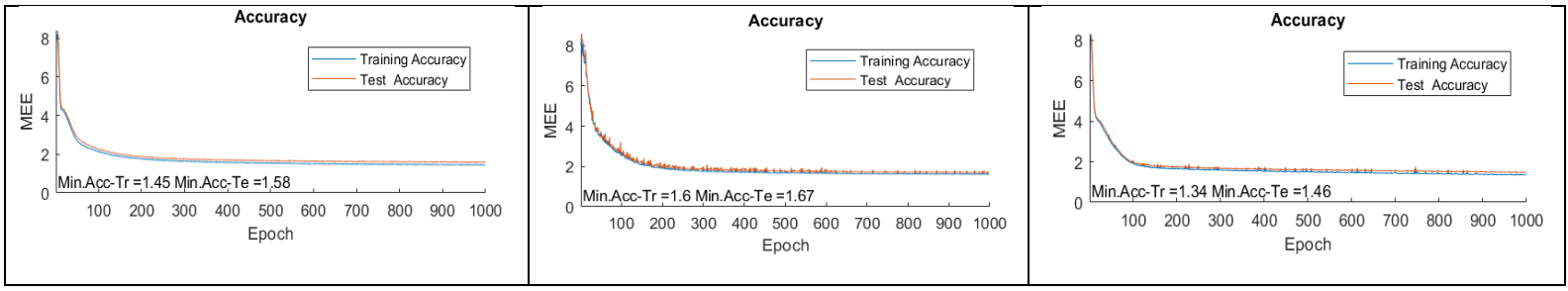


Figura 2 Learning curve dei tre modelli durante la stima del rischio. A sinistra modello con SGD, al centro AdaGrad, a destra Adadelta, è visibile l'instabilità di AdaGrad rispetto agli altri due.

In base hai risultati ottenuti è stato scelto il modello ottimizzato con AdaDelta come modello finale, in quanto anche se presenta delle leggere oscillazioni, ha una buona curva di learning e ha risposto positivamente sia nella validazione sia nel test, (addirittura migliorando in quest'ultima fase la sua accuratezza). La valutazione è stata fatta tenendo conto anche delle oscillazioni che affliggono la learning curve di AdaDelta la forma della curva e la velocità di convergenza; l'entità delle oscillazioni, è minima e riteniamo che l'accuratezza ottenuta, e la velocità di convergenza giustifichino questa scelta.

MODELLO FINALE

Come modello finale per il blind test è stato utilizzato quello ottimizzato addestrato con AdaDelta, la cui configurazione è la seguente, con i seguenti punteggi in test e in valitation, come criterio di stop è stata utilizzata la variazione della loss con tolleranza inferiore a 10^{-10} e limite massimo di 1000 epoche.

UNIT A'	F. ATT.	F. INIZ	η	α	Λ	ρ	DIM. BATCH	(MEE) TR/TV(CV)	(MEE) TR/TS
8	Logistic	Normal	0.0015	0.85	10^{-6}	0.9	100	1.43/1.48	1.34 / 1.46

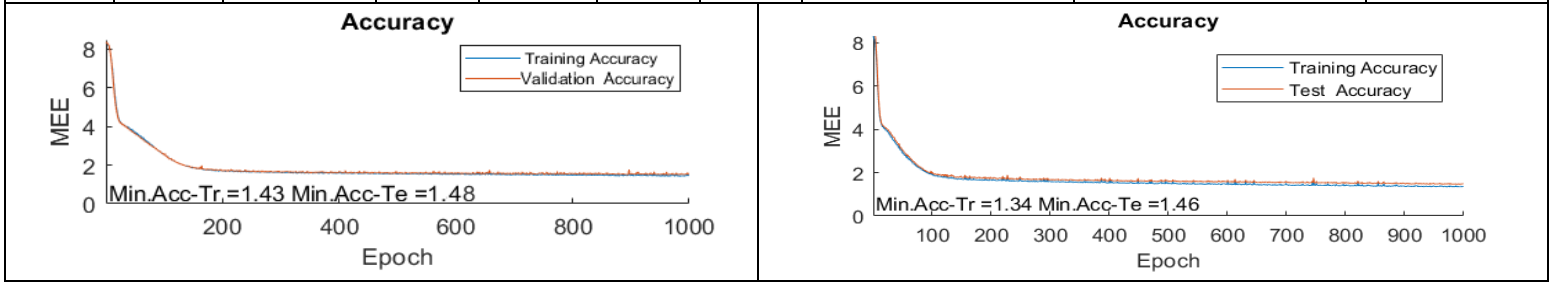


Tabella 13 Modello finale utilizzato nel blind test, con i punteggi ottenti nel 3-fold cross validation (a sinistra) e nel test (a destra)

Per il blind test il modello è stato addestrato con tutto il dataset e sono state proddotte le previsioni per il test set fornitoci.

Prestazioni e tempo di esecuzione

In questa sezione verranno brevemente mostrati i dati relativi al tempo di esecuzione e alle risorse hardware utilizzate nell'analisi. Si ricorda che questi tempi sono relativi al training effettuato sull' 80% del dataset fornitoci, in quanto il restante 20% è stato utilizzato per la stima del modello.

HARDWARE	TRAINING SET	3-FOLD CROSS-VALIDATION
Processore: Intel core i5-8250 1.6GHz	3.9 s	8.9 s

Ram: 8 GB DDR4 2.1 MHz Schede grafiche: Intel HD Graphics 620, GeForce MX 150		

Tabella 14 Tempo di esecuzione e hardware utilizzati per training con cross validation e trainig set del modello finale. Il training set

Non sono stati inseriti i tempi relativi alla ricerca su griglia e casuale, in quanto dipendono dal numero di configurazioni testate e possono essere facilmente stimate utilizzando i tempi utilizzando il tempo necessario per il Cross-Validation.

4. CONCLUSIONI

Nel report sono state inserite le informazioni più rilevanti relative ai test, sono stati fatti test anche con dati denormalizzati, e altre funzioni di attivazione. Nell'implementazione abbiamo cercato di rendere il simulatore, il più generale possibile, questa scelta ci ha permesso di testare molte varianti e architetture diverse, prima di intraprende la reale fase di sperimentazione. La sperimentazione, è stata svolta principalmente intorno agli ottimizzatori implementati, fornendo anche diverse funzioni di attivazione e di inizializzazione, allo scopo di ampliare il range di configurazioni possibili. In base ai risultati ottenuti possiamo confermare che AdaDelta ha prestazioni superiori ad AdGrad come affermato in [2], almeno per quanto concerne la nostra sperimentazione. Tuttavia ci saremmo aspettati con AdaGrad di avere risultati superiori di quelli ottenuti con SGD. Tra i tre ottimizzatori, SGD, risulta il più stabile e meno sensibile agli altri hyperparametri; AdaDelta, risulta molto più sensibile al learning rate, infatti in molte implementazioni come in Keras [3], viene consigliato di mantenere i valori standard di η e ρ. Sapendo che i pattern fornitici, provenivano da sensori, abbiamo assunto la presenza di rumore a abbiamo utilizzato il Cross-Validation per la selezione valdazione del modello. In base ai risultati ottenuti questa tecnica si è rivelata efficace anche se è comunque stata un'arma a doppio taglio in quanto ha mascherato la reale learning curve di AdaGrad, mediando i risultati dei fold e ottenendo plot più smussati di quanto siano realmente almeno in questo caso. In conclusione AdaDelta, è risultato il modello migliore ed è stato utilizzato come modello finale; è stato riaddestrato su tutto il dataset e sono state prodotte le previsioni che sono state denormalizzate nelle modalità precedentemente descritte al termine del paragrafo "METODO", facendo le dovute assunzioni.

ACKNOWLEDGEMENTS

I/we agree to the disclosure and publication of my name, and of the results with preliminary and final ranking **.**

REFERENCES

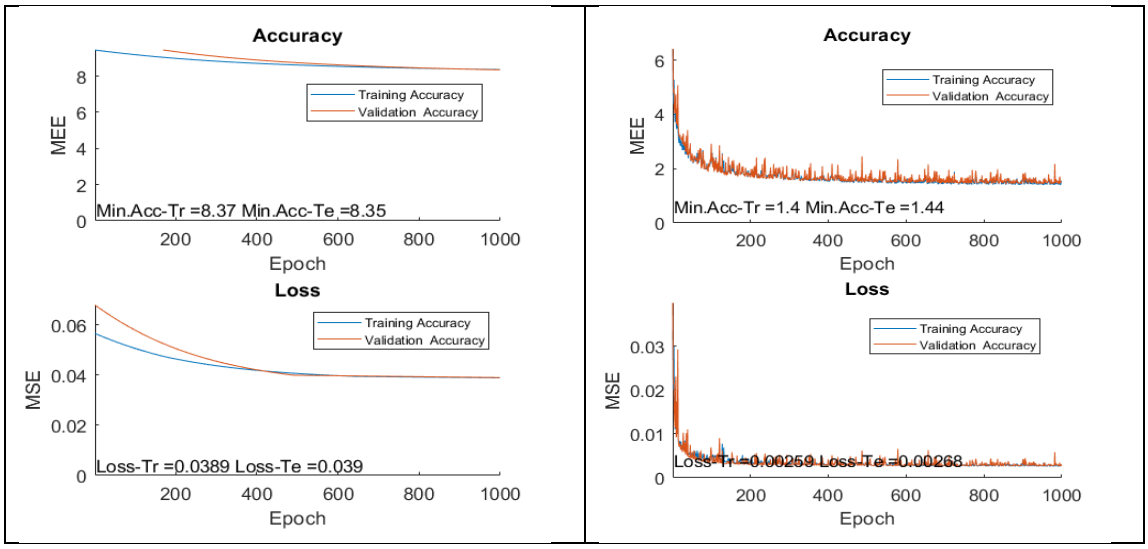
[1] Duchi J., Hazan E., Singer Y., "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization"

[2] Zeiler M.D, "ADADELTA: AN ADAPTIVE LEARNING RATE METHOD", Google Inc., New York university, USA

[3] "AdaDelta optimizer" disponibile su "https://keras.io/optimizers/"

[4] "https://keras.io/initializers/"

APPENDICE A


Accuracy
MEE
Training Accuracy
Validation Accuracy
Min.Acc-Tr =8.37 Min.Acc-Te =8.35
Epoch
Loss
MSE
Loss-Tr =0.0389 Loss-Te =0.039
Accuracy
Min.Acc-Tr =1.4 Min.Acc-Te =1.44
Loss
Loss-Tr =0.00259 Loss-Te =0.00268

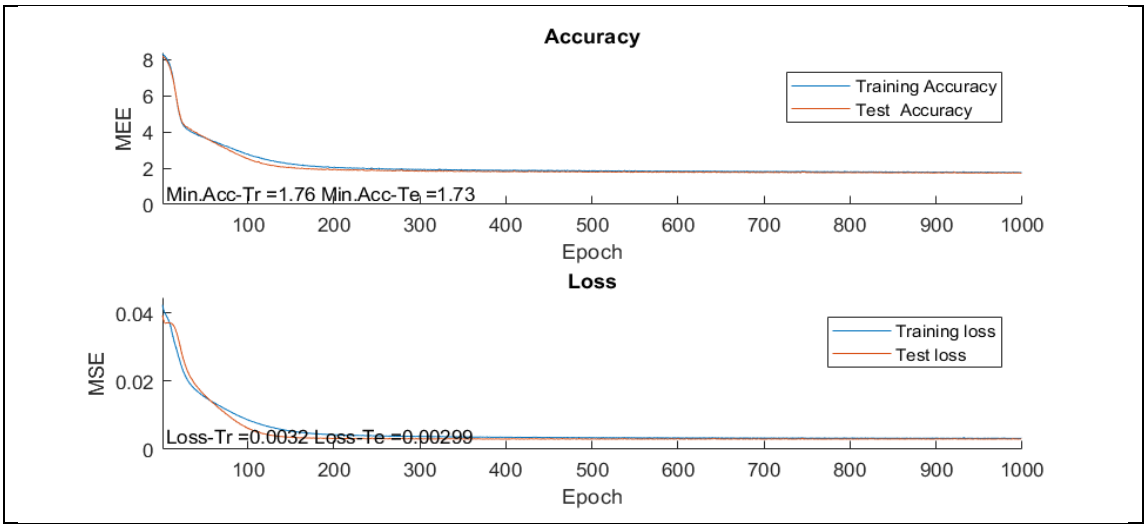


Figura 3 Esempi di risultati ottenuti nella Random-Search con SGD, in alto a sinistra l'esempio di bassa convergenza, in alto a destra l'esempio di buona convergenza ma con molte oscillazioni, in basso l'esempio di una buona convergenza senza oscillazioni.

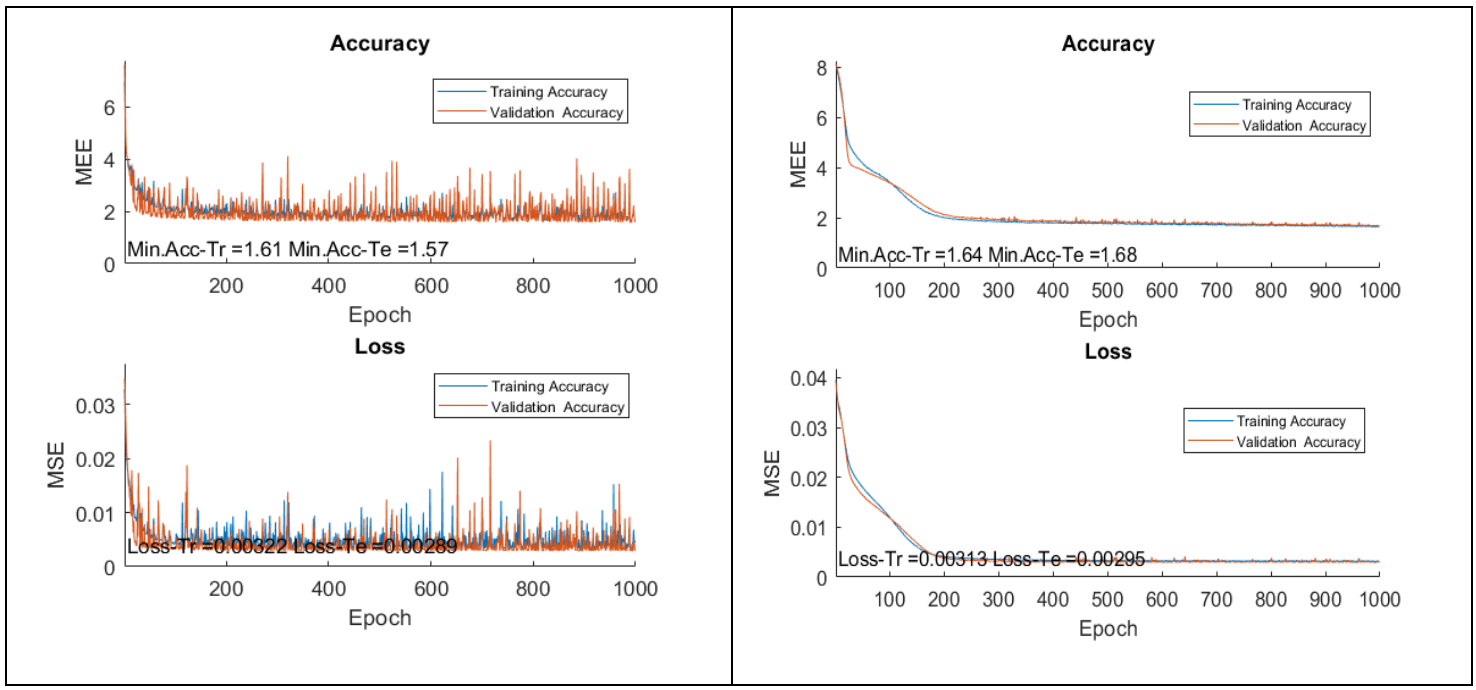


Figura 4 Random-Search due modelli addestrati con la stessa configurazione che utilizzano AdaDelta, sono evidenti le oscillazioni nel modello a sinistra. A sinistra è stato utilizzato un $\lambda = 10^{-6}$ e $\alpha = 0.8$, mentre a destra $\lambda = 10^{-12}$ $\alpha = 0.9$.

APPENDICE A.1

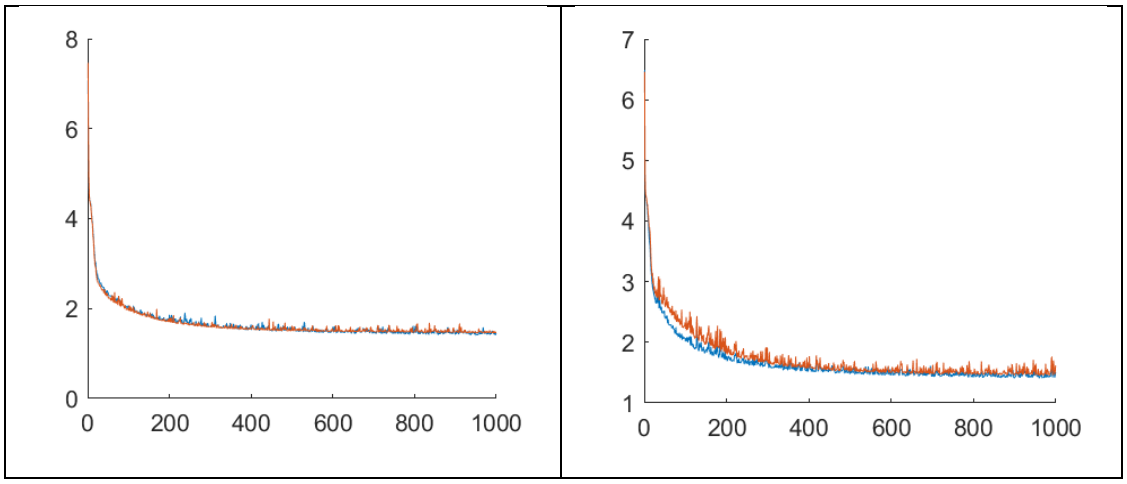


Figura 5 A sinistra inizializzazione con distribuzione normale, a destra inizializzazione con fanin, in SGD. Sono visibili le forti oscillazioni prodotte con fanin, tuttavia sembra che con fanin, l'accuratezza migliori. Questo fenomeno è stato riscontrato soprattutto con modelli ottimizzati con SGD

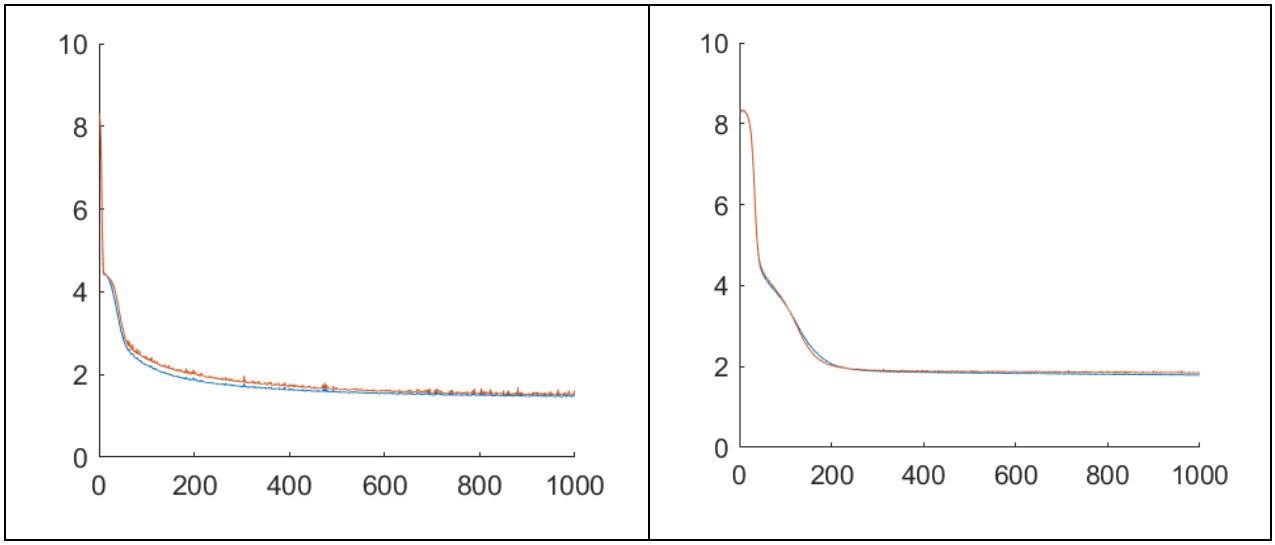


Figura 6 Stesso modello con TanH a sinistra e SoftPlus a destra in SGD. Sono visibili le oscillazioni prodotte dalla TanH

APPENDICE B

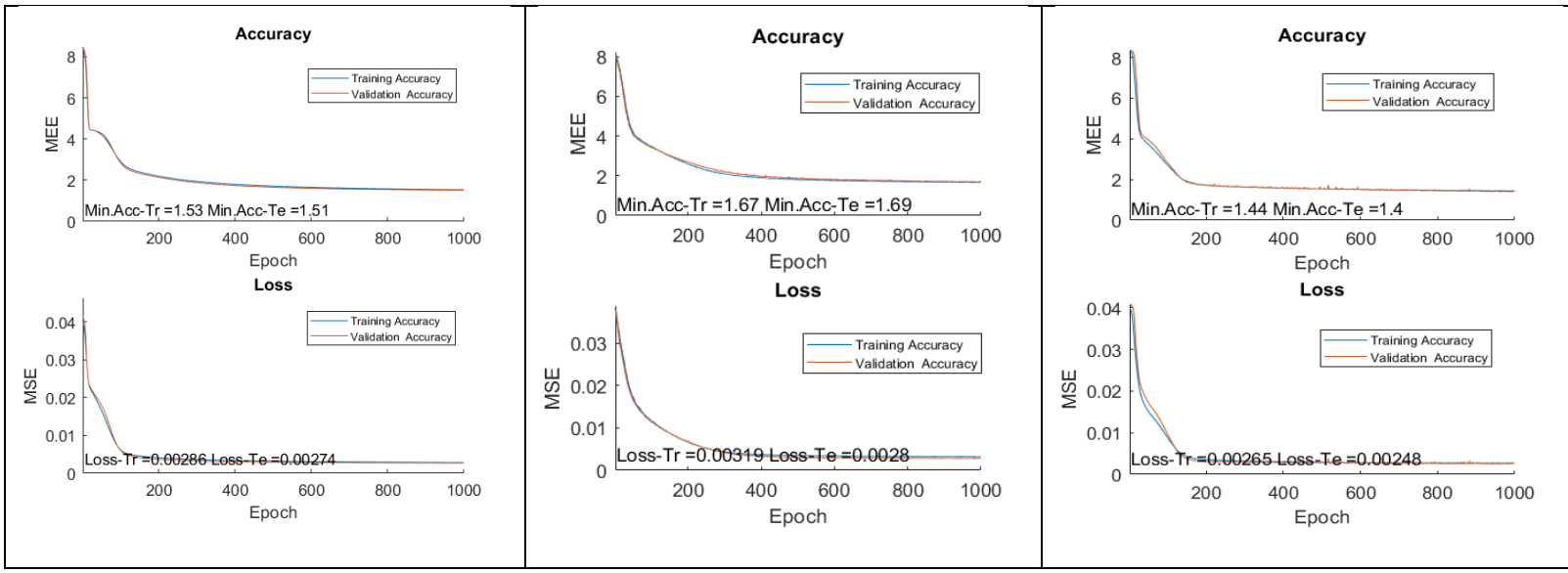


Figura 7 I migliori modelli individuati per ogni ottimizzatore. A sinistra SGD, al centro AdaGrad e a destra Adadelta. Confrontando le curve si nota come in SGD, la "gobba iniziale" dell'accuratezza sia più pronunciata rispetto agli altri due.

APPENDICE C

FUNZIONI DI INIZIALIZZAZIONE PESI

- Random
- Random con distribuzione Normale (Normal)
- Random con Fanin (Fanin)

Le funzioni di inizializzazione implementate fanno riferimento agli inizializer di Keras [4]. Ogni funzione di inizializzazione dispone di un range di default o può essere fornito come parametro, per quanto riguarda l'inizializzazione con distribuzione normale, possono essere passate anche la media e la varianza della distribuzione che si intende generare.

FUNZIONI DI ATTIVAZIONE IMPLEMENTATE

Ogni funzione di attivazione implementa anche la sua derivata

- Identità
- Logistic (Sigmoid)
- ReLU
- SoftPlus
- Tanh