



UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

FACOLTÀ DI SCIENZE E TECNOLOGIE APPLICATE
CORSO DI LAUREA IN INGEGNERIA INFORMATICA (LM-32)

**DEEP LEARNING IN SANITA':
ANALISI DI SEGNALI BIOMETRICI CON RETI
NEURALI**

Relatore:
Chiar.^{mo} Prof. Francesca Fallucchi

Candidato:
Enrico Sanna
Matr. N°: STA06233/LM32

ANNO ACCADEMICO
2017/2018

"Di solito sono proprio le cose non importanti che offrono il migliore campo di osservazione, e la possibilità di analizzare rapidamente cause ed effetti, il che è ciò che rende affascinante un'investigazione."

(Arthur Conan Doyle)

Sommario

Sommario	4
Indice Tabelle	8
Indice Figure	9
1. Introduzione	12
2. La CinC conference e la PhysioNet Challenge.....	17
2.1. Computing in Cardiology Conference.....	17
2.2. PhysioNet e PhysioBank.....	17
2.3. PhysioNet Challenge2017: AF Classification from a short single lead ECG recording	18
2.3.1. Dataset.....	19
2.3.2. Attribuzione del punteggio.....	21
2.3.3. Risultati.....	22
2.4. PhysioNet Challenge2018: You Snooze, You Win.....	23
2.4.1. Dataset.....	24
2.4.1. Attribuzione del punteggio.....	27
2.4.1. Risultati provvisori.....	29
3. Approccio teorico alla gestione di segnali continui con reti neurali	30
3.1. Vista d'insieme delle categorie di reti neurali	30
3.2. Modello matematico del Neurone e funzioni di attivazione	31
3.2.1. Sigmoid.....	33
3.2.2. Tanh.....	34
3.2.3. Rectified Linear Unit (ReLU)	35
3.2.4. SWISH.....	36

3.3.	Reti Neurali feed-forward	36
3.3.1.	Overfitting e dropout.....	40
3.3.1.	Overfitting e regolarizzazione.....	41
3.3.2.	Tipi algoritmi di ottimizzazione	42
3.4.	Reti Neurali Convoluzionali	43
3.4.1.	Filtro Convoluzionale	43
3.4.2.	Pooling	45
3.4.3.	Livelli di classificazione.....	47
3.5.	Residual Network ResNet	47
3.6.	Panoramica delle Reti Neurali ricorrenti (RNN).....	48
3.7.	Long Short Therm Memory (LSTM).....	50
4.	Panoramica delle principali librerie e framework per il Data Science in linguaggio Python.....	55
4.1.	Le librerie NumPy, SciPy, pandas, matplotlib	56
4.1.1.	La libreria NumPy	56
4.1.2.	La libreria SciPy.....	57
4.1.3.	La libreria Pandas.....	58
4.1.4.	La libreria Matplotlib.....	59
4.2.	Scikit-learn.....	59
4.3.	TensorFlow	62
4.3.1.	I tensori	62
4.3.2.	Lo struttura di un programma Tensorflow.....	64
4.3.3.	Tensorboard	65
4.3.1.	Performance su CPU, GPU e TPU	66
4.4.	Keras	67

4.4.1.	Keras Layers	69
5.	Modelli della PhysioNet / CinC Challenge2017	70
5.1.	Il modello di Xiong, Stiles, Zhao.....	70
5.2.	Il modello Andreotti (MATLAB Feature Based + Resnet)	72
5.3.	Modello Bidirectional LSTM - esempio MATLAB.....	78
5.3.1.	Il modulo di caricamento ReadPhysionetData	79
5.3.2.	Preparazione dataset in formato vettoriale	80
5.3.3.	Esecuzione modello LSTM senza estrazione feature	83
5.3.4.	Estrazione feature instantaneous frequency e power spectral entropy	86
5.3.5.	Esecuzione modello LSTM con estrazione feature.....	90
5.4.	Modello Bidirectional LSTM - Keras.....	92
5.4.1.	funzione di acquisizione dataset.....	92
5.4.2.	definizione del modello.....	94
5.4.3.	Esecuzione del modello	95
6.	Modelli della PhysioNet / CinC Challenge 2018.....	98
6.1.	Moduli comuni	98
6.1.1.	Prepare Entry	98
6.1.2.	Il modulo di acquisizione dei segnali.....	102
6.1.3.	Parti comuni: Il modulo di attribuzione del punteggio.....	104
6.1.1.	Parti comuni: Il modulo di log.....	107
6.2.	Descrizione Modello di esempio	108
6.3.	Descrizione Modello Keras con Dense.....	113
6.4.	Descrizione Modello Keras con ResNet	119
6.5.	Descrizione Modello Keras con bidirectional LSTM.....	122

6.6.	Descrizione Modello MATLAB con bidirectional LSTM.....	124
7.	Analisi dei risultati.....	127
7.1.	Risultati modelli CinC challenge 2017	127
7.1.1.	Risultati modelli LSTM MATLAB e Keras.....	128
7.2.	Risultati modelli CinC challenge 2018	128
7.2.1.	Risultati modello Logistic Regression Keras	130
7.2.1.	Risultati modello ResNet.....	132
7.2.1.	Risultati modello LSTM Keras e MATLAB.....	134
8.	Conclusioni	137
	Bibliografia	140

Indice Tabelle

Tabella 1 Distribuzione dei segnali nel training set	20
Tabella 2 Regole di conteggio per lo scoring della CinC Challenge 2017	22
Tabella 3 Risultati finali CinC Challenge 2017	23
Tabella 4 CinC challenge 2018: segnali fisiologici disponibili per la predizione delle apnee.....	24
Tabella 5 Distribuzione dei valori nei tracciati arousal della CinC challenge 2018	27
Tabella 6 Dimensioni comunità open source per librerie di Data Science	55
Tabella 7 Rank di un tensore in TensorFlow	62
Tabella 8 Forma e dimensione di un tensore in TensorFlow	63
Tabella 9 Risultati del modello Xiong, Stiles, Zhao.....	71
Tabella 10 Riepilogo risultati dei modelli di F. Andreotti per la Cinc Challenge 2017	78
Tabella 11 Risultati ufficiali CinC Challenge 2017.....	127
Tabella 12 CinC Challenge 2018, risultati dei modelli sperimentati.....	129
Tabella 13 CinC Challenge 2018, risultati ufficiali provvisori.....	129

Indice Figure

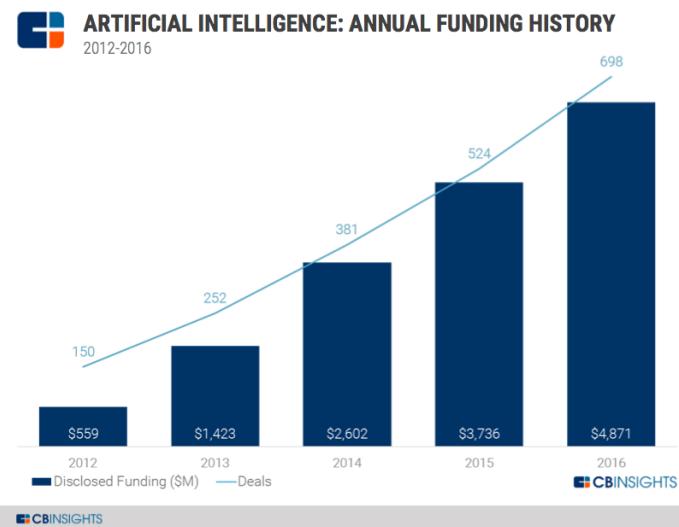
Figura 1 Esempio di segnali ECG del dataset	21
Figura 2 Rappresentazione grafica dei segnali di input delle CinC Challenge 2018	26
Figura 3 Rappresentazione matematica e geometrica della metrica AUPRC ...	28
Figura 4 Rappresentazione elementare di una rete neurale	30
Figura 5 Un esempio di rete neurale con vettore di input (x_1, x_2, \dots, x_n) a cui corrisponde un vettore di pesi (w_1, w_2, \dots, w_n) ed un rumore b . la funzione di attivazione è indicata con f	32
Figura 6 Grafico della funzione Gradient Descent	33
Figura 7 Grafico della funzione sigmoide e della sua derivata.....	34
Figura 8 Grafico della funzione Tanh e della sua derivata	35
Figura 9 Grafico della funzione ReLU e della sua derivata	35
Figura 10 Grafico della funzione SWISH e della sua derivata	36
Figura 11 Esempi di classificazione.....	37
Figura 12 Esempio di training con funzione di decisione lineare e nessun livello nascosto	38
Figura 13 Esempio di training con funzione di decisione non lineare e nessun livello nascosto	38
Figura 14 Esempio di training con funzione di decisione non lineare e 1 livello nascosto con 8 neuroni	39
Figura 15 Esempio di training con funzione di decisione non lineare e 2 livello nascosto con 4 neuroni	40
Figura 16 Una rete neurale standard (a sinistra), e a seguito dell'applicazione del dropout	41
Figura 17 Probabilità p di dropout per singolo neurone	41
Figura 18 Struttura di una rete convoluzionale	43
Figura 19 Filtro convoluzionale 2D con sfuocatura.....	45
Figura 20 Esempio di Filtro convoluzionale 2D con shift a sinistra dell'immagine	45

Figura 21 Rappresentazione dell'operazione di Max pooling	46
Figura 22 Esempio di Max Pooling con pool_size 3x3 e strides 1x1	47
Figura 23 Un esempio di blocco di Residual Network	48
Figura 24 Immagine schematica di una cella LSTM.....	49
Figura 25 Diagramma di una rete neurale ricorrente (RNN).....	49
Figura 26 Combinazione di input e stato precedente in una RNN.....	50
Figura 27 Struttura di una cella LSTM.....	51
Figura 28 Stato di una cella LSTM.....	51
Figura 29 LSTM Gate	52
Figura 30 LSTM, forget gate layer.....	52
Figura 31 LSTM, input gate layer	52
Figura 32 LSTM, generazione del nuovo stato Ct.....	53
Figura 33 LSTM, definizione dell'output	53
Figura 34 LSTM bidirezionale	54
Figura 35 Esempio di utilizzo della libreria matplotlib su una funzione esponenziale.....	59
Figura 36 Andamento della funzione sigmoidea logit.....	60
Figura 37 Il tool grafico Tensorboard	66
Figura 38 Confronto prestazioni della GPU Nvidia 1080 ti con altre soluzioni sul mercato	67
Figura 39 Benchmark librerie TensorFlow versione CPU e GPU.....	67
Figura 40 Immagine rappresentativa del modello Xiong per Cinc challenge 2017	71
Figura 41 Risultati del modello Xiong, Stiles, Zhao	72
Figura 42 Distribuzione delle quattro classi del dataset della Cinc challenge 2017 prima e dopo l'integrazione	72
Figura 43 Archittettura della rete Resnet proposta da Rajpurkar-Hannun e utilizzata da Ferndando Andreotti per la Cinc challenge 2017	74
Figura 44 Distribuzione dei record per durata temporale	80

Figura 45 Grafico dell'addestramento del modello LSTM MATLAB senza feature extraction	84
Figura 46 Matrici di confusione del modello LSTM MATLAB senza feature extraction.....	85
Figura 47 Rappresentazione grafica di due segnali di classe Norma e AFib	86
Figura 48 Instantaneous Frequency di un record di esempio (classe N e A) ...	87
Figura 49 Spectral Entropy di un record di esempio (classe N e A).....	88
Figura 50 Grafico dell'addestramento del modello LSTM MATLAB con feature extraction.....	90
Figura 51 Matrici di confusione del modello LSTM MATLAB con feature extraction.....	91
Figura 52 Rappresentazione degli indicatori intermedi TPR e TNR.....	106
Figura 53 Modello sample vs LogRegKeras, distribuzione dei record per valore di AUPRC	131
Figura 54 Performance indicate da Tensorflow per il modello LogisticRegressionKeras.....	132
Figura 55 Modello sample vs ResNet, distribuzione dei record per valore di AUPRC.....	133
Figura 56 Performance indicate da Tensorflow per il modello ResNet.....	134
Figura 57 Performance indicate da Tensorflow per il modello LSTM.....	135

1. Introduzione

Negli ultimi anni l'intelligenza artificiale, il machine learning e il deep learning hanno acquisito una sempre maggiore popolarità nella comunità scientifica, e parallelamente si è osservata una crescita enorme negli investimenti delle principali company nel campo dell'Information Technology.



Aziende del calibro di Amazon, Google, IBM, Microsoft, Tesla, NVIDIA e Facebook stanno estendendo il loro modello di business verso questo settore, con un'attenzione diffusa che per certi aspetti può ricordare la corsa all'oro negli Stati uniti del XIX secolo [1].

Tra le cause scatenanti di questo fenomeno si possono collocare diversi eventi:

- L'accumulazione di una enorme mole di dati grazie alla crescita di velocità della rete, la diffusione di dispositivi mobili, l'abbassamento dei costi dello storage, e lo sviluppo crescente di sistemi informativi aziendali integrati.
- Una capacità computazionale molto elevata, anche grazie all'avvento della nuova generazione di Graphical Processing Unit.

- La necessità di trovare un vantaggio competitivo industriale attraverso l'automazione o il monitoraggio di alcuni processi, oppure attraverso l'analisi di dati grezzi per comprendere meglio i bisogni del mercato.

Nel corso della mia attività lavorativa quasi decennale nel mondo della sanità ho potuto rilevare come molti aspetti siano paragonabili a quelli presenti nei settori industria e finanza. Non bisogna dimenticare che il costo della sanità incide in media tra l'otto e il dieci per cento del prodotto interno lordo degli stati [1], con picchi fino al 17% negli USA.

Intorno a questa enorme mole di danaro ruotano tantissime aziende pubbliche e private che fanno ricorso sempre più all'intelligenza artificiale come strumento per ricercare un vantaggio competitivo. Inoltre le stesse organizzazioni sanitarie ed enti di ricerca stanno facendo sempre più ricorso a queste tecnologie nella ricerca medica, nei processi di diagnosi delle patologie, e nello studio di protocolli terapeutici.

Si riporta di seguito una descrizione delle principali aree di interesse:

- Farmaceutica
- Scoperta di frodi alle assicurazioni sanitarie
- Diagnosi precoce di patologie degenerative come l'Alzheimer
- Mappatura del genoma e ricerca genetica
- Diagnosi automatica di immagini e segnali

L'area della **Diagnosi automatica** riguarda tutte le tecniche di analisi di immagini come Risonanza Magnetica e la Tomografia Computerizzata, e di segnali monodimensionali come Elettrocardiogramma, Elettroencefalogramma, Glicemia e altri, al fine di costruire modelli di predizione basati su deep learning per la diagnosi di malattie cardiologiche, oncologiche e diabetologiche.

Questo tipo di malattie dette croniche sono ai primi posti nelle cause di mortalità in quasi tutte le parti del mondo [2]. Si tratta di malattie che pur avendo origine in età giovanile, spesso richiedono decenni prima di manifestarsi clinicamente ed essere scoperte. Accanto ai fattori di rischio non modificabili come quelli

ereditari, ci sono una serie di fattori legati allo stile di vita, come alimentazione poco sana, consumo di alcol o tabacco, scarsa attività fisica.

In quest'area l'obiettivo che si cerca di perseguire è quello di sviluppare strumenti di supporto alla diagnosi sempre più efficienti, che possano aiutare il personale medico a ridurre l'inevitabile percentuale di errate o mancate diagnosi dovute ad errore umano, ed allo stesso tempo si persegue l'idea di poter sviluppare strumenti di diagnosi automatica a basso costo da diffondere capillarmente per migliore le coperture delle campagne di screening.

Alcuni dei risultati oggi più notevoli nel campo delle immagini sono costituiti da Watson Health[3] e DeepMind Health [4], due sistemi sviluppati rispettivamente da IBM e Alphabet (Google), che oltre agli strumenti di visione integrano strumenti di Natural Language processing e modelli per la composizione delle terapie oncologiche.

L'oggetto degli studi del candidato e quindi di questa tesi riguarda la classificazione di segnali biometrici. In particolare sono state affrontate due tematiche di ricerca oggetto di una challenge scientifica promossa dall'ente americano PhysioNet e dalla conferenza internazionale Computing in Cardiology. Questa conferenza si ripete annualmente dal 1974 in modo itinerante, con le più importanti università del mondo che si alternano per la sua organizzazione.

La prima tematica, oggetto della Challenge 2017 organizzata dall'Università di Rennes (FR), riguarda l'analisi di segmenti ECG per la diagnosi della fibrillazione atriale, la patologia cardiologica più diffusa nell'ambito delle aritmie, ovvero le alterazioni del ritmo cardiaco. In relazione a questo argomento sono state analizzate diverse soluzioni presentate dai vari gruppi di ricerca partecipanti e si è effettuata una prova sperimentale di implementazione di modello RNN.

La seconda tematica, oggetto della Challenge 2018 organizzata dall'Università di Maastricht(NL), riguarda invece l'analisi di paziente affetti da Sindrome da apnee ostruttive del sonno [5] con un dataset composto da 13 segnali biometrici. Questa competizione non è ancora terminata, pertanto sono disponibili solo

risultati provvisori. In questo caso si è provato a riprodurre il modello di esempio e poi a produrre ulteriori due soluzioni rispettivamente facenti uso di reti neurali convoluzionali e ricorrenti.

Storicamente i modelli e gli elaborati proposti si suddividono in due categorie:

- Modelli basati sull'analisi delle caratteristiche dei segnali, con una progettazione di algoritmi tradizionali basati sullo stato dell'arte della conoscenza medica e tradotti in codice implementando delle regole rigide suggerite dai clinici
- Modelli basati sul deep learning, che non prevedono la scrittura di regole predefinite ma solo la costruzione sperimentale di modelli di classificatori che derivano dinamicamente le regole dall'osservazione dei dati

In questa tesi discuteremo solo la seconda categoria, con qualche breve cenno alla prima prevalentemente con finalità comparative.

Si è scelto di utilizzare come strumenti tecnologici le librerie Python TensorFlow e Keras, per la loro sempre maggiore diffusione, e la loro caratteristica di essere open source e promossi da un gigante come il gruppo Google Brain.

La trattazione prosegue nella maniera seguente:

- Il capitolo 2 descrive nel dettaglio gli argomenti delle due challenge e la storia delle organizzazioni promotrici; viene inoltre fornita una descrizione dei dataset ed una sintesi dei risultati raggiunti.
- Il capitolo 3 fornisce una trattazione teorica sintetica di tutti gli argomenti approfonditi che sono serviti al candidato come base tecnica per la comprensione di questa complessa area dell'Informatica.
- Il capitolo 4 fornisce una panoramica delle librerie di calcolo numerico e deep learning
- Il capitolo 5 riporta una descrizione dei modelli analizzati e sperimentati per la challenge 2017
- Il capitolo 6 riporta una descrizione dei modelli analizzati e sperimentati per la challenge 2018

- Il capitolo 7 riporta un riepilogo dei risultati ed una loro discussione d'insieme
- Il capitolo 8 riprende i punti del percorso di studio affrontato e riporta le conclusioni e le ipotesi di lavoro future

2. La CinC conference e la PhysioNet Challenge

In questa sezione viene approfondita la trattazione di due importanti organismi della ricerca in campo di analisi dei segnali fisiologici come la Computing in Cardiology Conference e la risorsa PhysioNet.

2.1. Computing in Cardiology Conference

La Computing in Cardiology¹ (di seguito CinC) è una associazione scientifica che raggruppa ricercatori e professionisti per discutere delle più attuali tematiche di ricerca nei campi della medicina, della fisica, dell'ingegneria e della computer science volte all'analisi computerizzata della cardiologia clinica e della fisiologia cardiovascolare.

La CinC organizza dal 1974 una conference annuale durante la quale i ricercatori vengono invitati a sottomettere le loro pubblicazioni e poi a presentare i risultati delle proprie ricerche. Le sedi di queste conferenze si sono alternate negli anni tra gli Stati Uniti e l'Europa, con due significative aperture verso il resto del Mondo costituire da China 2011 e Israele 1989. L'Italia è stata scelta come sede della CinC per ben 3 volte, con Firenze 1981, Venezia 1991, Bologna 2008. La partecipazione italiana è molto forte anche nella composizione degli Organismi direttivi con tre componenti su dodici, seconda solo agli Stati Uniti. Le università coinvolte sono il Politecnico di Milano e l'Università di Bologna.

L'edizione 2018 della CinC si terrà nei giorni 23-27 Settembre presso l'università di Maastricht. Dalla lettura del programma preliminare emerge una grande partecipazione con una media di 25 presentazioni per giorno. Nell'ultima giornata del 26 Settembre verranno presentati i contributi della PhysioNet / CinC Cardiology Challenge 2018, su cui sono basati gli studi di questa tesi.

2.2. PhysioNet e PhysioBank

PhysioNet.org è un portale e un archivio pubblico di risorse per la ricerca nell'ambito di segnali fisiologici complessi che è mantenuto da due istituti di ricerca statunitensi:

¹ Il sito pubblico della CinC conference è <http://cinc.org>

- il *National Institute of General Medical Sciences* (NIGMS);
- il *National Institute of Biomedical Imaging e Bioengineering* (NIBIB);

entrambi facenti capo al *United States Department of Health and Human Services (HHS)*, il dicastero del governo federale degli USA che si occupa della salute dei cittadini. Il portale è il risultato di venti anni di studi di diversi ricercatori clinici, matematici e fisici ed è stato fondato in occasione dell'edizione della CinC conference del 2000 presso il Massachusetts Institute of Technology.

Si compone di tre macroaree indipendenti:

- PhysioBank, un ampio archivio contenente segnali fisiologici registrati in digitale e altri dati destinati all'uso nell'ambito della ricerca biomedica;
- PhysioToolkit, una raccolta di software per l'acquisizione e la processazione dei segnali, e per la cognizione di eventi significativi nei segnali;
- PhysioNetWorks, un laboratorio virtuale per la collaborazione tra i ricercatori.

PhysioNet è uno spazio completamente libero, e l'unico vincolo per l'utilizzo dei suoi strumenti e dei suoi dati è quello di riportarne una citazione nelle pubblicazioni e nei lavori che vengono prodotti a partire dalle sue risorse. In questo portale ci sono anche una serie di tutorial a livelli di difficoltà crescente per la formazione di nuovi aspiranti ricercatori.

PhysioNet, in collaborazione con la conferenza annuale Computing in Cardiology, si occupa di ospitare una challenge annuale nella quale studenti e ricercatori più affermati possono misurarsi su problemi aperti di interesse scientifico, con l'utilizzo di dataset e strumenti comuni.

2.3. PhysioNet Challenge2017: AF Classification from a short single lead ECG recording

LaPhisionet/CinC Challenge 2017 ha avuto luogo in Francia presso l'Università di Rennes, nei giorni 24/27 Settembre. L'obiettivo è stato lo sviluppo di algoritmi di classificazione di singoli segnali ECG della durata compresa tra 30 e 60 secondi.

Sono individuate quattro classi:

- N, ritmo cardiaco normale;
- A, fibrillazione atriale (AF);
- O, ritmo alternato;
- ~, segnale troppo disturbato per essere classificato.

La fibrillazione atriale è uno dei più comuni tipi di aritmia, e colpisce circa 1-2% della popolazione. Secondo la definizione dell'American College of Cardiology è un tipo di tachiaritmia caratterizzata da un'attivazione atriale prevalentemente scoordinata, che provoca nel tempo un deterioramento della funzione meccanica atriale. L'individuazione della fibrillazione atriale è problematica, in quanto essa è episodica. I rilevatori di AF basati sulle caratteristiche si distinguono in due categorie:

- rilevatori basati sull'analisi dell'attività atriale;
- rilevatori basati sulla risposta ventricolare

Inoltre ci sono degli approcci basati sul machine learning che cercano di combinare entrambi gli aspetti ottenendo delle performance di classificazione migliori.

2.3.1. *Dataset*

Il dataset di training è costituito da 8.528 registrazioni ECG campionati singolarmente da dispositivi AliveCor, della durata totale dai 9 ai 60 secondi. Il target del training test viene fornito in un unico file con una riga per segnale, con ID del record e nome della classe (ad esempio A00001, N).

La frequenza di campionamento dei segnali è di 300 Hertz e gli stessi sono stati preprocessati con un filtro passa banda e quindi digitalizzati in formato MATLAB V4 (WFDB).

Il dataset di test contiene 3.658 registrazioni di lunghezza e frequenza analoga, e come consuetudine non è stato inizialmente condiviso con il pubblico per poter essere utilizzato come punteggio per la challenge. Il mantenere il test set privato è opportuno per evitare lo sviluppo di soluzioni non generali e troppo

specifiche, sarebbe ad esempio facilissimo sviluppare un classificatore che riconosce “a memoria” i segnali del test set.

Nella Tabella 1 è riportata la distribuzione dei segnali del training set per classe di appartenenza e durata temporale. Come si può osservare la numerosità delle classi è fortemente sbilanciata a favore della classe Normal, per la quale sono presenti un numero di record doppi rispetto alla classe O e di sette volte il numero dei record della classe A. Inoltre come si può notare la maggior parte dei record ha una durata di circa 9000 campioni (30 secondi campionati a 300Hz).

Tabella 1 Distribuzione dei segnali nel training set

Type	# recording	Time length (s)				
		Mean	SD	Max	Median	Min
Normal	5154	31.9	10.0	61.0	30	9.0
AF	771	31.6	12.5	60	30	10.0
Other rhythm	2557	34.1	11.8	60.9	30	9.1
Noisy	46	27.1	9.0	60	30	10.2
Total	8528	32.5	10.9	61.0	30	9.0

Si riporta nella Figura 1 un esempio di segnale per ciascuna classe. L'osservazione dei segnali evidenzia alcuni aspetti:

- la classe Normal sia caratterizzata da una frequenza ed una forma abbastanza costanti nel tempo;
- al contrario la classe AF contente i record di pazienti affetti da fibrillazione atriale mostra una forma regolare ma una periodicità irregolare;
- la classe Other rhythm mostra una regolarità temporale ma una forma più irregolare della classe Normal;
- infine la classe Noisy è caratterizzata da un segnale fortemente irregolare e spigoloso.

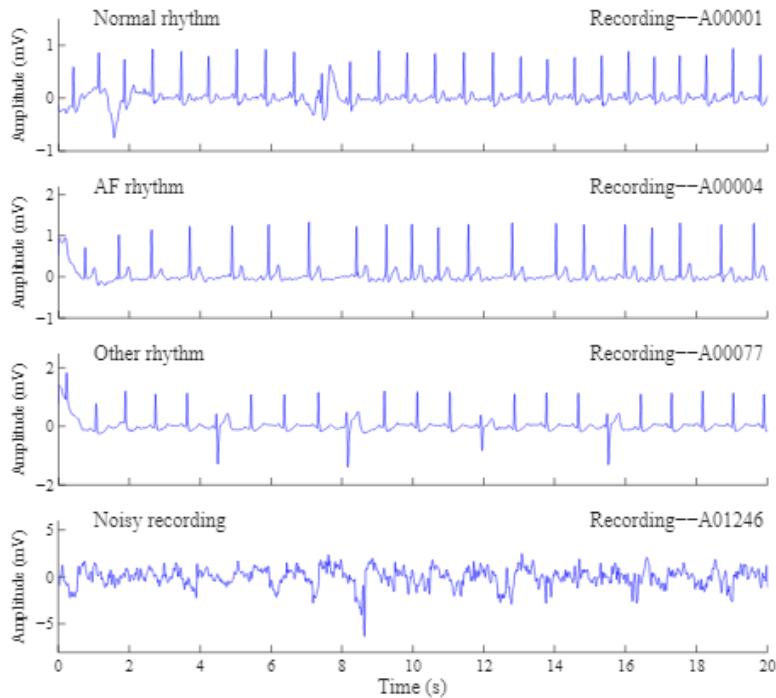


Figura 1 Esempio di segnali ECG del dataset

2.3.2. Attribuzione del punteggio

Il punteggio di valutazione è stato definito come la F1 measure (detta anche F-score) media delle quattro classi. La F1 è una misura che combina insieme la precisione e il recupero di una classificazione binaria, dove la precisione è il numero di veri positivi diviso il numero di tutti i risultati positivi, e il recupero (recall) è il numero dei veri positivi diviso il numero di tutti i test che sarebbero dovuti essere classificati come positivi, ovvero veri positivi più falsi negativi.

L'attribuzione dei segnali nelle quattro classi può essere riepilogata nella Tabella 2, dove la lettera Maiuscola (primo carattere) rappresenta la classe predetta e la lettera minuscola la classe reale. La diagonale in colore verde rappresenta le predizioni corrette.

Tabella 2 Regole di conteggio per lo scoring della CinC Challenge 2017

		Predicted Classification				
		Normal	AF	Other	Noisy	Total
Reference Classification	Normal	<i>Nn</i>	<i>Na</i>	<i>No</i>	<i>Np</i>	$\sum N$
	AF	<i>An</i>	<i>Aa</i>	<i>Ao</i>	<i>Ap</i>	$\sum A$
	Other	<i>On</i>	<i>Oa</i>	<i>Oo</i>	<i>Op</i>	$\sum O$
	Noisy	<i>Pn</i>	<i>Pa</i>	<i>Po</i>	<i>Pp</i>	$\sum P$
	Total	$\sum n$	$\sum a$	$\sum o$	$\sum p$	

La F1-Measure viene quindi definita per le quattro classi come:

$$\text{Normal rhythm: } F1n = \frac{2 \times Nn}{\sum N + \sum n}$$

$$\text{AF rhythm: } F1a = \frac{2 \times Aa}{\sum A + \sum a}$$

$$\text{Other rhythm: } F1o = \frac{2 \times Oo}{\sum O + \sum o}$$

$$\text{Noisy rhythm: } F1p = \frac{2 \times Pp}{\sum P + \sum p}$$

Il punteggio finale è dato dalla media dei quattro valori:

$$F1 = \frac{F1n + F1a + F1o + F1p}{4}$$

2.3.3. Risultati

La CinC 2017 ha coinvolto 75 gruppi indipendenti di partecipanti, i quali hanno approcciato il problema con una grande varietà di metodi tradizionali e innovativi, dai classificatori random forest al deep learning, proponendo un totale di circa 300 soluzioni. Il primo posto è andato a pari merito a quattro gruppi di ricercatori con un F1 score medio di 0.83. I gruppi di ricercatori che condividono la prima posizione fanno parte delle seguenti centri:

- Università di Santiago de Compostela, Santiago de Compostela, Spain
- Tata Consultancy Services, un'azienda indiana di servizi informatici parte del grandissimo gruppo industriale Tata Group (100 miliardi di fatturato nel 2017)

- Gruppo misto delle Università di Tampere (Finlandia), Illinois (USA), Doha (Qatar)
- Peking University di Pechino, Cina

Si riportano di seguito nella Tabella 3 i risultati finali. Come si può osservare, a seguito della conclusione della conference, è stato prodotto un modello riepilogativo a votazione che è riuscito a migliorare sensibilmente il risultato dei primi classificati sfiorando la soglia di 0.87. Il processo di voting è costituito essenzialmente dal raccogliere le probabilità predette da ciascun modello, e farne la media pesata per risultato dei test. Ad esempio un valore di probabilità di 1 del modello Teijeiro et al. viene considerato come un valore pari a 0.83.

Tabella 3 Risultati finali CinC Challenge 2017

Rank	Entrant	Test	Validation	Train
=1	<i>Teijeiro et al.</i>	0.831	0.912	0.893
=1	<i>Datta et al.</i>	0.829	0.990	0.970
=1	<i>Zabihi et al.</i>	0.826	0.968	0.951
=1	<i>Hong et al.</i>	0.825	0.990	0.970
=5	<i>Baydoun et al.</i>	0.822	0.859	0.965
=5	<i>Bin et al.</i>	0.821	0.870	0.875
=5	<i>Zihlmann et al.</i>	0.821	0.913	0.889
=5	<i>Xiong et al.</i>	0.818	0.905	0.877
-	Voting (top 10)	0.844	-	-
-	Voting (top 30)	0.847	-	-
-	Voting (top 50)	0.851	-	-
-	Voting (all 75)	0.855	-	-
-	Voting (LASSO)	0.858	-	-
-	Voting (LASSO+)	<u>0.868</u>	-	-

2.4. PhysioNet Challenge2018: You Snooze, You Win

L'edizione 2018, dal titolo “*You Snooze, You Win*” [6] verte attorno al miglioramento della comprensione delle ragioni per le quali dormiamo. In particolare l'analisi si focalizza su uno dei più studiati disturbi del sonno, ovvero la Sindrome delle Apnee Ostruttive del Sonno.

L'apnea è caratterizzata da un completo collasso delle vie respiratorie che provoca un risveglio, e un conseguente deterioramento della qualità del sonno. Tra le altre cause di risveglio ci sono il digrignamento dei denti, da ostruzioni delle vie aeree parziali o dal russamento.

La challenge prende in esame una serie di segnali fisiologici, raccolti durante studi sul sonno tramite polisonnografia, al fine di rilevare altre fonti di eccitazione diverse dall'apnea.

2.4.1. Dataset

I dati utilizzati sono stati prodotti da una collaborazione tra il *Massachusetts General Hospital's (MGH)*, il *Computational Clinical Neurophysiology Laboratory (CCNL)*, e il *Clinical Data Animation Laboratory (CDAC)*. Il dataset è costituito dal campionamento di 1,985 soggetti monitorati nel corso di una notte di sonno dal MGH. Il dataset è quindi suddiviso in modo bilanciato con 994 record nel training set e 989 elementi nel test set.

I segnali fisiologici registrati su ciascun soggetto campione durante la notte, sono l'elettroencefalogramma (EEG), l'elettrooculografia (EOG), l'elettromiografia (EMG), l'elettrocardiogramma (ECG) e la saturazione di ossigeno (SaO₂). Escludendo SaO₂, tutti i segnali sono stati campionati a 200 Hz e sono stati misurati in microvolt. Per comodità analitica, SaO₂ è stato ricampionato a 200 Hz e viene misurato in percentuale.

La Tabella 4 riporta la codifica dei segnali e l'unità di misura in cui gli stessi sono espressi.

Tabella 4 CinC challenge 2018: segnali fisiologici disponibili per la predizione delle apnee

Signal Name	Units	Signal Description
SaO ₂	%	Oxygen saturation
ABD	µV	Electromyography, a measurement of abdominal movement
CHEST	µV	Electromyography, measure of chest movement
Chin1-Chin2	µV	Electromyography, a measure of chin movement
AIRFLOW	µV	A measure of respiratory airflow
ECG	mV	Electrocardiogram, a measure of cardiac activity
E1-M2	µV	Electrooculography, a measure of left eye activity
O2-M1	µV	Electroencephalography, a measure of posterior activity
C4-M1	µV	Electroencephalography, a measure of central activity
C3-M2	µV	Electroencephalography, a measure of central activity
F3-M2	µV	Electroencephalography, a measure of frontal activity
F4-M1	µV	Electroencephalography, a measure of frontal activity
O1-M2	µV	Electroencephalography, a measure of posterior activity

I segnali fisiologici sono forniti in un unico file formato MATLAB v4 con dentro 13 canali. I diversi record hanno durata di campionamento differente. Di seguito nella Figura 2 una rappresentazione grafica dei 13 segnali sovrapposti, dalla quale si possono intuire le loro differenti caratteristiche.

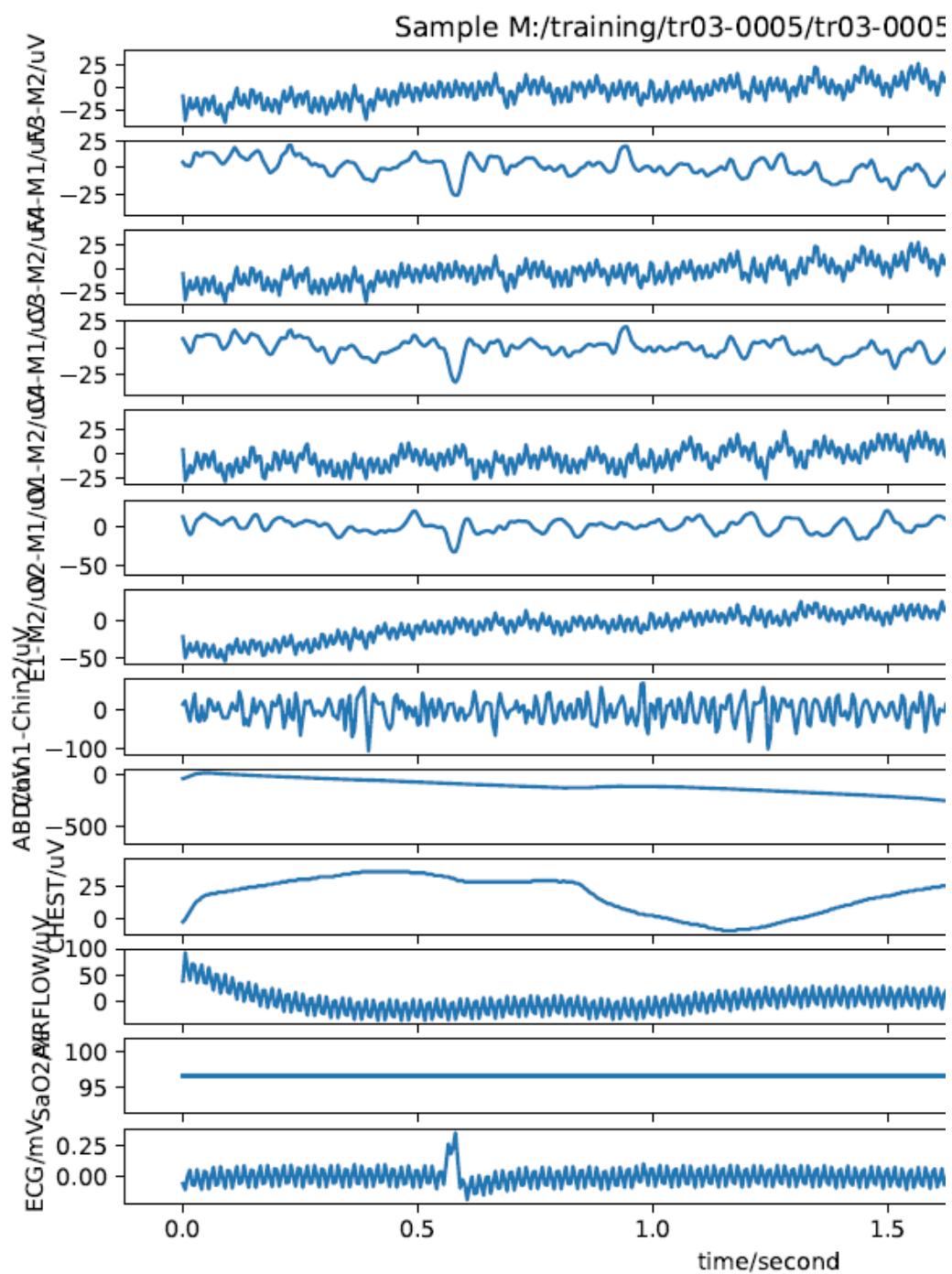


Figura 2 Rappresentazione grafica dei segnali di input delle CinC Challenge 2018

L'obiettivo della sfida è utilizzare le informazioni provenienti dai segnali disponibili per classificare correttamente le regioni di eccitazione target (non-RERA, risveglio non correlato a sforzo respiratorio). Queste classificazioni sono

già riassunte in un vettore monodimensionale, di lunghezza pari ai segnali fisiologici campionati, che presenta per ogni elemento i seguenti valori:

- 1 Designates arousal regions: risveglio da individuare
- 0: Designates non-arousal regions: zone da non individuare
- -1: zone dubbie (si considerano valide entrambe le predizioni)

Il vettore con i risultati attesi è fornito in formato MATLAB V 7.3

L'obiettivo della challenge è quello di stimare sulla base dei 13 segnali un vettore di arousal, e quindi confrontarlo con quello presente.

Come consuetudine, attualmente nel test set non è disponibile il tracciato arousal, che verrà utilizzato per lo scoring delle varie soluzioni sottoposte.

Si riporta di seguito nella Tabella 5 un esempio (tr08-0157) di distribuzione dei valori del tracciato arousals, che mostra come oltre l'80% sia costituita dal valore zero.

Tabella 5 Distribuzione dei valori nei tracciati arousal della CinC challenge 2018

Valore	casi	percentuale
0.	4578818	83%
-1	816907	15%
1.	106275	2%
totale	5502000	1

2.4.1. Attribuzione del punteggio

Viene richiesto ai partecipanti di produrre un modello che sulla base del training set (segnali fisiologici e tracciato arousals) sia in grado di effettuare predizione sul record del test set. L'oggetto della sottomissione è quindi la generazione di un file testuale (.vec) per ognuno dei 989 soggetti che descriva la possibilità di apnea (valore 1) per ogni istante del tracciato.

La valutazione dei risultati avviene sulla base gli indicatori di classificazione binaria “Precisione” e “Richiamo” (recall).

Il recall è definito da un rapporto che ha al numeratore l'intersezione tra i record arousal (valore =1) reali e quelli predetti (prob. stimata $>0,001$) e al denominatore il numero totale di campioni valorizzati a uno presenti nel

tracciato. È un indicatore che serve per stimare qual'è la porzione di dati che il modello riesce a recuperare.

La precisione è un rapporto che ha al numeratore sempre l'intersezione tra i record arousal (valore =1) reali e quelli predetti (prob. stimata >0,001) ma ha al denominatore il numero di campioni attinenti. È un indicatore che misura la capacità del modello di marcare solo i casi afferenti a quella classe, evitando i falsi positivi.

Le formule riportate di seguito sono riportate dal sito PhyioNet:

$$R_j = \frac{\text{number of arousal sample with predicted probability } (j/1000) \text{ or greater}}{\text{total number of arousal samples}}$$

$$R_j = \frac{\text{number of arousal sample with predicted probability } (j/1000) \text{ or greater}}{\text{total number of samples with predicted probability } (j/1000) \text{ or greater}}$$

Per combinare questi due dimensioni in un unico indicatore si utilizza l'*Area Under The Precision Recall Curve* (AUPRC) ovvero l'area definita inserendo precisione e richiamo in un diagramma cartesiano, come nella seguente Figura 3.

$$\text{AUPRC} = \sum_j P_j(R_j - R_{j+1})$$

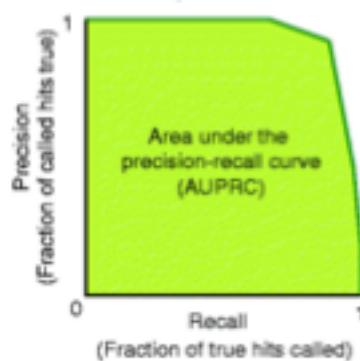


Figura 3 Rappresentazione matematica e geometrica della metrica AUPRC

2.4.1. Risultati provvisori

Il numero di partecipanti ammessi alla fase preliminare è di 25 team. Il modello di esempio proposto implementa in Python con le librerie SkLearn un semplice classificatore lineare, con l'utilizzo del solo segnale SaO₂.

```
AUPRC,AUROC,authors
0.439,0.902,Matthew HP; Bahareh Pourbabae
0.244,0.852,Yang Liu; Runnan He
0.228,0.822,Márton Görög; Bálint Varga; Péter Hajas
0.214,0.815,Andrea Patane'; Shadi Ghiasi; Marta Kwiatkowska
0.160,0.765,Sardar Ansari
0.129,0.735,Filip Plesinger; Petr Nejedly; Ivo Viscor; Josef Halamek; Petr Andrla; Pavel Jurak
0.119,0.739,Philip de Chazal; Nadi Sadr
0.088,0.700,Wang Chao
0.088,0.700,Sample entry (Python)
```

3. Approccio teorico alla gestione di segnali continui con reti neurali

In questa sezione vengono approfonditi in dettaglio alcuni temi della teoria delle reti neurali che sono stati oggetto di studio durante questo lavoro di tesi e che saranno di supporto per la comprensione dei capitoli successivi.

3.1. Vista d'insieme delle categorie di reti neurali

Una rete neurale è un modello matematico che si ispira vagamente ad una rete neurale biologica ed è composto da una serie di strati di neuroni interconnessi tra loro in modo caratterizzato in uno stato di ingresso, uno o più strati nascosti e uno strato di uscita. La Figura 4 riporta in colori differenti le diverse tipologie di neuroni, e le connessioni tra essi definite da archi direzionali.

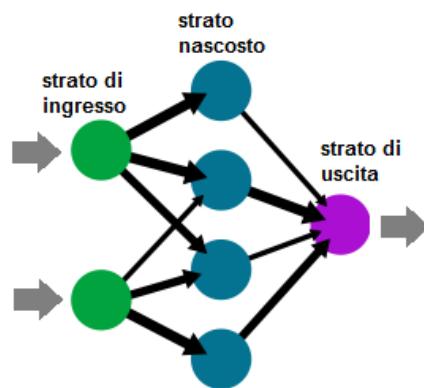


Figura 4 Rappresentazione elementare di una rete neurale

Nel caso di apprendimento supervisionato, dove si dispone di un archivio di dati di training già associati al risultato desiderato, l'obiettivo della rete è quello di provare empiricamente a stabilire le relazioni tra dati di input e valore di output atteso. Questo lavoro si traduce nell'assegnazione di pesi (valori numerici) assegnati a ciascun nodo, pesi che vengono modificati secondo vari algoritmi in dipendenza di un risultato corretto o meno.

Il pregio delle reti neurali è quello di essere costruita per lavorare in parallelo, e poter quindi processare archivi di dati di dimensioni molto grandi. Il loro principale difetto è quello di produrre un modello black box che è difficilmente comprensibile dalla mente umana e spiegabile in linguaggio naturale. I risultati

di un modello a rete neurale devono essere semplicemente accettati così come stimati, e per questo una grande importanza deve essere dedicata alla rappresentatività del dataset utilizzato per il training rispetto al mondo reale.

Dal punto di vista della topologia delle reti neurali si distinguono tra le altre due categorie:

- le reti neurali feed-forward in cui non ci sono cicli tra i singoli neuroni e strati, e dove quindi il flusso di informazioni avviene in un solo verso (dallo strato di input verso lo strato di output). Le reti con questa topologia sono le più semplici e le prime storicamente utilizzate, e si definiscono come senza memoria. Un esempio di questo tipo di reti è dato nella Figura 4.
- le reti neurali ricorrenti, dove la propagazione dei dati avviene sia in avanti che all'indietro. In queste reti la presenza di cicli di informazioni va a costituire una sorta di memoria della rete neurale, in quanto l'output al tempo t è dipendenza non solo del rispettivo ingresso ma anche direttamente degli input precedenti.

3.2. Modello matematico del Neurone e funzioni di attivazione

Il modello matematico di un neurone artificiale ricorda il comportamento di una rete neurale biologica, dove sono presenti i seguenti elementi:

- il soma o corpo cellulare,
- l'assone o la linea di uscita delle informazioni,
- il dendrite o linea di entrata del neurone che riceve segnali multipli da altri assoni tramite le sinapsi.

Quando le sollecitazioni ricevute dai dendriti superano una certa soglia, il neurone si attiva e produce un segnale detto "potenziale d'azione" che viene trasmesso tramite l'assone. In caso contrario rimane in uno stato di riposo.

In analogia con il caso biologico, un neurone artificiale è caratterizzato da n segnali di input x_1, x_2, \dots, x_n che vengono pesati tramite moltiplicazione con i pesi w_1, w_2, \dots, w_n , aventi valori in $[0,1]$, e poi passati a quella che si chiama funzione di attivazione. Quando presente il vettore del rumore bias, i segnali vengono anche sommati a questo canale.

Il funzionamento di un neurone artificiale [7] è raffigurato nella seguente Figura 5, ma può essere riassunto in forma vettoriale come $Y = (X^*W + b)$, dove Y è il vettore in uscita dal neurone.

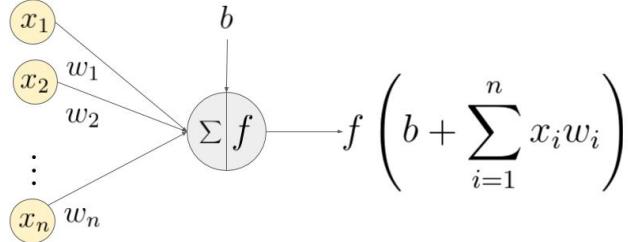


Figura 5 Un esempio di rete neurale con vettore di input (x_1, x_2, \dots, x_n) a cui corrisponde un vettore di pesi (w_1, w_2, \dots, w_n) ed un rumore b . la funzione di attivazione è indicata con f

Da una parte sia l'applicazione dei pesi che del bias sono trasformazioni lineari, rispettivamente moltiplicazioni e somme, mentre la funzione di attivazione è solitamente una trasformazione non lineare.

L'apprendimento di una rete neurale funziona secondo questi passi:

- addestra la rete con casi noti (già classificati), ovvero con un input X ed un output desiderato Y ;
- il passaggio di X nella rete neurale provoca l'uscita di Y' ;
- La differenza tra valore atteso e predetto, $(Y - Y')$ viene convertita in una metrica chiamata loss function, che assume un valore alto quando la rete neurale commette molti errori;
- Al fine di ridurre il loss la rete provvede ad aggiornare i propri pesi e bias.

La riduzione del loss avviene attraverso quella che si chiama discesa del gradiente, ovvero si utilizzano le derivate parziali della funzione di loss per ottenere la direzione in cui muoversi verso il punto più basso della funzione, che come vede dall'immagine seguente ha una forma a "ciotola". Nella Figura 6 la discesa del gradiente è rappresentata in un diagramma tridimensionale dove l'altezza corrisponde alla funzione loss, e le parti tendenti verso il colore blu sono quelle aree dove il valore è più basso.

Gradient Descent

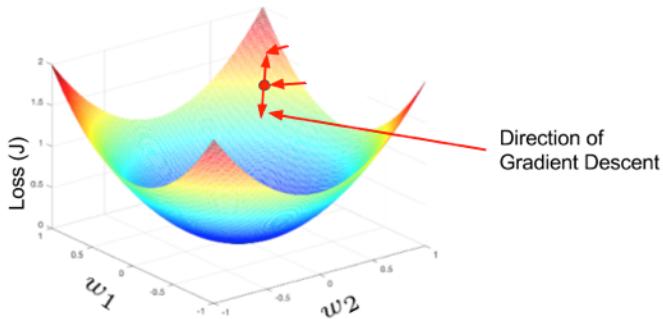


Figura 6 Grafico della funzione Gradient Descent

Le funzioni di attivazione divenute famose nel corso degli anni sono differenti, e tutt'oggi c'è una branca della ricerca che si occupa di analizzare quali di esse siano più adatta ai diversi scenari di applicazione delle reti neurali.

Se ne riporta di seguito un elenco non esaustivo, comprendente quelle che sono state analizzate o sperimentate nel corso dei mesi di studio da parte del candidato.

3.2.1. Sigmoid

Il *sigmoid*, chiamato anche funzione logistica, prende in ingresso un numero reale e lo riporta nell'intervallo [0,1]. Per questa sua caratteristica è anche usato negli ultimi nodi della rete neurale per predire probabilità. Il suo comportamento è tale per cui oltre una certa soglia di numeri positivi troppo grandi o negativi troppo piccoli rende rispettivamente 1 e 0.

La sua funzione è la seguente:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Si riporta di seguito nella Figura 7 la rappresentazione grafica del sigmoide e della sua derivata.

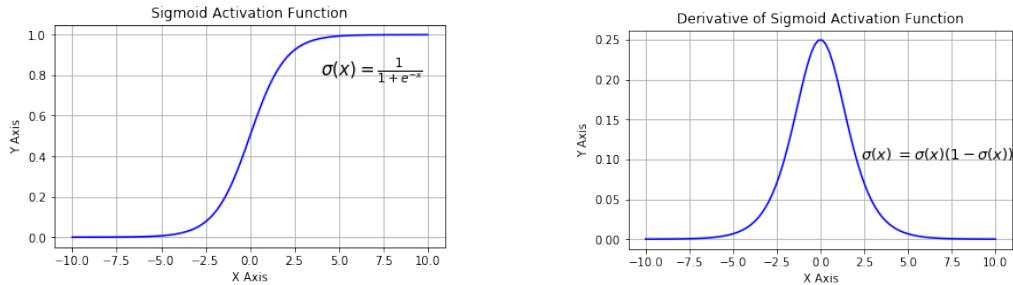


Figura 7 Grafico della funzione sigmoide e della sua derivata

Il sigmoid è caratterizzato dai problemi seguenti:

- **scomparsa del gradiente:** la funzione è troppo piatta in corrispondenza di valori vicini a zero o uno, questo comporta una difficoltà di aggiornamento dei pesi in presenza di valori di output nei bordi dell'intervallo 0,1
- **non è centrata in zero:** la distribuzione dei valori uscita di un sigmoide non è centrata in zero
- **computazionalmente inefficiente:** la funzione esponenziale è relativamente poco efficiente rispetto ad altre funzioni di attivazione non lineari.

3.2.2. Tanh

La funzione di attivazione Tanh deriva il suo nome dalla tangente iperbolica. In modo similare a sigmoid effettua una trasformazione dei valori in ingresso dal campo dei numeri reali a valori nell'intervallo [-1,1]. Questo comporta che a differenza del sigmoid i valori in uscita dal Tanh sono centrati in zero.

Rimane valido il problema della scomparsa del gradiente, in quanto come si vede dall'immagine seguente il comportamento è simile intorno ai bordi dell'intervallo.

Per la rappresentazione grafica della funzione tanh si veda la Figura 8.

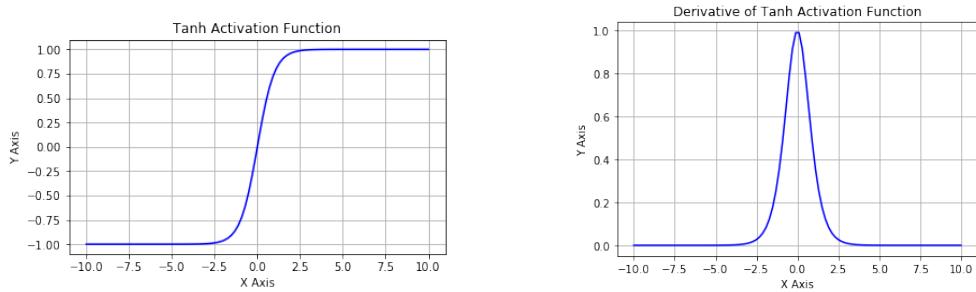


Figura 8 Grafico della funzione Tanh e della sua derivata

3.2.3. Rectified Linear Unit (ReLU)

La funzione Rectified Linear Unit è definita dall' equazione $y=\max(0,x)$, pertanto essa assume valore 0 per tutti i valori negativi ricevuti in ingresso, e rende invece il valore di ingresso se maggiore di zero.

Questo tipo di funzione, oltre ad essere computazionalmente molto efficiente, inoltre non risente del problema di scomparsa del gradiente. Come si può osservare dalla Figura 9 la forma di questa funzione non è curvilinea come le precedenti.

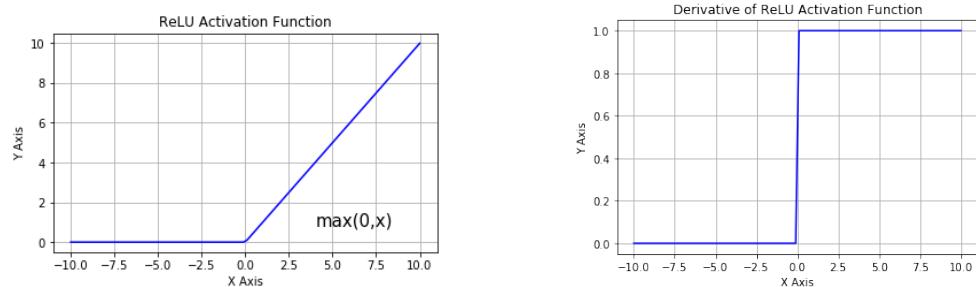


Figura 9 Grafico della funzione ReLU e della sua derivata

La funzione presenta tuttavia i seguenti problemi:

- non è centrata in zero
- rimane inattiva per tutti i valori negativi ricevuti in ingresso

Per risolvere quest'ultimo problema sono state introdotte delle varianti come Leaky ReLU, caratterizzata dalla funzione $y=\max(0.1*x, x)$ che mantiene una piccola parte di attivazione anche nella zona negativa, come si può osservare nell'immagine seguente.

3.2.4. SWISH

Negli ultimi mesi del 2017 i ricercatori di Google hanno diffuso una nuova funzione di attivazione chiamata SWISH, la cui funzione è la moltiplicazione di una funzione lineare con una logistica, ovvero $y = x \cdot \text{sigmoid}(x)$.

Nella loro pubblicazione [8], gli autori spiegano come dai test effettuati questa funzione sembra avere una performance superiore rispetto a ReLU.

La funzione SWISH è simile a ReLU nella parte positiva, ma è caratterizzata da una lieve inflessione vicino allo zero. È quindi una funzione smussata e non monotona. Si riporta nella Figura 10 una rappresentazione grafica dei valori assunti da questa funzione.

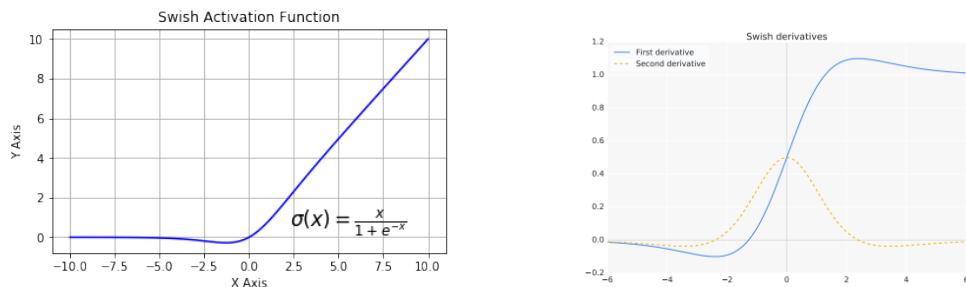


Figura 10 Grafico della funzione SWISH e della sua derivata

3.3. Reti Neurali feed-forward

Le reti neurali di tipo feed-forward, conosciute anche col nome Multi-layer Perceptrons, sono quelle più semplici in quanto sono caratterizzate dall'assenza di cicli e sono state le prime a diffondersi. In questo tipo di reti ogni output è funzione solamente dell'attuale input.

Questa tipologia di reti sono adatte per affrontare problemi di classificazione, un tipo di apprendimento supervisionato il compito è quello di dividere gli esempi forniti in gruppi predefiniti attraverso una funzione di decisione.

La funzione di decisione è appresa attraverso l'esame di una serie di esempi la cui classe è nota, questo processo viene chiamato training (addestramento).

Nella Figura 12 si può osservare a livello grafico la modalità di apprendimento, i punti sono gli elementi preclassificati del training set e le aree sono le regole di decisione apprese.

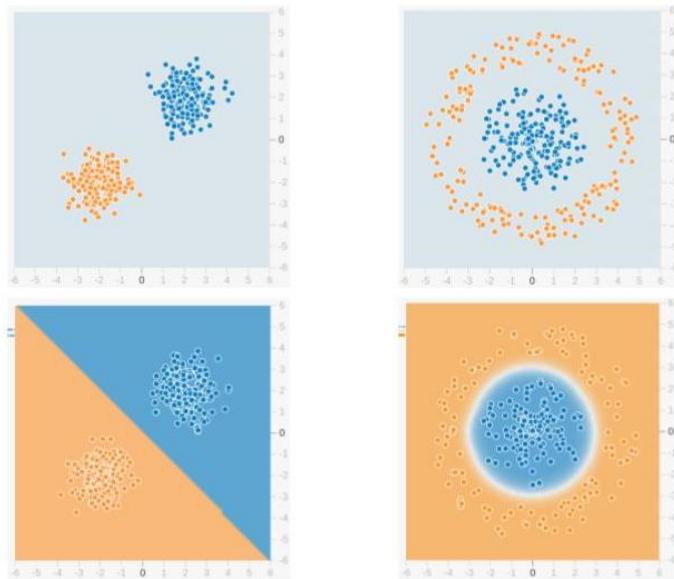


Figura 11 Esempi di classificazione

Le immagini sopra riportate sono state generate dal sito web playground.tensorflow.org², dove una pagina con finalità didattiche permette di muovere i primi passi affrontando 4 semplici casi di rete neurale. I dati di training sono quelli con il bordo bianco e quelli di test quelli con il bordo nero. Attraverso questa applicazione è intuitivamente possibile rendersi conto dell'utilità dei livelli nascosti. Nell'esempio di sinistra la funzione di decisione è lineare, in quanto è possibile separare le classi con una semplice retta, e in questo caso il modello riesce a convergere velocemente già dalla prima epoca anche senza l'utilizzo di livelli nascosti.

² I casi d'uso presentati possono essere riprodotti attraverso il link <http://playground.tensorflow.org/#activation=tanh&batchSize=10&dataset=circle®Dataset=reg-plane&learningRate=0.03®ularizationRate=0&noise=15&networkShape=4,4&seed=0.64055&showTestData=true&discretize=false&percTrainData=50&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false>

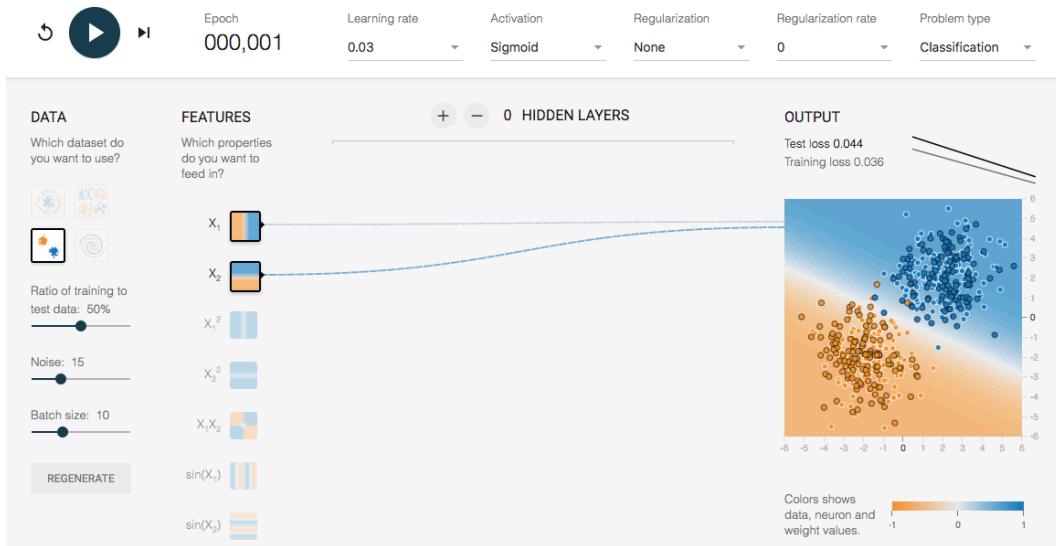


Figura 12 Esempio di training con funzione di decisione lineare e nessun livello nascosto

Nel secondo esempio invece la conformazione del dataset è tale per cui le classi non sono linearmente separabili, in quanto come si può osservare dalla Figura 13 sono costituite approssimativamente come una circonferenza circondata da un anello. In questo caso il modello senza hidden layers non riesce mai a convergere neanche dopo un numero molto elevato di epoche, e il loss rimane sopra il valore di 0,5 anche dopo 500 epoche. Questo perché non esiste una retta in grado di separare le due classi.

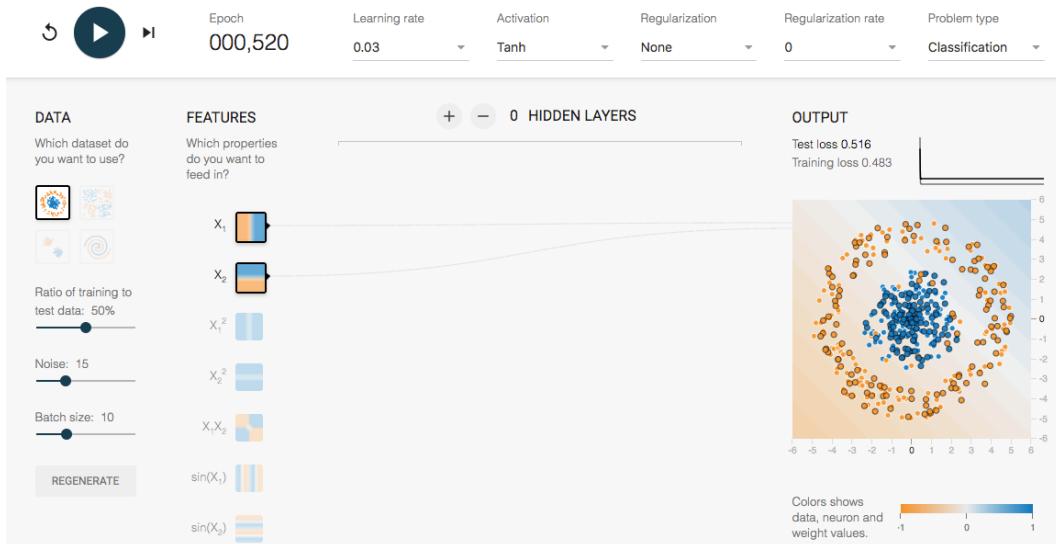


Figura 13 Esempio di training con funzione di decisione non lineare e nessun livello nascosto

Tuttavia è sufficiente aggiungere un livello nascosto con tre neuroni per ottenere un modello in grado di fare una predizione accettabile già dopo 60 epoche, dove viene minimizzata la funzione di loss sotto il valore di 0,1.

Osservando la Figura 14 è l'immagine di ogni riquadro riporta fornisce una rappresentazione grafica delle regole di classificazione di ciascun neurone; da ciò si può intuire il valore dei neuroni sia per quanto riguarda le feature di input che relativamente al livello nascosto.

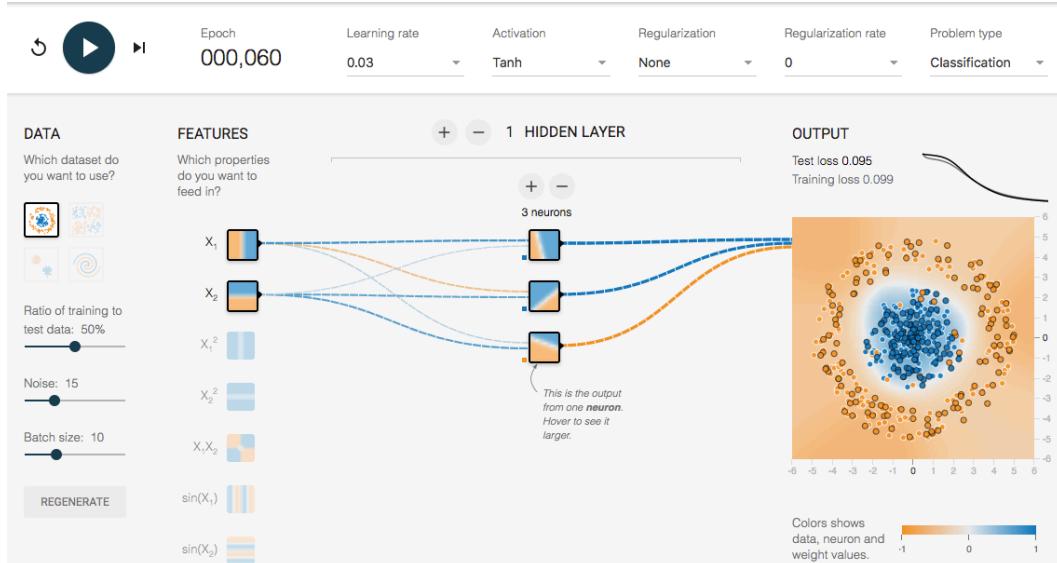


Figura 14 Esempio di training con funzione di decisione non lineare e 1 livello nascosto con 8 neuroni

Per migliorare ancora le performance di questa rete neurale di esempio si possono aggiungere neuroni al livello nascosto, che come si vede nell'immagine seguente riescono ad ottenere lo stesso risultato con circa 50 epoche.

Un'altra possibilità è quella di aggiungere più di un livello nascosto in cascata, e questa soluzione si dimostra migliore. Nell'esempio riportato di seguito nella Figura 15 si vede come a parità di numero di neuroni nascosti, la configurazione 2x4 (livelli x neuroni) sia migliore di quella 1x8, riuscendo ad abbattere la soglia di 0.1 nel loss in sole 30 epoche.

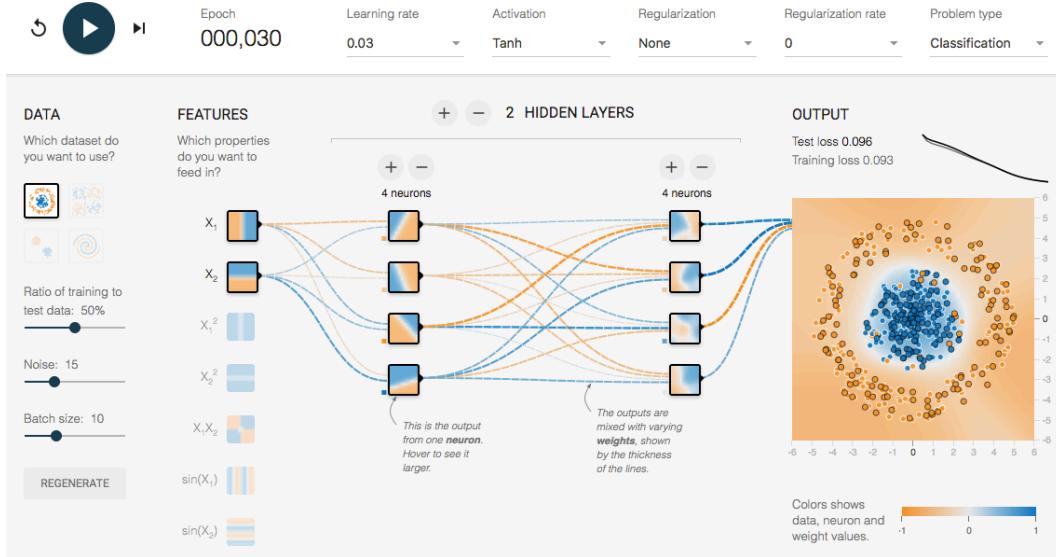


Figura 15 Esempio di training con funzione di decisione non lineare e 2 livello nascosto con 4 neuroni

3.3.1. *Overfitting e dropout*

Le reti neurali complesse contengono un certo numero di livelli nascosti non lineari, e questo permette loro di apprendere relazioni non banali tra i dati passati in input e i valori di output (valori target). Tuttavia quando il dataset non è sufficientemente grande si può verificare un eccessivo apprendimento tale per cui le regole apprese non danno poi risultati affidabili su dati di test o reali, in quanto il modello apprende anche dal rumore o dalle eccezioni del training set e non riesce a generalizzare con efficacia.

Questo fenomeno è definito overfitting, e viene affrontato con diverse tecniche tra cui l'interruzione del training quando le performance sul test set iniziano a degradare oppure l'applicazione di penalità di regolarizzazione L1 L2, ovvero l'aggiungere un termine aggiuntivo variabile alla funzione di costo.

La soluzione ideale nell'ipotesi di capacità computazionali infinite sarebbe quella di utilizzare una combinazione di svariati modelli differenti e addestrare questi modelli su una mole di dati di training molto alta; tuttavia queste due ipotesi sono spesso irrealistiche.

La tecnica del dropout cerca di soddisfare questi due principi, spegnendo ad ogni step di apprendimento alcuni dei neuroni nascosti o visibili scelti in modo

causale. Nella Figura 16 sono indicati con una croce i neuroni disattivati, i cui pesi quindi non contribuiscono alla predizione dell'output corrente.

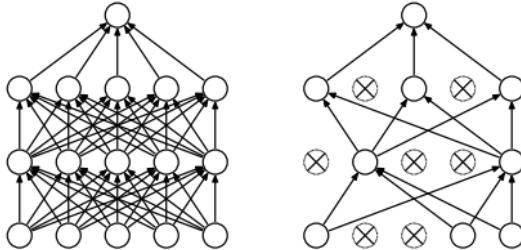


Figura 16 Una rete neurale standard (a sinistra), e a seguito dell'applicazione del dropout

La scelta di quali neuroni disattivare è causale, nel caso più semplice è prevista per ciascuno di essi una probabilità p indipendente dalle altre di essere disattivato. Solitamente questa probabilità è vicina al valore di 0.5 per i nodi interni e molto più bassa per i nodi di input per evitare perdita di informazione. Nella fase di test la rete viene invece consultata con tutti i neuroni attivati e il valore di p viene utilizzato per moltiplicare il valore in uscita. La Figura 17 sintetizza il diverso comportamento del neurone nella fase di addestramento e in quella di test.

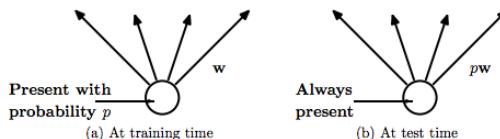


Figura 17 Probabilità p di dropout per singolo neurone

L'applicazione del dropout a una rete neurale di n nodi si può quindi vedere come il training di una potenziale collezione di 2^n modelli ridotti diversi. La moltiplicazione del peso w di ciascun nodo per una probabilità p può essere vista come il merge di questi modelli.

3.3.1. Overfitting e regolarizzazione

Un'altra tecnica per contrastare l'overfitting del modello sui dati di train è quella di applicare la regolarizzazione. Questa opera ponendo dei limiti al valore dei pesi dei neuroni, come se si volesse indicare di avere una fiducia limitata nel training set.

In termini matematici la regolarizzazione consiste nell'inserire un termine aggiuntivo alla funzione di costo, in modo che sia penalizzante per alcune configurazioni. Ad esempio, data la funzione di costo, dove x sono i valori di input, w i pesi e y i valori di output:

$$S(w, X) = - \sum_i \log P(Y = y_i | x_i; w)$$

si ottiene una nuova funzione sommando a quest'ultima una funzione regolarizzante che prende in ingresso il vettore dei pesi:

$$E(w, X) = S(w, X) + \lambda R(w)$$

Una funzione regolarizzante molto diffusa è la seguente:

$$R(w) = \left(\sum_i |w_i|^p \right)^{1/p}$$

dove in base al valore di p 1 o 2 si parla di regolarizzazione L1 o L2.

La funzione di regolarizzazione L1, chiamata anche LASSO Regression (Least Absolute Shrinkage and Selection Operator) come termine di penalità una grandezza in valore assoluto.

La funzione di regolarizzazione L2, chiamate anche Ridge Regression, aggiunge un termine di penalità una grandezza quadratica.

In entrambi i casi se lambda è uguale a zero significa che non viene fatta nessuna regolarizzazione.

3.3.2. Tipi algoritmi di ottimizzazione

Gli algoritmi di ottimizzazione, detti anche optimizer, sono delle funzioni che aiutano all'apprendimento riducendo la funzione di errore e quindi il loss del modello durante la sua fase di training. È naturale che la strategia di ottimizzazione che viene utilizzata per aggiornare i vettori W (pesi) e b (bias) del modello rivesta un ruolo molto importante.

Questi algoritmi si dividono in due categorie:

- Algoritmi di ottimizzazione di primo ordine, basati sul gradiente e sulla derivata di primo ordine
- Algoritmi di ottimizzazione di secondo ordine, basati sulla derivata di secondo ordine

3.4. Reti Neurali Convoluzionali

Le reti neurali convoluzionali (CNN) sono una sottoclasse delle reti feed forward, solitamente utilizzate per l'analisi di immagini. Le CNN sono ispirate al funzionamento della corteccia cerebrale visiva, dove i neuroni corticali rispondono solo in una regione ristretta del campo visivo, chiamata campo recettivo. I diversi neuroni lavorano in campi recettivi differenti con una sovrapposizione parziale degli stessi e il campo visivo viene poi ricostruito per aggregazione di tutti i segnali raccolti.

Nella Figura 18 è riportata la struttura di un blocco convoluzionale, divisa nelle seguenti categorie:

- Livelli nascosti, con il filtro convoluzionale e pooling;
- Livelli di classificazione con un flatten layer, fully connected layer e loss layer.

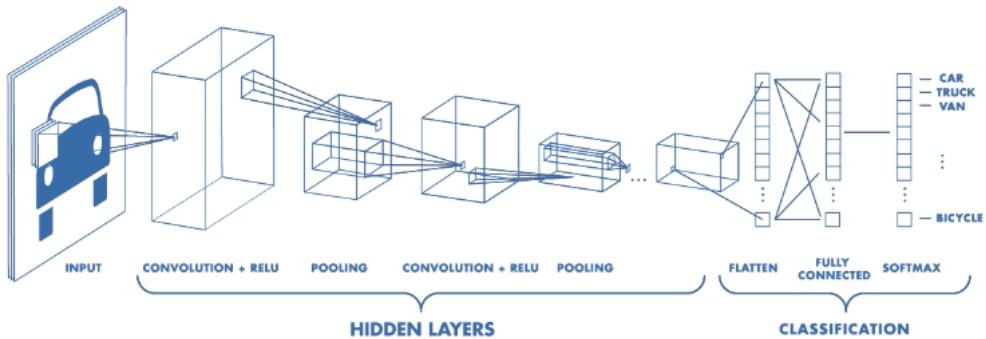


Figura 18 Struttura di una rete convoluzionale

3.4.1. Filtro Convoluzionale

La convoluzione è un'operazione importante nella processazione di segnali (vettori a una dimensione) e immagini (matrici a due dimensioni) e si può definire come l'applicazione di un filtro (detto anche kernel) al dato di input.

Nel caso di una dimensione, dato un vettore di input f di lunghezza n e un kernel g di lunghezza m dispari, la convoluzione $f * g$ di f si può definire come:

$$(f * g)(i) = \sum_{j=1}^m g(j) \cdot f(i - j + m/2)$$

Ad esempio con un vettore di input

$$f = [10 \ 50 \ 60 \ 10 \ 20 \ 40 \ 30]$$

ed un kernel

$$g = [1/3 \ 1/3 \ 1/3]$$

se voglio calcolare il vettore risultato h devo shiftare il vettore g su f , centrando g rispetto ad ogni elemento di f . Vado quindi ad applicare la moltiplicazione con il corrispondente elemento del kernel e poi sommo tutti valori per ottenere $h(i)$. Come si vede si può comprendere meglio nella rappresentazione seguente, $h(3)$ sarà uguale a $50/3+60/3+10/3 = 40$

10	50	60	10	20	30	40
0	1/3	1/3	1/3	0	0	0

Nel caso di kernel che sfiora rispetto a f ignoro semplicemente gli elementi di g che non sono sovrapposti ad un elemento di f . Quindi per calcolare $h(1)$ ignorerò il primo elemento di g e utilizzo solo gli altri due, quindi $10/30+50/3=20$.

L'applicazione su ogni elemento di f produce l'output h seguente:

$$h = [20 \ 40 \ 40 \ 30 \ 20 \ 30 \ 23.333]$$

L'estensione al caso a due dimensioni è immediata, con l'estensione sia di f che di g a due dimensioni. In questo caso il filtro viene fatto scorrere pixel per pixel, riportando il risultato nel rispettivo pixel di h . Nell'ipotesi di applicazione di un kernel g di dimensione 3×3 si può comprendere meglio i vari significati che l'operazione di convoluzione assume in base alla conformazione della maschera. Ad esempio nella Figura 19 viene assegnato il valore $1/9$ a ciascuno dei 9 elementi di g , con il risultato di produrre un'immagine simile sfuocata, in quanto ogni pixel dell'output $h(i)$ mescola l'informazione del pixel originario $f(i)$ con gli altri 8 pixel adiacenti comprese le diagonali.

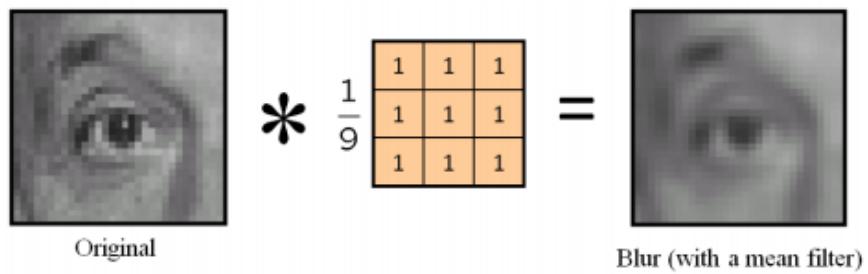


Figura 19 Filtro convoluzionale 2D con sfuocatura

Valorizzando solo alcuni dei valori nei bordi di g si possono compiere operazioni di traslazione dell'immagine, come nell'esempio seguente in Figura 20 dove viene effettuata lo spostamento a sinistra di un pixel.

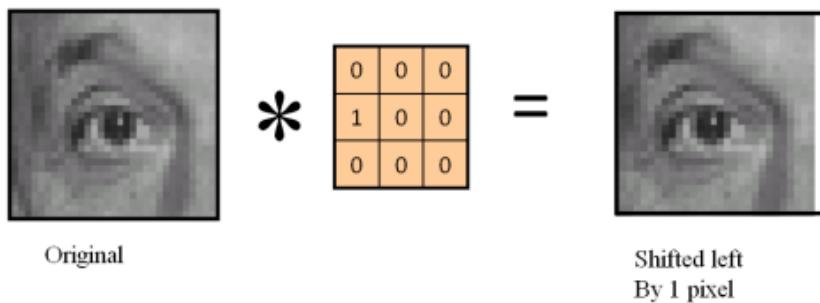


Figura 20 Esempio di Filtro convoluzionale 2D con shift a sinistra dell'immagine

3.4.2. Pooling

I pooling layer sono utilizzate per ridurre le dimensioni del dato in input e quindi ridurre i tempi di addestramento del modello. In estrema sintesi questa operazione consiste nel raggruppare i dati in blocchi e scegliere per ciascun blocco un valore rappresentativo.

Nell'esempio seguente rappresentato dalla Figura 21 si riporta l'operazione di Max Pooling, dove il valore scelto per rappresentare il blocco è quello maggiore. Si supponga di avere in input una matrice a due dimensioni 4x4 che si vuole ridurre a una matrice due per due. La prima operazione è quella di suddividere idealmente la matrice in quattro sotto blocchi, dopodiché si sceglie da ciascun blocco il valore più grande.

La matrice 2x2 risultante avrà quindi il valore massimo di ogni sotto blocco, ovvero 9 per il blocco viola, 2 per il blocco azzurro, 6 per il blocco verde e 3 per il blocco rosso.

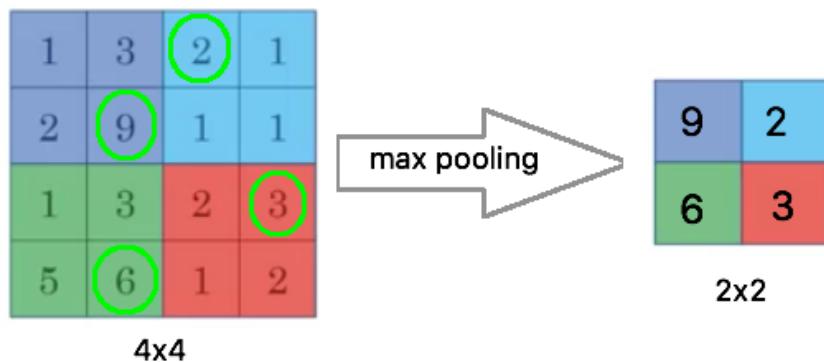


Figura 21 Rappresentazione dell'operazione di Max pooling

Nel caso di Average Pooling varia solamente l'operazione di scelta del rappresentante, questa volta individuato nella media dei valori del sotto blocco. Facendo riferimento all'immagine sopra, avrei avuto nel blocco a destra i valori 3.75, 1.25, 4,2.

L'operazione di pooling è definita dai seguenti due hiperparametri:

- f: pool_size una coppia di interi che indica la dimensione della finestra utilizzata per individuare i sottoblocchi;
- s: strides, ovvero una coppia di interi che indica lo scivolamento della finestra.

Quando questi due valori corrispondono, come nel caso precedente non si ha sovrapposizione. Si riporta di seguito un esempio con sovrapposizione, dove si applica un pooling Layer ad un input di forma 5x5 per produrre un output 3x3.

In questo avremo la dimensione f valorizzata a (3,3), e lo spostamento s valorizzato a (1,1). In questo caso la finestra avanza di una casella per volta e il numero 9 nella posizione 2,2 viene ad esempio ripetuto quattro volte in quanto compare in quattro sottoblocchi. La Figura 22 mostra in maniera grafica lo scorrimento delle finestre di pooling con sovrapposizione, difatti le celle centrali più utilizzate sono raffigurate con un colore verde via via più scuro.

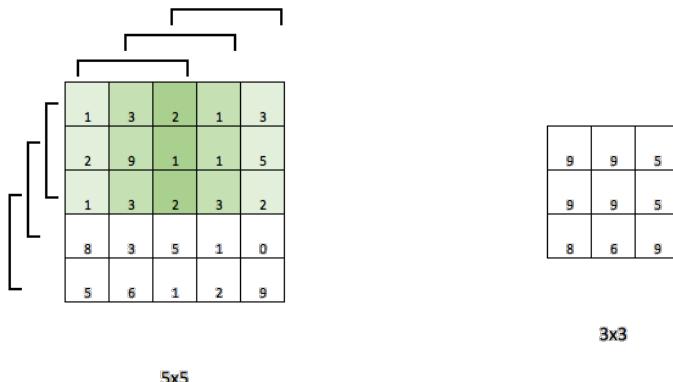


Figura 22 Esempio di Max Pooling con pool_size 3x3 e strides 1x1

3.4.3. *Livelli di classificazione*

I livelli di classificazione si occupano di eseguire il comportamento descritto nella sezione dedicata alle reti feed forward.

Il flatten layer è necessario in quanto l'output in uscita dai livelli nascosti è multi dimensionale e deve essere ridotto a una dimensione per poter essere utilizzato. Il fully connected layer effettua le operazioni lineari classiche di aggiustamento pesi e il loss layer applica una funzione di attivazione come sigmoid, nel caso di una o due classi, o softmax nel caso di più classi

3.5. Residual Network ResNet

Le Residual Network (ResNet) sono una categoria di CNN recentemente diffusasi a partire dagli studi di alcuni ricercatori Microsoft che nel loro lavoro di ricerca [9] che sono riusciti con questa nuova tipologia di rete a vincere le competizioni Image Net e COCO 2015.

La profondità di una rete neurale è molto importante per la sua performance, ma purtroppo al crescere della profondità diventa più difficile effettuare l'addestramento, in quanto l'accuratezza non riesce a convergere ma alterna rapide salite a rapidi decadimenti. La soluzione proposta con le ResNet è quella di inserire delle scorciatoie o skip connections per collegare i livelli più profondi con quelli terminali. Nella Figura 23 si può osservare come il ramo a destra riporti una copia del valore di input x nell'livello più avanzato.

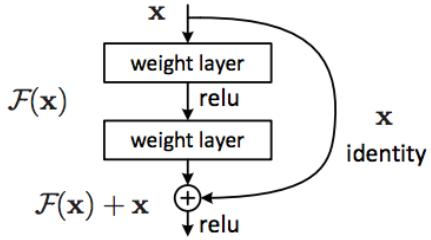


Figura 23 Un esempio di blocco di Residual Network

A livello intuitivo il motivo del buon funzionamento delle ResNet si può attribuire a due aspetti:

- Nelle fasi iniziali dell'apprendimento la gestione dei pesi si concentrava troppo nei livelli iniziali, per poi espandersi gradualmente ai livelli più terminali. In questo caso fornire in avanti una copia dell'input velocizza l'apprendimento
- Durante le fasi più avanzate dell'apprendimento quando tutti i livelli sono ben valorizzati, una rete senza la parte residuale proverebbe ad esplorare maggiormente lo spazio delle feature, generando instabilità. Con le skip connection si hanno invece dei riferimenti che evitano questa eccessiva variabilità dei risultati.

3.6. Panoramica delle Reti Neurali ricorrenti (RNN)

Le reti neurali ricorrenti (RNN) sono una classe di reti neurali che hanno la capacità di analizzare sequenze temporali con un comportamento dinamico nel tempo. A differenza delle reti feed-forward possiedono uno stato di memoria interna che viene utilizzato insieme agli input per generare i valori di output.

Il comportamento delle RNN è molto più simile a quello della mente umana, che non inizia a pensare da zero a ogni istante, ma interpreta ogni parola letta o ascoltata sulla base del contesto e della sua esperienza precedente.

Ad esempio in lingua italiana la parola sole può essere riferita sia alla stella più vicina alla terra che ad un insieme di persone che non hanno compagnia, a seconda del contesto della frase e del suo caratterizzarsi come sostantivo o aggettivo.

Per questo motivo sono la architettura di rete neurale più naturale per analizzare sequenze e liste, e per quanto riguarda il riconoscimento vocale o della scrittura manuale.

La particolarità di queste reti è quella di possedere un loop, che permette all'informazione di persistere da uno step all'altro. Nella Figura 24 è mostrata una rappresentazione sintetica di RNN con il loop rappresentato dalla freccia che rientra nello stesso blocco A.

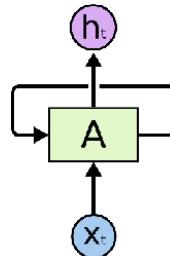


Figura 24 Immagine schematica di una cella LSTM

Una RNN può essere vista come una serie di copie della stessa rete neurale, con ciascuna di queste copie che trasmette un input alla copia successiva.

Nella seguente Figura 25 viene schematizzato il comportamento di base di una RNN: X, O sono gli input e output (X_t , O_t i valori ad ogni istante t_i), U, V, W sono rispettivamente lo stato di input, lo stato nascosto, lo stato di output.

Ad ogni istante t , la RNN riceve un input X_t , che viene utilizzato sia per modificare gli stati U, V, W sia per generare l'output O_t assieme agli stati.

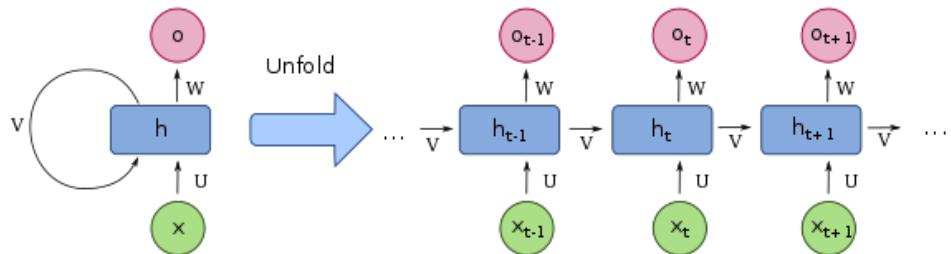


Figura 25 Diagramma di una rete neurale ricorrente (RNN)

Le RNN sono distinguibili in sottocategorie come:

- Fully Recurrent: ogni nodo è collegato direttamente con tutti gli altri nodi;

- Independently recurrent (RNN);
- Recursive neural network;
- Hopfield Network;
- Elman Network e Jordan Network;
- Echo State;
- Neural history compressor;
- Long short-term memory.

3.7. Long Short Therm Memory (LSTM)

La LSTM è una tipologia particolare di RNN che è molto più efficiente della versione standard. Le RNN tradizionali funzionano molto bene quando la distanza tra le informazioni rilevanti e l'elemento corrente è abbastanza piccola, tuttavia le loro prestazioni degradano velocemente al crescere di questa distanza. La struttura di base di una RNN, standard è piuttosto semplice, in pochi termini si può dire che lo stato precedente e l'input corrente X_t vengono combinati in layer tanh. Si veda la Figura 26, dove la freccia proveniente dalla cella a sinistra rappresenta lo stato precedente, combinato con la funzione tanh.

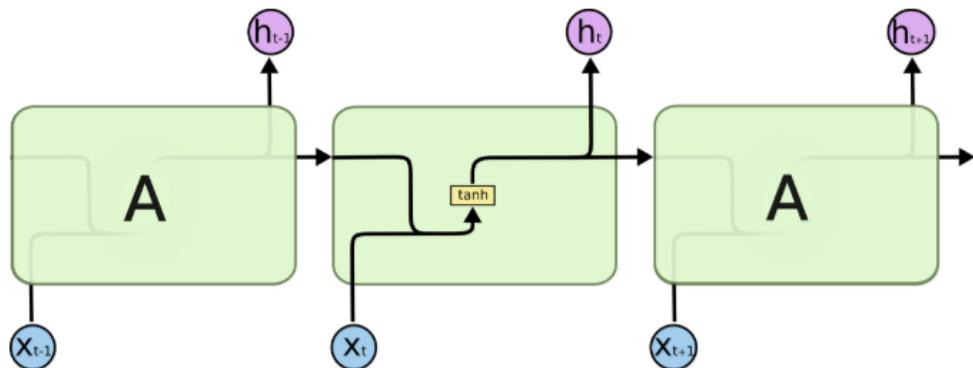


Figura 26 Combinazione di input e stato precedente in una RNN

In relazione alle dipendenze lontane in linea teorica tutte le RNN dovrebbero essere in grado di apprendere, ma nella pratica questo non si verifica. Le LSTM sono disegnate per evitare questo tipo di problema, grazie alla particolare struttura del modulo che contempla 4 layer anziché uno solo.

Nell'immagine riportata in Figura 27 i blocchi gialli rappresentano i layer, i cerchi rossi le operazioni puntali, le frecce il trasferimento dei vettori, la concatenazione e la duplicazione dei valori.

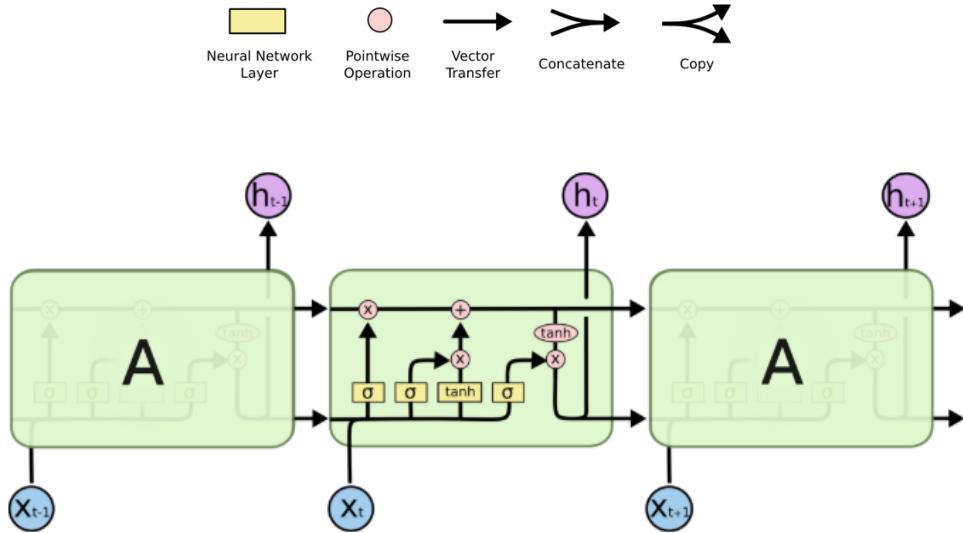


Figura 27 Struttura di una cella LSTM

La parte fondamentale di una LSTM è il suo stato, rappresentato nella Figura 28 dalla linea orizzontale in alto, che percorre tutta la catena con solo qualche interazione lineare minore.

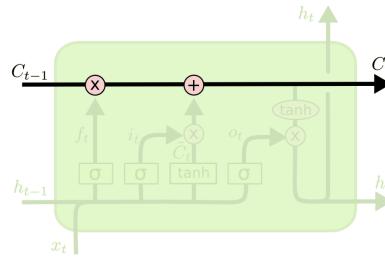


Figura 28 Stato di una cella LSTM

Per permettere alla cella di rimuovere o aggiungere informazione allo stato della cella sono presenti tre porte composte da un layer sigmoid seguito da una moltiplicazione puntuale. La funzione di attivazione sigmoidea ha in uscita valori nell'intervallo [0,1], dove lo zero significa "non fa passare nulla" e uno "fai passare tutto". Si osservi in merito la Figura 29.

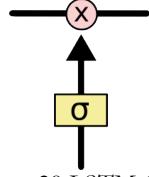


Figura 29 LSTM Gate

Si riportano ora passo per passo le operazioni svolte dalla cella LSTM.

Il primo passo è decidere cosa dimenticare, e questo viene fatto con la prima porta chiamata anche *forget gate layer*. Questa prende in esame h_{t-1} e x_t , ovvero l'output del ciclo precedente e l'input del ciclo corrente, ed estrarre un valore di tra zero e uno per ogni cifra nello stato precedente della cella C_{t-1} . La Figura 30 mostra una rappresentazione grafica corredata dalla rispettiva funzione analitica. Si ricorda che i vettori W e b sono rispettivamente i pesi e il bias che caratterizzano questo componente

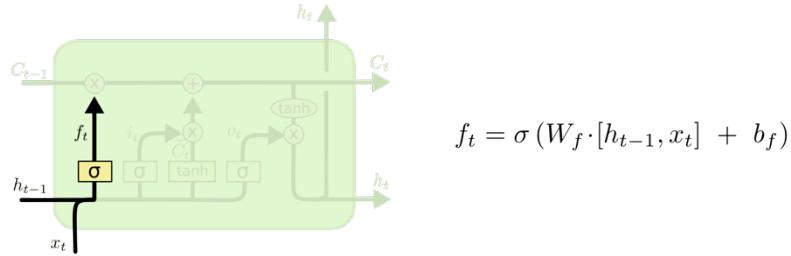


Figura 30 LSTM, forget gate layer

Il secondo passo è quello di decidere cosa ricordare delle informazioni nuove, grazie alla seconda porta chiamata *input gate layer* combinata con layer tanh. Questa operazione produce un vettore di candidato a nuovo stato \tilde{C}_t . La figura Figura 31 riporta graficamente e analiticamente questo sottoblocco.

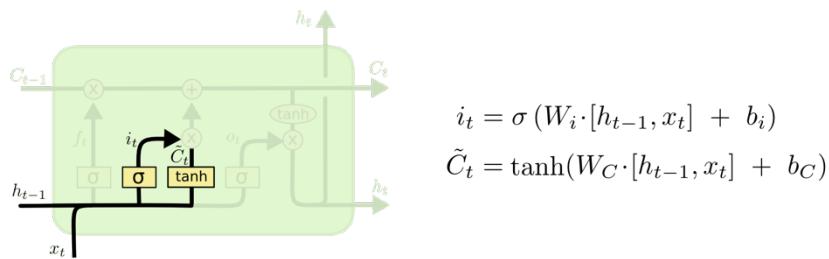


Figura 31 LSTM, input gate layer

A questo punto si aggiorna lo stato della cella nel nuovo stato C_t , prima moltiplicando lo stato precedente per l'esito della prima porta, f_t , e poi sommando il risultato della seconda porta, $i_t * \tilde{C}_t$. Nella figura Figura 32 è mostrata l'operazione di aggiornamento di stato.

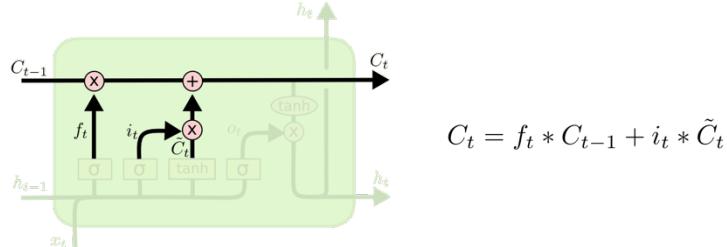


Figura 32 LSTM, generazione del nuovo stato C_t

Infine dopo aver aggiornato lo stato si decide cosa restituire come output. Questa operazione viene eseguita moltiplicando tra loro il terzo layer sigmoid e il nuovo stato corrente filtrato da una funzione tanh. Si veda l'ultima immagine della serie nella Figura 33.

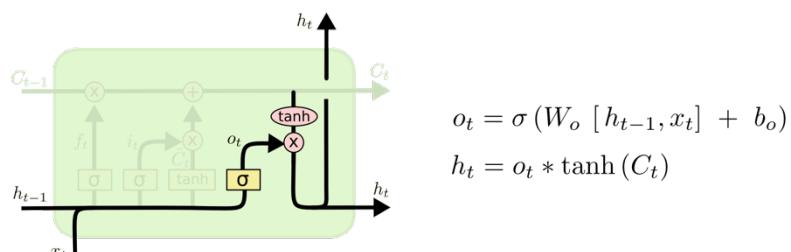


Figura 33 LSTM, definizione dell'output

Una dei limiti principali della LSTM e in generale delle RNN descritte finora è quello di essere in grado di imparare solo attraverso l'analisi dei passi precedenti. Tuttavia qualche volta può avere senso cercare di imparare anche dalle sequenze future, quando sono già note, al fine di capire meglio il contesto e ridurre l'ambiguità. Questo è esattamente il comportamento di una famosa variante delle RNN, le bidirectional RNN (BRNN).

Queste reti si compongono di due tipi di connessioni, verso il passato e verso il futuro. L'addestramento della rete viene eseguito facendo scorrere il dataset prima dall'inizio alla fine, e poi in senso inverso. Questo meccanismo è definito

nella Figura 34, si noti come questo tipo rete genera ad ogni istante due uscite, che possono poi venir combinate con metodi quali, somma, media, massimo, concatenazione.

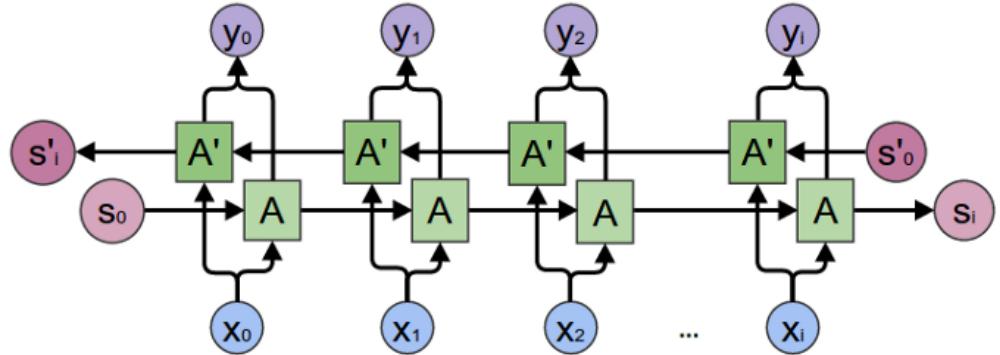


Figura 34 LSTM bidirezionale

4. Panoramica delle principali librerie e framework per il Data Science in linguaggio Python

In questa sezione vengono dapprima elencate alcune importanti librerie di base per la gestione di array multidimensionali, e di tutte le attività a corredo del machine learning, per poi proseguire con tre dei più noti framework per l'implementazione di reti neurali in linguaggio Python. La trattazione non ha la pretesa né l'obiettivo di fornire una panoramica completa dello stato dell'arte, ma si concentra su quelli che sono gli strumenti usati in tutti gli esempi pratici analizzati o prodotti dallo studente nell'ambito della sua attività di studio relativa alla Computing in Cardiology Challenge.

In un articolo [10] pubblicato sulla piattaforma Medium nel mese di giugno 2018, gli strumenti open source oggetto di questa tesi risultano comunque tra i più popolari per numero di commit e numero di collaboratori, come viene riepilogato nella Tabella 6:

Tabella 6 Dimensioni comunità open source per librerie di Data Science

Categoria	Libreria	Commits	Contributors
Core Libraries e Statistica	NumPy	17.911	641
Core Libraries e Statistica	SciPy	19.150	608
Core Libraries e Statistica	Pandas	17.144	1165
Virtualizzazione	Matplotlib	25.747	725
Machine Learning	Scikit-learn	22.753	1084
Deep Learning	TensorFlow	33.339	1469
Deep Learning	Keras (ora parte di Tensorflow)	4.539	671

4.1. Le librerie NumPy, SciPy, pandas, matplotlib

In questa sezione vengono descritte brevemente alcune delle librerie di base per il calcolo matematico che vengono ad oggi utilizzate nella maggior parte dei progetti di computazione scientifica.

Inizialmente il linguaggio Python non era predisposto per supportare a pieno l'analisi numerica, tuttavia esso attirava l'attenzione della comunità scientifica per la sua semplicità nella sintassi, ed il suo essere gratuito e open source. Questo ha portato alla progressiva produzione di La svolta nell'introduzione di queste librerie è stata quella di offrire un'ampia gamma di funzioni e operazioni su matrici tipiche di linguaggi come MATLAB.

4.1.1. La libreria NumPy

La libreria NumPy è utilissima estensione del linguaggio Python rilasciata con licenza open-source a partire dall'anno 2006. Questa libreria offre delle API veramente efficienti la gestione di array e matrici multidimensionali di grandi dimensioni, ed offre una vasta gamma di operazioni matematiche ad alto livello. Il gruppo matrix-sign diede alla luce una prima versione delle librerie sotto il nome di Numeric, rinominate 10 anni dopo in NumPy.

Il cuore di questa libreria è la definizione della struttura dati ndarray, ovvero array n-dimensionale. La caratteristica di questo tipo di dati è che tutti gli elementi di questo array devono essere dello stesso tipo.

Si riporta una breve descrizione delle funzioni più note e che verranno utilizzate nella descrizione dei progetti durante le sezioni successive.

- `numpy.array(object, dtype=None)`: permette di creare un array con la semplice enumerazione dei suoi elementi. Es: `numpy.array([1,2,3], float)`
- `numpy.zeros, numpy.ones`: permette di inizializzare un array con tutti i valori rispettivamente a 0 o 1. Es: `numpy.zeros(2,2)` genera una matrice 2x2 con tutti i valori a zero.
- `numpy.ravel(object, order ='C')`: prende in ingresso un array multidimensionale e lo trasforma in un array piatto, per default secondo l'ordinamento riga, colonna.

- `numpy.reshape(a,newshape)`: effettua il ridimensionamento di un array, senza che venga modificato il numero complessivo di elementi: Es: trasformare un array a due dimensioni di forma 5,2 in un array a tre dimensioni di forma 5,2,1

4.1.2. La libreria SciPy

Questa libreria³ Python, rilasciata con licenza free e open source, contiene vari moduli per ottimizzazione, calcoli di algebra lineare, calcolo di integrali, risoluzione di equazioni differenziali, interpolazione, processazione di immagini e di segnali. Il suo primo rilascio è stato nell'anno 2011, ed attualmente è disponibile nella versione 1.1.0.

I moduli di questa libreria sono composti dai seguenti sub-package:

- **constants**: physical constants and conversion factors (since version 0.7.0)
- **cluster**: hierarchical clustering, vector quantization, K-means
- **fftpack**: Discrete Fourier Transform algorithms
- **integrate**: numerical integration routines
- **interpolate**: interpolation tools
- **io**: data input and output
- **lib**: Python wrappers to external libraries
- **linalg**: linear algebra routines
- **misc**: miscellaneous utilities (e.g. image reading/writing)
- **ndimage**: various functions for multi-dimensional image processing
- **optimize**: optimization algorithms including linear programming
- **signal**: signal processing tools
- **sparse**: sparse matrix and related algorithms
- **spatial**: KD-trees, nearest neighbors, distance functions
- **special**: special functions

³ SciPy Repository su github in <https://github.com/scipy/scipy>

- **stats**: statistical functions
- **weave**: tool for writing C/C++ code as Python multiline strings

In particolare nel progetto viene fatto uso del modulo `scipy.io` per l'acquisizione di segnali in formato MATLAB .mat V4 (tracciato arousals).

4.1.3. La libreria Pandas

La libreria Pandas⁴, acronimo che derivato dalle parole Panel Data, è una libreria Python open source che offre delle semplici strutture dati e algoritmi utilizzate principalmente in fase di acquisizione e preparazione dei dati.

Il cuore di questa libreria è l'oggetto `DataFrame`, una struttura dati a due dimensioni ben indicizzata sia nelle righe che nelle colonne. Questa struttura rende molto espressivo ed elegante il lavorare con dati tabellari di tipo relazionale o etichettato.

Nell'esempio seguente si mostra la creazione di una matrice con valori random, l'estrazione di una riga con indice numerico e infine l'estrazione di una colonna con l'indicazione della lettera, quindi come array associativo.

```
import pandas as pd
import numpy as np

# array numerico da 2 a 4
rows = np.arange(2,4)
# lista di caratteri da usare come colonne
cols = list('ABCDF')
df = pd.DataFrame(np.random.randn(3,5),index=rows,columns=cols)

# estraggo le righe 2 e 3
rows_label23 = df.loc[2:3]
#estratto la colonna D
column_labelD = df['D']
```

⁴ <https://pandas.pydata.org/>

4.1.4. *La libreria Matplotlib*

La libreria grafica Matplotlib⁵ fornisce una serie di API orientate agli oggetti per la creazione di grafici. Il suo primo rilascio è avvenuto nell'anno 2015 ed attualmente si trova nella versione stabile 2.2.3 con licenza di tipo open source. La sua interfaccia è progettata per essere semplice e usabile, con affinità rispetto al linguaggio MATLAB.

Si riporta di seguito nella Figura 35 un esempio elementare di diagramma a linee di una funzione esponenziale. il metodo plot si occupa di disegnare il diagramma e il metodo show avvia il visualizzatore.

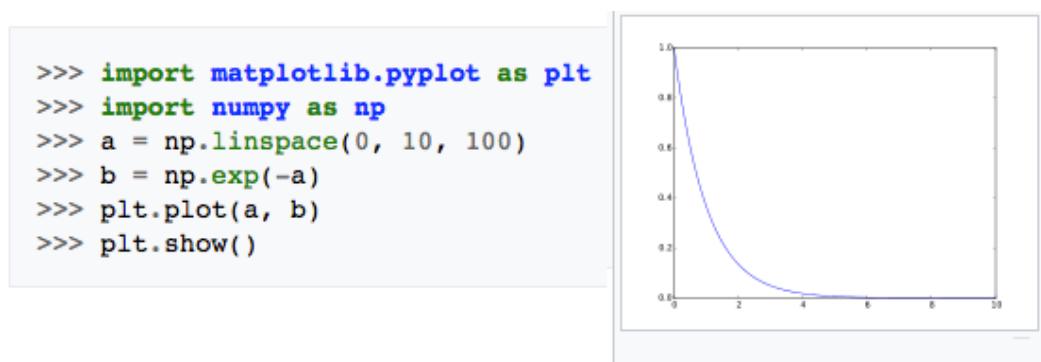


Figura 35 Esempio di utilizzo della libreria matplotlib su una funzione esponenziale

4.2. Scikit-learn

Scikit-learn⁶ è una libreria di machine learning gratuita scritta in linguaggio Python. La sua prima versione è stata rilasciata nell'anno 2011 ed attualmente è disponibile nella versione 0.19.2. Il progetto ha tra i suoi sponsor INRIA⁷, Google e la Python Software Foundation.

Contiene una vasta gamma di funzionalità nell'ambito dell'apprendimento supervisionato e non supervisionato. In particolare per quanto riguarda la classificazione e la regressione offre algoritmi come Support Vector Machine, Nearest neighbors, Random forest, Ridge Regression, e Lasso. Nell'area dell'apprendimento non supervisionato offre algoritmi di clustering come K-means, Spectral clustering e Mean Shift.

⁵ <https://matplotlib.org/>

⁶ <http://scikit-learn.org/stable/>

⁷ Istituto di ricerca francese dal nome Institut National de Recherche en Informatique et en Automatique)

Questa libreria si caratterizza come a più alto livello rispetto a Tensorflow, costituendo un'interfaccia semplificata per costruire modelli di machine learning standard. Questa caratteristica fa sì che questa libreria sia uno strumento facile da utilizzare, ma meno personalizzabile rispetto ad altri.

Inoltre tra i suoi svantaggi c'è quello di non avere una versione compilata per l'utilizzo delle GPU, che oggi offrono la possibilità di eseguire calcoli complessi su grandi dataset con velocità di molto superiori a qualsiasi CPU presente sul mercato.

Come esempio si descrivere l'implementazione di un classificatore logistico, utilizzato nel modello di esempio della Physionet CinC Challenge 2018.

Il modello LogisticRegression è un modello lineare di classificazione noto anche come logit regression, Maximum-entropy classification (MaxEnt) o log-linear classifier. In questo modello le probabilità del verificarsi di un evento di tipo dicotomico (definibile con una variabile dal valore 0 o 1) sono calcolate con una funzione logistica.

Si definisce la funzione logistica come una funzione sigmoidea, ovvero con andamento a S, con la seguente formula:

$$P(y) = \frac{1}{1 + e^{-y}}$$

L'andamento della funzione logit è rappresentato nella Figura 36.

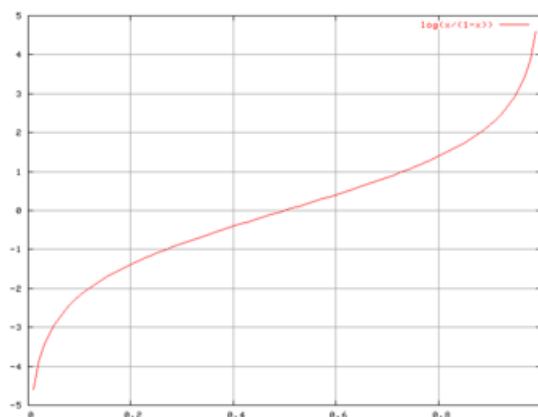


Figura 36 Andamento della funzione sigmoidea logit

Nel caso in cui questo modello venga applicato a più di due classi viene utilizzata la strategia one-vs-rest, ovvero quella di addestrare un classificatore separato per ogni classe che predice una probabilità compresa tra 0 e 1 di quella classe.

Nell'esempio seguente si riporta un esempio base di utilizzo di questo classificatore. Il caso d'uso è quello di generare variabili casuali di tipo blob suddivise in due classi, e quindi testare il funzionamento di base del classificatore. Le variabili di ingresso posso essere paragonate agli esempi di base esempio riportati nella sezione 3.3.

- La funzione *make_blobs* del modulo *sample_generator* genera un dataset di training con due classi;
- Viene istanziato il modello Logistic Regression;
- Il dataset viene passato al modello per addestramento con il metodo *fit*;
- Viene generato il dataset di training con lo stesso metodo e viene avviata la predizione delle classi con il metodo *predict*;
- A questo punto l'array *y_predict* può essere stampato su console o file oppure passato ad una funzione di punteggio per una sua valutazione secondo l'indicatore desiderato.

```
from sklearn.linear_model import LogisticRegression
from sklearn.datasets.samples_generator import make_blobs
# generate 2d classification dataset
X, y = make_blobs(n_samples=100, centers=2, n_features=2, random_state=1)
# fit final model
model = LogisticRegression()
model.fit(X, y)

# generate 2d test dataset
X_test, y_test = make_blobs(n_samples=20, centers=2, n_features=2, random_state=1
#predict values
y_prediction = model.predict(X_test)
#user defined function to score the classifier
score = scorePrediction(y_test, y_prediction)
```

4.3. TensorFlow

TensorFlow⁸ è una libreria software open source per il deep learning ottimizzata per il calcolo numerico ad altissime prestazioni. Questo prodotto è stato sviluppato inizialmente dal team Google Brain e è impiegato in tanti prodotti commerciali Google ad esempio nel riconoscimento vocale, in Gmail, in Google Foto. Attualmente è rilasciato con versione Open Source e nella versione 1.9.0.

4.3.1. I tensori

Uno dei concetti alla base di queste librerie è quello di tensore⁹, una generalizzazione di tutte le strutture definite usualmente nell'algebra lineare a partire da un singolo spazio lineare. In estrema sintesi un tensore è una rappresentazione matematica di una entità fisica che può essere caratterizzata da una grandezza e direzioni multiple. TensorFlow implementa il tensore attraverso array dinamici multidimensionali.

Un oggetto `tf.Tensor` ha le seguenti proprietà:

- un tipo di dati (ad esempio `int32`, `float32`, `string`);
- un rank, ovvero il numero delle sue dimensioni
- una forma o shape, che identifica oltre al numero di dimensioni il numero di elementi per ciascuna dimensione.

Si definisce il rank o grado di un oggetto `tf.Tensor` come il numero delle sue dimensioni, riassumibili nella Tabella 7.

Tabella 7 Rank di un tensore in TensorFlow

Rank	Entita' matematica
0	Scalare (possiede solo la magnitudo)
1	Vettore (magnitudine e direzione)
2	Matrice (tabella di numeri)
3	3-tensor (cubo di numeri)
n	n-tensor (oggetto multidimensionale)

⁸ <https://www.tensorflow.org/>

⁹ <https://www.tensorflow.org/guide/tensors>

Si riporta di seguito un esempio di creazione di oggetto per ciascun rank.

```
#rank 0, una variabile di tipo intero a 16bit dal valore 451
ignition = tf.Variable(451, tf.int16)

#rank 1, un vettore contenente i numeri primi minori di 10
first_primes = tf.Variable([2, 3, 5, 7], tf.int32)

#rank 2, tabella contenente le prime potenze di 2 e 3
potenze_2_3 = tf.Variable([[2, 4, 8, 16], [3, 9, 27, 81]], tf.int32)

#la funzione rende il rank dell'oggetto passato in input, in questo caso 2
tf.rank(potenze_2_3)
```

La forma di un tensore può essere rappresentata in linguaggio Python con una lista di interi positivi. La Tabella 8 riporta alcuni esempi 0,1,2, n dimensioni.

Tabella 8 Forma e dimensione di un tensore in TensorFlow

Rank	Shape	Numero di dimensioni	Esempio
0	[]	0-D	Uno scalare.
1	[D0]	1-D	vettore di 5 elementi, con shape [5].
2	[D0, D1]	2-D	Matrice con forma [3, 4] ovvero 3 righe e 4 colonne.
n	[D0, D1, ... Dn-1]	n-D	Un tensore con forma [D0, D1, ... Dn-1].

A parità di numero di elementi presenti in un tensore può essere necessario modificarne la forma, ad esempio passando da forma [n,m] alla forma [n,m,1] quando si deve passare una matrice bidimensionale ad un modello che si aspetta ingresso oggetti a tre dimensioni. Per questa operazione si può utilizzare la funzione *tf.reshape*, come descritto nell'esempio seguente.

```
#genera un tensore di forma [3,4] inizializzato con tutti i valori a 1.
rank_due_tensor = tf.ones([3, 4])

# converto in un oggetto a una dimensione
rank_uno_tensor = tf.reshape(rank_due_tensor, [12])

# converto in un oggetto a tre dimensioni
```

```

rank_due_tensor = tf.reshape(rank_due_tensor, [3,4,1])
# la seguente riga restituisce errore perché varia il numero di elementi
rank_due_tensor = tf.reshape(rank_due_tensor, [3,4,2])
# la seguente riga restituisce errore perché la dimensione deve essere un
# numero intero positivo
rank_due_tensor = tf.reshape(rank_due_tensor, [3, 4, -1])

```

4.3.2. Lo struttura di un programma Tensorflow

Un programma TensorFlow si divide concettualmente in due parti, la prima dove si disegna un modello e con la definizione di array e operazioni, e la seconda dove vengono inizializzati i dati ed eseguite le operazioni. Nell'esempio riportato di seguito viene dapprima definito un modello come tipi di dato accettati e operazioni previste, e successivamente viene avviata la sua esecuzione con il comando `sess.run()`. La ragione di questa separazione è quella di poter definire dei modelli da poter eseguire più volte su dataset differenti.

Nell'esempio seguente vengono definiti tre tensori di rank 1 chiamati `x1`, `x2`, `x3`, che vengono definiti ma non valorizzati grazie all'uso della funzione `tf.placeholder`. Vengono quindi definite due operazioni di moltiplicazione ed assegnato il valore all'oggetto `result`. Quest'ultimo rappresenta il grafo computazionale del semplice modello definito.

Durante l'esecuzione del modello con il metodo `sess.run()` si provvedere al caricamento dei valori nei tensori `x1` e `x2`, attraverso il passaggio di due array NumPy.

```

# Import `tensorflow`
import tensorflow as tf
import numpy as np

# Definizione tensori con indicazione tipo e forma
x1 = tf.placeholder(tf.int64,(4))
x2 = tf.placeholder(tf.int64,(4))

# definizione operazioni
x3 = tf.multiply(x1, x2)
result = tf.multiply(x1, x3)

```

```

# esecuzione del modello `result`
with tf.Session() as sess:
    output = sess.run(result,feed_dict={x1:np.array([1,2,3,4]),x2:np.array([5,6,7,8])})
    print(output)

```

4.3.3. Tensorboard

All'interno della libreria Tensorflow è compresa e integrata anche una console di visualizzazione Web denominata Tensorboard, che permette di visualizzare i grafici di andamento delle metriche di esecuzione di un modello e di mostrare la sua struttura sottoforma di grafo.

```

logs_path = '/tmp/tensorflow_logs/example/'
# op to write logs to Tensorboard
summary_writer = tf.summary.FileWriter(logs_path, graph=tf.get_default_graph())

```

L'aspetto della console una volta configurato è simile a quello rappresentato nella Figura 37. Le linee arancione e viola mostrano l'andamento della metrica cross entropy rispettivamente durante la fase di training e di test. Portando il mouse sopra il grafico viene mostrato il riquadro nero in basso con il dettaglio analitico dei valori in quel punto.

L'asse orizzontale può essere selezionata in base alle seguenti opzioni:

- step: il numero progressivo di step, in questo caso ripetuto per ogni epoca;
- wall: il tempo di esecuzione relativo dall'avvio del programma, utile per vedere in linea l'andamento durante l'esecuzione di epoche successive;
- relative: il tempo di esecuzione relativo dall'ultimo data point, oppure dall'inizio dell'esecuzione come wall.

Il parametro Smoothing fornisce un'addolcimento della curva utile a capire la direzione presa dall'addestramento in presenza di sbalzi. Nella Figura 37 si può vedere in colore arancione chiarissimo la distribuzione reale dei valori di train e in arancione più acceso la sua versione smussata.

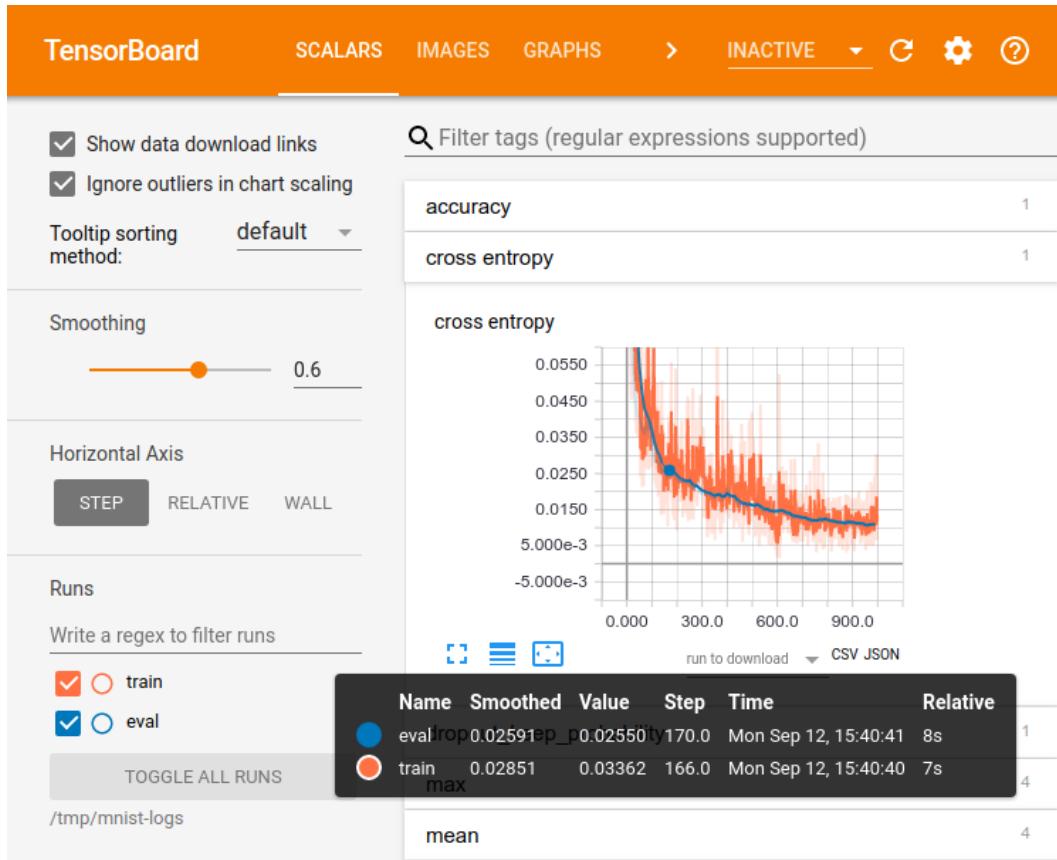


Figura 37 Il tool grafico Tensorboard

4.3.1. *Performance su CPU, GPU e TPU*

Uno dei punti di forza di questa libreria è il supporto nativo per le GPU Nvidia attraverso l'utilizzo dell'architettura CUDA (Compute Unified Device Architecture).

Secondi diversi benchmark pubblicati sul sito medium.com [11] [12] sperimentati la velocità nell'elaborazione dei modelli è anche di 10 volte superiore a quella dei più potenti CPU.

Si riportano nella Figura 38 e nella Figura 39 seguenti alcuni estratti a scopo esemplificativo.

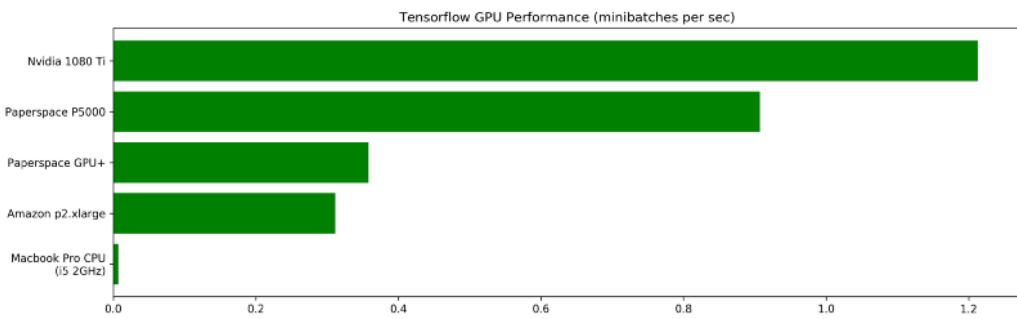


Figura 38 Confronto prestazioni della GPU Nvidia 1080 ti con altre soluzioni sul mercato

Device	Speed of training, examples/sec
2 x AMD Opteron 6168	440
i7-7500U	415
GeForce 940MX	1190
GeForce 1070	6500

Figura 39 Benchmark librerie TensorFlow versione CPU e GPU

La workstation su cui sono state eseguite le sperimentazioni ha le seguenti caratteristiche hardware:

- Scheda Madre. Asus Prime Z270-A
- Processore Intel i7-7700K
- GPU NVIDIA GeForce GTX 1080 ti
- 56GB Ram DDR4 2400Mhz
- 2 x 500GB SSD Crucial MX500

4.4. Keras

Keras è una libreria open source per reti neurali scritta in Python, e si caratterizza come un'astrazione di alto livello per la creazione di modelli che vengono poi tradotti ed eseguiti con diverse altre librerie tra cui Theano e TensorFlow. A partire dal 2017, Google ha deciso di includere Keras nella libreria principale di TensorFlow, in modo da rendere l'approccio al deep learning più intuitivo e ad alto livello.

I vantaggi dell'utilizzo di Keras sono i seguenti:

- Facilità di utilizzo: Keras ha interfacce semplici e ottimizzate per i più comuni casi d'uso;
- Modularità: I modelli Keras sono costituiti da moduli combinabili tra loro, con poche limitazioni;
- Estendibilità: È facile sviluppare componenti ad hoc per esprimere nuove idee di ricerca.

In Keras un modello viene definito da una serie di strati (Layer). Quando il modello è stato definito si utilizza il metodo `compile`, definendo gli obiettivi del training:

- *optimizer*: un oggetto che specifica la modalità di training
- *loss*: la funzione da minimizzare in fase di ottimizzazione
- *metrics*: il valore usato per monitorare il training.

Il modello quindi può utilizzato con i metodi *evaluate* (se passo anche il valore atteso) e *predict* (se non ho il valore atteso e lo voglio stimare).

```
import tensorflow as tf
from tensorflow import keras

model = keras.Sequential()
# Adds a densely-connected layer with 64 units to the model:
model.add(keras.layers.Dense(64, activation='relu'))
# Add another:
model.add(keras.layers.Dense(64, activation='relu'))
# Add a softmax layer with 10 output units:
model.add(keras.layers.Dense(10, activation='softmax'))
# Configure a model for categorical classification.
model.compile(optimizer=tf.train.RMSPropOptimizer(0.01),
              loss=keras.losses.categorical_crossentropy,
              metrics=[keras.metrics.categorical_accuracy])
model.evaluate(x, y, batch_size=32)
model.predict(x, batch_size=32)
```

4.4.1. Keras Layers

Vengono descritti in questa sezione alcuni dei principali layer di una rete neurale Keras.

La funzione Dense() implementa un livello fully connected, ovvero un'operazione lineare sul vettore di input.

Gli attributi di questa funzione sono i seguenti:

```
keras.layers.Dense(units, activation=None, use_bias=True,  
kernel_initializer='glorot_uniform', bias_initializer='zeros', kernel_regularizer=None,  
bias_regularizer=None, activity_regularizer=None, kernel_constraint=None,  
bias_constraint=None)
```

- **units**: numero intero positivo che indica le dimensioni del vettore di output;
- **activation**: Funziona di attivazione da utilizzare (ad esempio "linear": $a(x) = x$);
- **use_bias**: Booleano, se deve essere utilizzato o meno un vettore di bias (rumore);
- **kernel_initializer**: tipo di inizializzazione della matrice dei pesi;
- **bias_initializer**: tipo di inizializzazione per il vettore di bias;
- **kernel_regularizer**: Funzione di regolarizzazione applicata alla matrice dei pesi;
- **bias_regularizer**: Funzione di regolarizzazione applicata al vettore di bias; **activity_regularizer**: Funzione di regolarizzazione applicata all'output del layer;
- **kernel_constraint**: Funzione di vincolo applicata alla matrice dei pesi (es. non negatività);
- **bias_constraint**: Funzione di vincolo applicata al vettore di bias.

5. Modelli della PhysioNet / CinC Challenge2017

In questa sezione vengono riportati alcuni dei modelli proposti nell'ambito della challenge 2017 relativa alla classificazione di segnali ECG per la classificazione di casi di ritmo cardiaco normale e fibrillazione atriale.

5.1. Il modello di Xiong, Stiles, Zhao

La struttura e i risultati del modello prodotto dai ricercatori Xiong, Jichao e Zhao dell'università di Auckland (Nuova Zelanda) sono riassunti nella pubblicazione [13] "*Robust ECG Signal Classification for Detection of Atrial Fibrillation Using a Novel Neural Network*" consultabile gratuitamente dagli archivi della Cinc Conference.

Il modello proposto è una rete neurale convoluzionale (CNN) a 16 livelli migliorata con l'inserimento di alcune *skip connection* per collegare tra loro i livelli iniziali con quelli finali. Grazie a questo accorgimento sono state migliorate sia le capacità di apprendimento che il tempo di esecuzione necessario per l'addestramento del modello.

In ogni blocco vengono eseguite le stesse operazioni ai batch di dati alimentati con il meccanismo di feed:

- Batch-Normalization: effettua la normalizzazione dei valori sottraendo il valore medio di ogni batch e dividendo per la varianza, per garantire che tutti i valori siano scalati alla stessa magnitudine.
- ReLu Activation: funzione di attivazione rettificatore, definita come parte positiva dell'argomento (valore in input), definita come $f(x) = x^+ = \max(0, X)$.
- Dropout: una tecnica di regolarizzazione che ignora una parte casuale dei neuroni di una rete neurale per prevenire l'overfitting, ovvero l'eccessivo adattamento del modello ai dati ricevuti e quindi la sua scarsa capacità di generalizzare.

- 1D Convolution, l'applicazione di un filtro convoluzionale 15x1, che scorre il vettore del segnale per estrarre le caratteristiche della forma d'onda.
- 1D Average Pool: un layer per la riduzione della dimensione del batch che prende il valore medio di ogni due elementi consecutivi.
- 1D Max Pool: un layer per la riduzione della dimensione del batch che prende il valore massimo di ogni due elementi consecutivi.

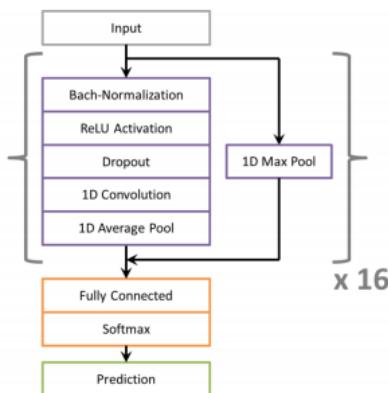


Figura 40 Immagine rappresentativa del modello Xiong per Cinc challenge 2017

Alla fine viene inserito un livello fully connected per trasformare l'output dei blocchi precedenti in un vettore 4x1 contenente le probabilità di classificazione di ciascuna delle quattro classi (N, A, O, ~). Questi valori sono normalizzati tra 0 e 1 con la funzione softmax.

I risultati di questo modello sono riepilogati nella Tabella 9 e nella Figura 41. Il valore di F-score complessivo è di 0.82, con un comportamento migliore nell'individuazione delle classi etichettate come Normale e Fibrillazione Atriale, e con performance lievemente inferiori nelle classi Other e Noisy. Queste ultime in particolare circa nel 35% dei casi sono state etichettate come N o AF.

Tabella 9 Risultati del modello Xiong, Stiles, Zhao

N	A	O	Complessivo
0.90	0.82	0.75	0.82

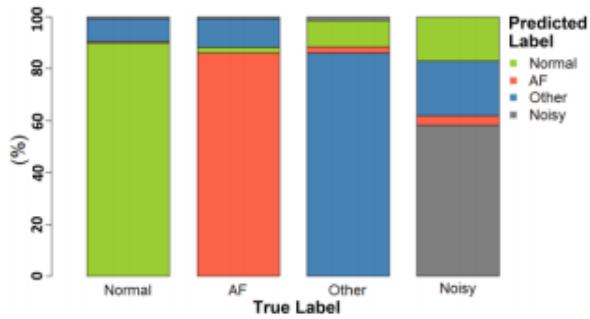


Figura 41 Risultati del modello Xiong, Stiles, Zhao

5.2. Il modello Andreotti (MATLAB Feature Based + Resnet)

Il lavoro [14] presentato da Fernando Andreotti e Oliver Carr, dell'Institute of Biomedical Engineering dell'University of Oxford, si presenta come un benchmark test comparativo tra un approccio tradizionale basato sull'analisi delle caratteristiche d'onda del segnale ECG ed un approccio Deep Learning, per il quale viene riproposto un modello di rete convoluzionale derivato dagli studi [15] di alcuni ricercatori dell'università di Stanford.

Come prima operazione i ricercatori si sono occupati di integrare il dataset di training con ulteriori 2000 record prelevati da diversi altri database PhysioNet (INCART-DB, LTAFDB, AFDB), chiamando il nuovo dataset AUG-DB. Questo per bilanciare meglio i casi, che erano molto sproporzionati verso la classe N.

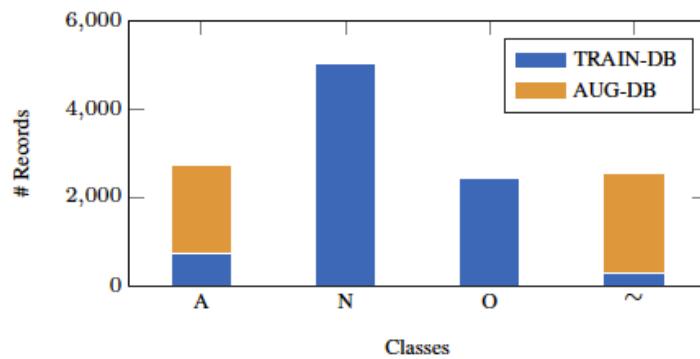


Figura 42 Distribuzione delle quattro classi del dataset della Cinc challenge 2017 prima e dopo l'integrazione

Tralasciamo la descrizione dettagliata dell'approccio feature based, in quanto non è oggetto di questa tesi e richiede competenze non banali dal punto di vista medico e l'utilizzo di strumenti di QRS detection, limitandoci a riportare come

questi approccio abbia portato alla scomposizione di ogni segnale in 169 features.

L'approccio deep learning è stato affrontato con un Residual Network (ResNet), una recente evoluzione delle reti neurali convoluzionali (CNN) che, grazie all'aggiunta di collegamenti tra i livelli più profondi e quelli più vicini all'output finale si avvicinano alle prestazioni delle Long Short Term Memory, della categoria delle RNN, pur richiedendo minori risorse in termini computazionali. Questa seconda soluzione è stata sviluppata in linguaggio Python e con le librerie Tensorflow e Keras.

L'architettura di questa rete, riportata graficamente nell'immagine seguente, è strutturata in 33 livelli convoluzionali seguiti un blocco fully connected e un softmax. Il blocco centrale è costituito da 15 blocchi residuali con 2 livelli convoluzionali ciascuno aventi un filtro di lunghezza 16. Ogni filtro convoluzionale è preceduto da un layer di batch normalization.

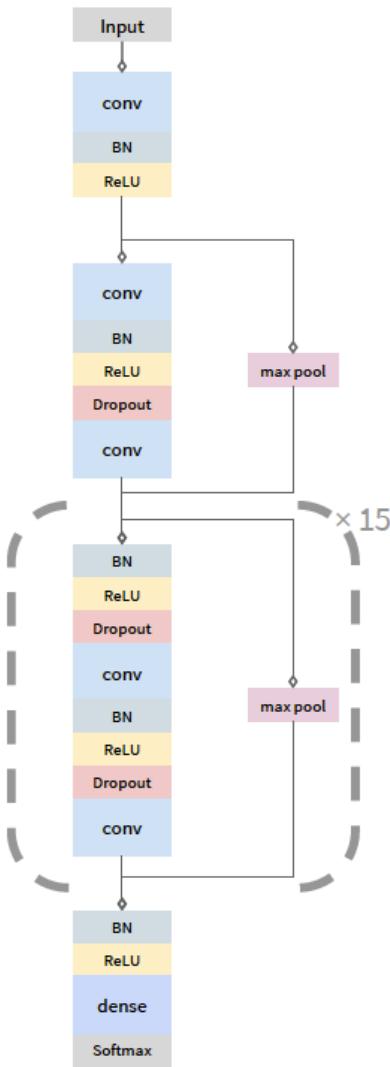


Figura 43 Archittettura della rete Resnet proposta da Rajpurkar-Hannun e utilizzata da Ferndando Andreotti per la Cinc challenge 2017

L'implementazione del modello è riportata nel blocco di codice riportato di seguito. La dimensione della finestra passata al modello è di 30 secondi per 300 Hz (frequenza di campionamento del segnale).

```
#####
## Model definition      ##
## ResNet based on Rajpurkar  ##
## Parameters
## FS = 300
## WINDOW_SIZE = 30*FS  # padding window for CNN
#####
```

```

def ResNet_model(WINDOW_SIZE):
    # Add CNN layers left branch (higher frequencies)
    # Parameters from paper
    INPUT_FEAT = 1
    OUTPUT_CLASS = 4  # output classes

    k = 1  # increment every 4th residual block
    p = True # pool toggle every other residual block (end with 2^8)
    convfilt = 64
    convstr = 1
    kszie = 16
    poolsize = 2
    poolstr = 2
    drop = 0.5

    # Modelling with Functional API
    #input1 = Input(shape=(None,1), name='input')
    input1 = Input(shape=(WINDOW_SIZE,INPUT_FEAT), name='input')

    ## First convolutional block (conv,BN, relu)
    x = Conv1D(filters=convfilt,
               kernel_size=kszie,
               padding='same',
               strides=convstr,
               kernel_initializer='he_normal')(input1)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)

    ## Second convolutional block (conv, BN, relu, dropout, conv) with residual net
    # Left branch (convolutions)
    x1 = Conv1D(filters=convfilt,
                kernel_size=kszie,
                padding='same',
                strides=convstr,
                kernel_initializer='he_normal')(x)
    x1 = BatchNormalization()(x1)
    x1 = Activation('relu')(x1)
    x1 = Dropout(drop)(x1)
    x1 = Conv1D(filters=convfilt,

```

```

        kernel_size=ksize,
        padding='same',
        strides=convstr,
        kernel_initializer='he_normal')(x1)
x1 = MaxPooling1D(pool_size=poolsize,
                  strides=poolstr)(x1)
# Right branch, shortcut branch pooling
x2 = MaxPooling1D(pool_size=poolsize,
                  strides=poolstr)(x)
# Merge both branches
x = keras.layers.add([x1, x2])
del x1,x2

## Main loop
p = not p
for l in range(15):

    if (l%4 == 0) and (l>0): # increment k on every fourth residual block
        k += 1
        # increase depth by 1x1 Convolution case dimension shall change
        xshort = Conv1D(filters=convfilt*k,kernel_size=1)(x)
    else:
        xshort = x
    # Left branch (convolutions)
    # notice the ordering of the operations has changed
    x1 = BatchNormalization()(x)
    x1 = Activation('relu')(x1)
    x1 = Dropout(drop)(x1)
    x1 = Conv1D(filters=convfilt*k,
                kernel_size=ksize,
                padding='same',
                strides=convstr,
                kernel_initializer='he_normal')(x1)
    x1 = BatchNormalization()(x1)
    x1 = Activation('relu')(x1)
    x1 = Dropout(drop)(x1)
    x1 = Conv1D(filters=convfilt*k,
                kernel_size=ksize,
                padding='same',

```

```

        strides=convstr,
        kernel_initializer='he_normal')(x1)

    if p:
        x1 = MaxPooling1D(pool_size=poolsize,strides=poolstr)(x1)

    # Right branch: shortcut connection
    if p:
        x2 = MaxPooling1D(pool_size=poolsize,strides=poolstr)(xshort)
    else:
        x2 = xshort # pool or identity
    # Merging branches
    x = keras.layers.add([x1, x2])
    # change parameters
    p = not p # toggle pooling

# Final bit
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = Flatten()(x)
#x = Dense(1000)(x)
#x = Dense(1000)(x)
out = Dense(OUTPUT_CLASS, activation='softmax')(x)
model = Model(inputs=input1, outputs=out)
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
#model.summary()
#sequential_model_to_ascii_printout(model)
plot_model(model, to_file='model.png')
return model

```

La tecnica utilizzata per l'addestramento di questo modello è stata la seguente:

- trasformare il dataset unendo prima tutti i record in un unico file matlab, e poi caricandolo su una matrice di segmenti di 30 secondi.

- suddividere cinque volte il dataset di training in 4/5 di dati da passare con metodo fit per l'addestramento e 1/5 di dati da passare per la validazione, con scelta dei campioni casuali.
- generare per ognuna di queste 5 cross-validation un file di pesi diverso.
- scegliere la configurazione di pesi più performante

Il risultato migliore ottenuto dalla soluzione con ResNet lavorando sul solo dataset ufficiale è quello di una F1 - measure uguale a 0.79, pari al valore massimo ottenuto con l'approccio feature-based. Tuttavia la soluzione ResNet addestrata con il database integrato AUG-DB supera questo valore e ottiene un punteggio 0.83, lo stesso risultato dei primi 5 gruppi classificati alla Cinc Challenge 2017.

Tabella 10 Riepilogo risultati dei modelli di F. Andreotti per la Cinc Challenge 2017

Description	Validation on training (%)						Results for TEST-DB (%)			
	Set	N	A	O	~	Total	N	A	O	Total
Feature-based approach (no segmentation)	TRAIN-DB	90.6	72.1	74.8	50.5	72.0	89	80	69	79
Feature-based approach (10 s segments)	TRAIN-DB	89.9	76.8	73.7	66.0	76.6	90	77	68	78
ResNet (original 34 L, 30s SEG)	AUG-DB	90.2	65.7	69.8	64.0	72.4	n.a.	n.a.	n.a.	n.a.
ResNet (16 CF, 60s SEG)	TRAIN-DB	82.6	46.6	60.0	60.2	62.4	92	70	75	79
ResNet (16 CF, 60s SEG)	AUG-DB	88.5	67.7	66.6	65.6	72.1	93	78	78	83
ResNet (16 CF, 34 L, 30s SEG, no noise)	AUG-DB	89.6	65.3	69.1	0.0	56.0	71	41	43	52

5.3. Modello Bidirectional LSTM - esempio MATLAB

Il modello descritto in questa sezione è stato riportato [16] all'interno degli esempi di MATLAB versione R2018a nella categoria Deep Learning, Signal Processing Toolbox. A differenza dei modelli precedenti, è stato scelto di ignorare le classi Other Rhythm (O) e Noisy Recording(~), per concentrarsi solo sulle classi Normal (N) e Afib (A). Il modello di questo esempio è basato su bidirectional LSTM, un tipo di rete neurale ricorrente (RNN) la cui architettura è descritta nelle sezioni precedenti.

La possibilità di testare questo modello e calcolarne la sua F1-measure è stata offerta da una licenza trial di 30 giorni.

5.3.1. Il modulo di caricamento ReadPhysionetData

Il modulo di caricamento effettua il download direttamente dal sito web PhysioNet, provvede a eliminare i record relativi alle classi O e ~ e a caricare i dati in unico file MATLAB in due array Signals e Labels.

```
% This script parses the data from the PhysioNet 2017 Challenge and saves
% the data into PhysionetData.mat for quick and easy future use.

% Copyright 2017 The MathWorks, Inc.

% Download and unzip the data, training2017.zip, from the PhysioNet website
unzip('https://physionet.org/challenge/2017/training2017.zip')

% Navigate to the directory
cd training2017

% File with filenames and labels
ref = 'REFERENCE.csv';

% Create a table that contains the filenames and corresponding label data
tbl = readtable(ref,'ReadVariableNames',false);
tbl.Properties.VariableNames = {'Filename','Label'};

% Delete 'Other Rhythm' and 'Noisy Recording' signals
toDelete = strcmp(tbl.Label,'O') | strcmp(tbl.Label,'~');
tbl(toDelete,:) = [];

% Load each file in the table and store the corresponding signal data
H = height(tbl);
for ii = 1:H
    fileData = load([tbl.Filename{ii},'.mat']);
    tbl.Signal{ii} = fileData.val;
end

% Leave the training2017 directory
cd ..

% Format the data properly for LSTM training
% Signals: Cell array of predictors
% Labels: Categorical array of responses
Signals = tbl.Signal;
Labels = categorical(tbl.Label);
% Save the variables to a MAT-file
save PhysionetData.mat Signals Labels
```

5.3.2. Preparazione dataset in formato vettoriale

Il dataset univoco letto dal file PhysionetData.mat sono caratterizzati da una lunghezza variabile, anche se la loro rappresentazione su istogramma, riportata nella Figura 44, mostra come la maggior parte di essi abbiano una lunghezza di 9000 campioni (30 secondi campionati a 300 Hz).

```
L = cellfun(@length,Signals);
h = histogram(L);
xticks(0:3000:18000);
xticklabels(0:3000:18000);
title('Signal Lengths')
xlabel('Length')
ylabel('Count')
```

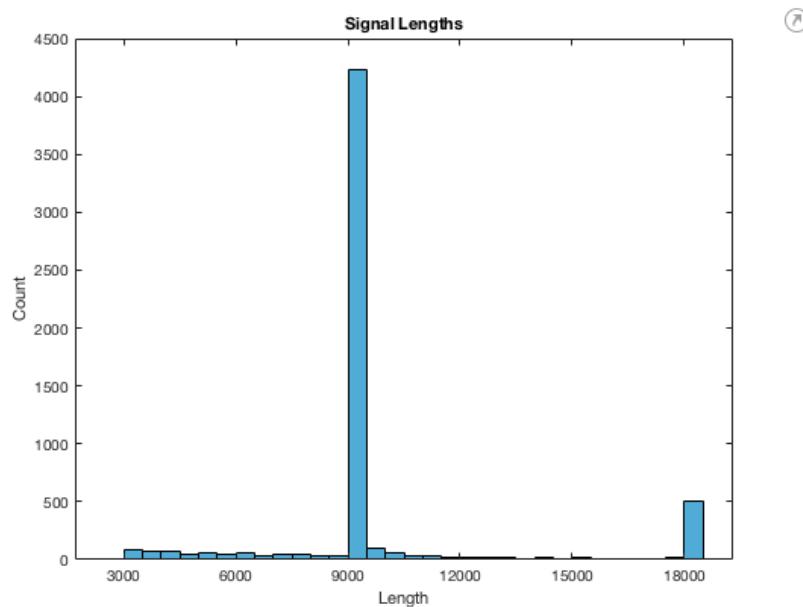


Figura 44 Distribuzione dei record per durata temporale

Per ottenere campioni tutti uguali viene quindi scelto di troncare tutti i campioni di lunghezza superiore, e scartare quelli di lunghezza inferiore. Questo lavoro viene svolto dalla funzione *segmentSignals*.

```
[Signals,Labels] = segmentSignals(Signals,Labels);
```

Si riporta di seguito la sua implementazione nel modulo *segmentSignal.m*

```
function [signalsOut, labelsOut] = segmentSignals(signalsIn,labelsIn)
%SEGMENTSIGNS makes all signals in the input array 9000 samples long
% Copyright 2017 The MathWorks, Inc.

targetLength = 9000;
signalsOut = {};
labelsOut = {};
for idx = 1:numel(signalsIn)
    x = signalsIn{idx};
    y = labelsIn(idx);
    % Ensure column vector
    x = x(:);
    % Compute the number of targetLength-sample chunks in the signal
    numSigs = floor(length(x)/targetLength);
    if numSigs == 0
        continue;
    end
    % Truncate to a multiple of targetLength
    x = x(1:numSigs*targetLength);
    % Create a matrix with as many columns as targetLength signals
    M = reshape(x,targetLength,numSigs);
    % Repeat the label numSigs times
    y = repmat(y,[numSigs,1]);
    % Vertically concatenate into cell arrays
    signalsOut = [signalsOut; mat2cell(M.',ones(numSigs,1))]; %#ok<AGROW>
    labelsOut = [labelsOut; cellstr(y)]; %#ok<AGROW>
end
labelsOut = categorical(labelsOut);
end
```

A questo viene confrontato lo sbilanciamento del dataset nella attribuzione dei record alle due classi A e N, individuando un rapporto 1:7

```
summary(Labels)
A 718
N 4937
```

Poiché i record di classe N sono pari all'87,3%, sarebbe troppo facile per il modello raggiungere prestazioni elevate predicendo sempre la classe Normal. Per evitare questo problema i record di classe Afib sono duplicati in modo da pareggiare il numero dei record di classe N. Questa operazione viene definita *oversampling*.

Vengono divisi i dataset secondo le due classi, e poi suddivise ulteriormente in record da usare per il train e record da usare per il test del modello.

```
afibX = Signals(Labels=='A');  
afibY = Labels(Labels=='A');  
normalX = Signals(Labels=='N');  
normalY = Labels(Labels=='N');  
[trainIndA,~,testIndA] = dividerand(718,0.9,0.0,0.1);  
[trainIndN,~,testIndN] = dividerand(4937,0.9,0.0,0.1);  
  
XTrainA = afibX(trainIndA);  
YTrainA = afibY(trainIndA);  
  
XTrainN = normalX(trainIndN);  
YTrainN = normalY(trainIndN);  
  
XTestA = afibX(testIndA);  
YTestA = afibY(testIndA);  
  
XTestN = normalX(testIndN);  
YTestN = normalY(testIndN);
```

A questo punto il dataset di training si compone di 639 record AFib e 4441 Normal, mentre il dataset di test si compone di 71 record AFib e 494 Normal. Per pareggiare la numerosità di dei casi AFib con quelli Normal viene usata la funzione repmat e vengono rimossi alcuni record di classe N.

```
XTrain = [repmat(XTrainA(1:634),7,1); XTrainN(1:4438)];  
YTrain = [repmat(YTrainA(1:634),7,1); YTrainN(1:4438)];
```

```

XTest = [repmat(XTestA(1:70),7,1); XTestN(1:490)];
YTest = [repmat(YTestA(1:70),7,1); YTestN(1:490)];
summary(YTrain)
A 4438
N 4438
summary(YTest)
A 490
N 490

```

5.3.3. Esecuzione modello LSTM senza estrazione feature

Viene quindi eseguita una prima versione del modello BiLSTM, con i layer e le opzioni riportati di seguito. In particolare si osservi come venga scelto l'ottimizzatore ADAM, che funziona meglio di SGD nelle RNN.

```

layers = [ ...
    sequenceInputLayer(1)           " Sequence input with 1 dimensions
    bilstmLayer(100,'OutputMode','last')      "BiLSTM with 100 hidden units
    fullyConnectedLayer(2)          "2 fully connected layer
    softmaxLayer                  "softmax activation
    classificationLayer          " Classification Output crossentropyex
]
options = trainingOptions('adam', ...
    'MaxEpochs',10, ...
    'MiniBatchSize', 150, ...
    'InitialLearnRate', 0.01, ...
    'SequenceLength', 1000, ...
    'GradientThreshold', 1, ...
    'plots','training-progress', ...
    'Verbose',false);

```

L'addestramento del modello viene avviato con il seguente comando:

```
net = trainNetwork(XTrain,YTrain,layers,options);
```

I risultati di accuratezza mostrano una difficoltà dell'apprendimento, con l'accuratezza che oscilla intorno al 56% ed un loss intorno al valore 0.7. Come

si può osservare graficamente nella Figura 45, sia la linea azzurra dell'accuratezza che la linea arancione del loss sono caratterizzati da un comportamento altalenante ma senza una convergenza verso valori ottimali.

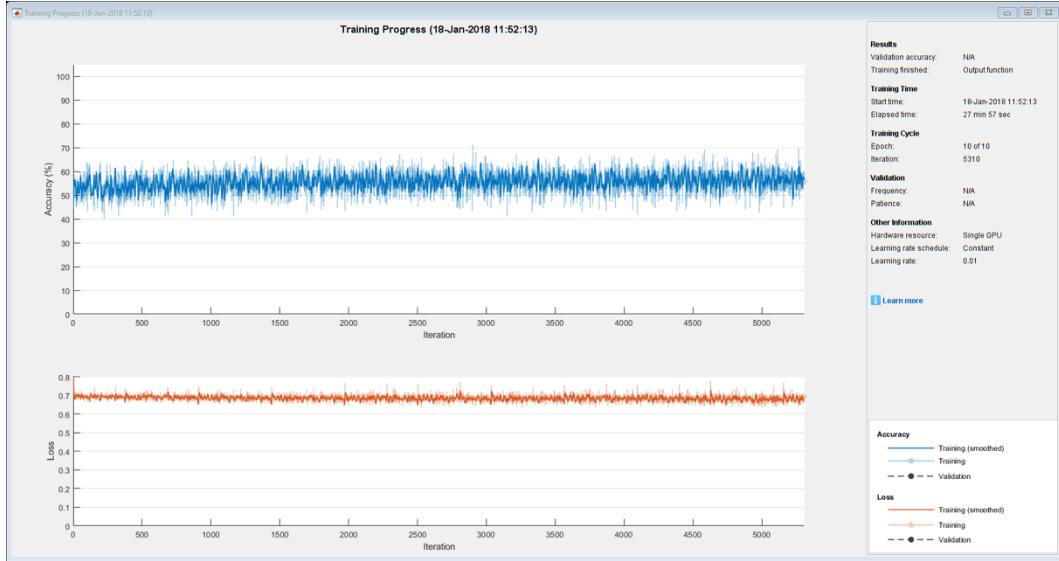


Figura 45 Grafico dell'addestramento del modello LSTM MATLAB senza feature extraction

In particolare come riportato nella matrice di confusione nella Figura 46 solo il 31% dei record di classe N del dataset di training viene correttamente classificato. Questo valore si riduce a poco meno del 29% nella matrice di confusione del dataset di test.

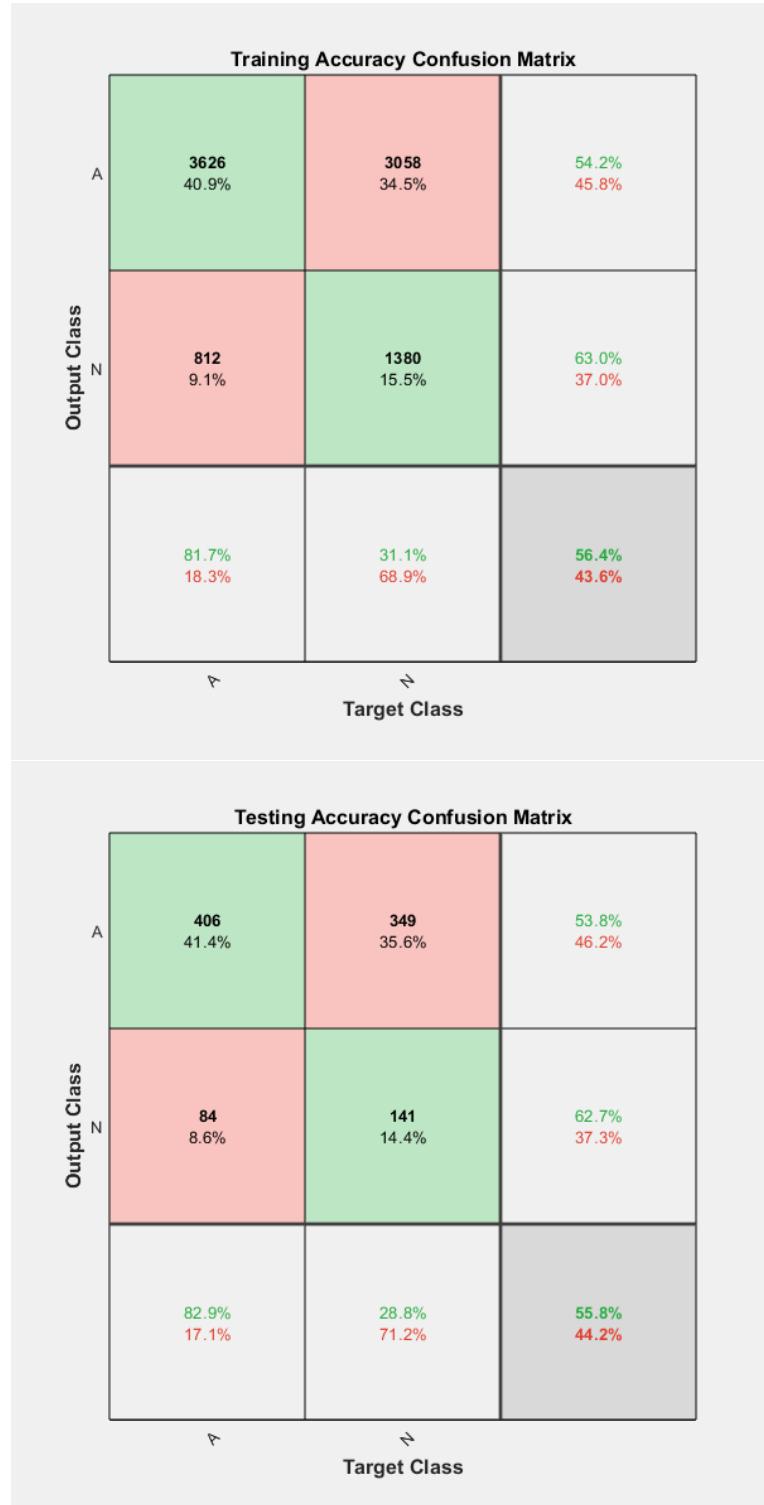


Figura 46 Matrici di confusione del modello LSTM MATLAB senza feature extraction

5.3.4. Estrazione feature instantaneous frequency e power spectral entropy

Per cercare di migliorare le prestazioni si è cercato di individuare alcune feature che fossero più caratterizzanti del segnale. Alcune ricerche pregresse [17] nel campo dell'analisi dei segnali musicali hanno evidenziato la possibilità di leggere la variabilità della frequenza del tempo con i così definiti Time-frequency (TF) moment [18].

L'applicazione di queste funzioni permette di cogliere meglio il fatto che i segnali di classe A N, seppure simili, hanno un comportamento diverso nel tempo: difatti come si può osservare nella Figura 47, il ritmo normale ha una periodicità abbastanza regolare mentre la fibrillazione atriale si caratterizza con una distanza più variabile tra i picchi.

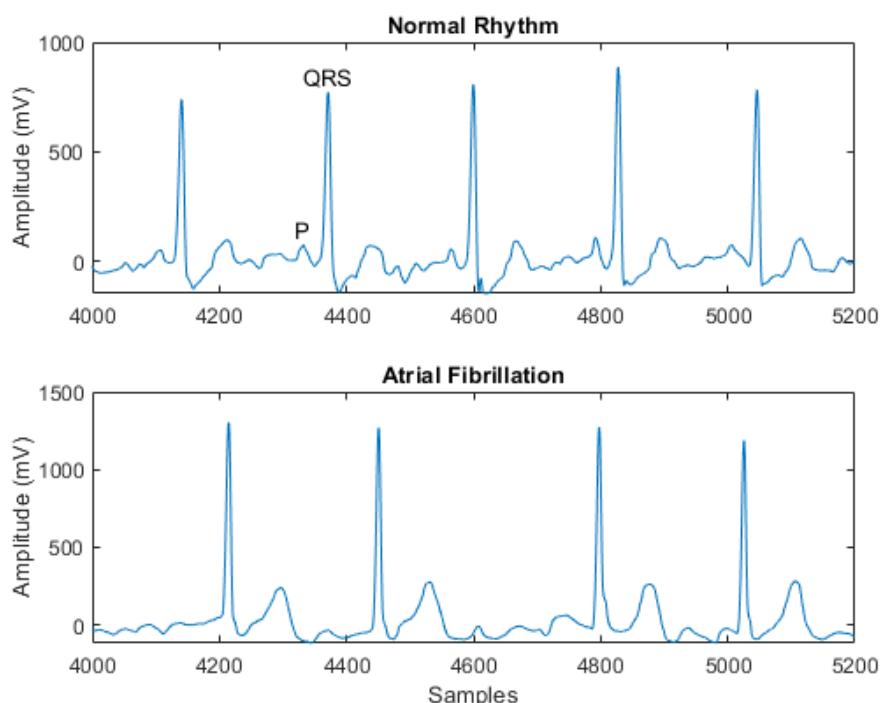


Figura 47 Rappresentazione grafica di due segnali di classe Norma e AFib

In particolare vengono utilizzati i due momenti:

- Instantaneous frequency (funzione `instfreq` [19])
- Power Spectral Entropy (funzione `pentropy` [20])

Le rispettive funzioni sono contenute all'interno del componente aggiuntivo di MATLAB denominato *Signal Processing Toolbox*.

La frequenza istantanea è una funzione tempo dipendente che misura la variabilità della frequenza nel tempo. Si riporta nella Figura 48 un esempio del risultato della applicazione di questa funzione su un record.

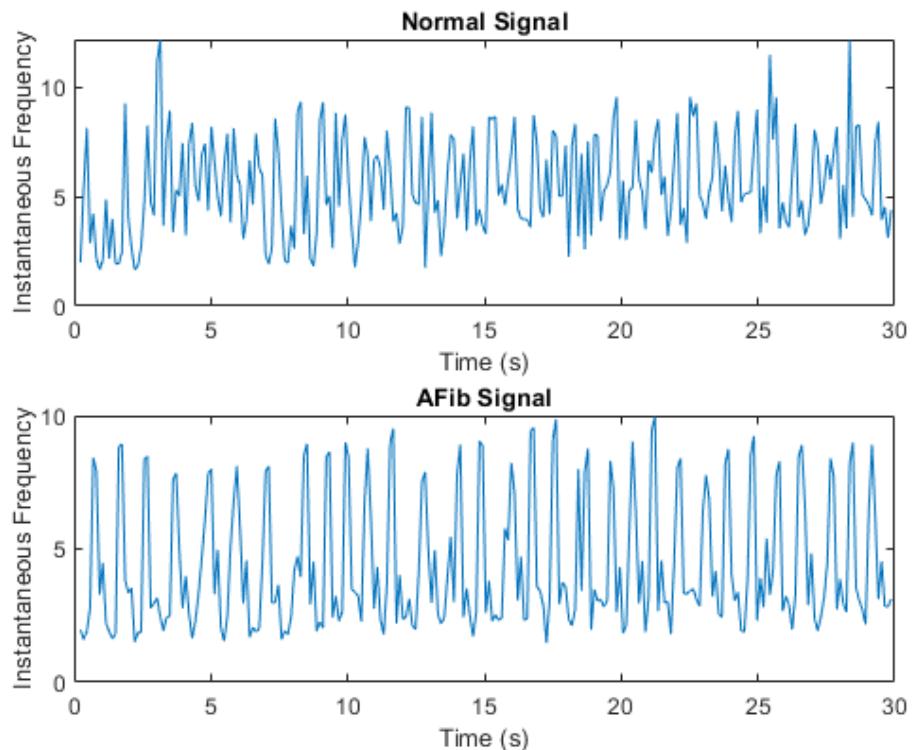


Figura 48 Instantaneous Frequency di un record di esempio (classe N e A)

L'entropia spettrale misura quanto spigoloso è lo spettro di un segnale. Si riporta nella Figura 49 un esempio del risultato della applicazione di questa funzione su un record.

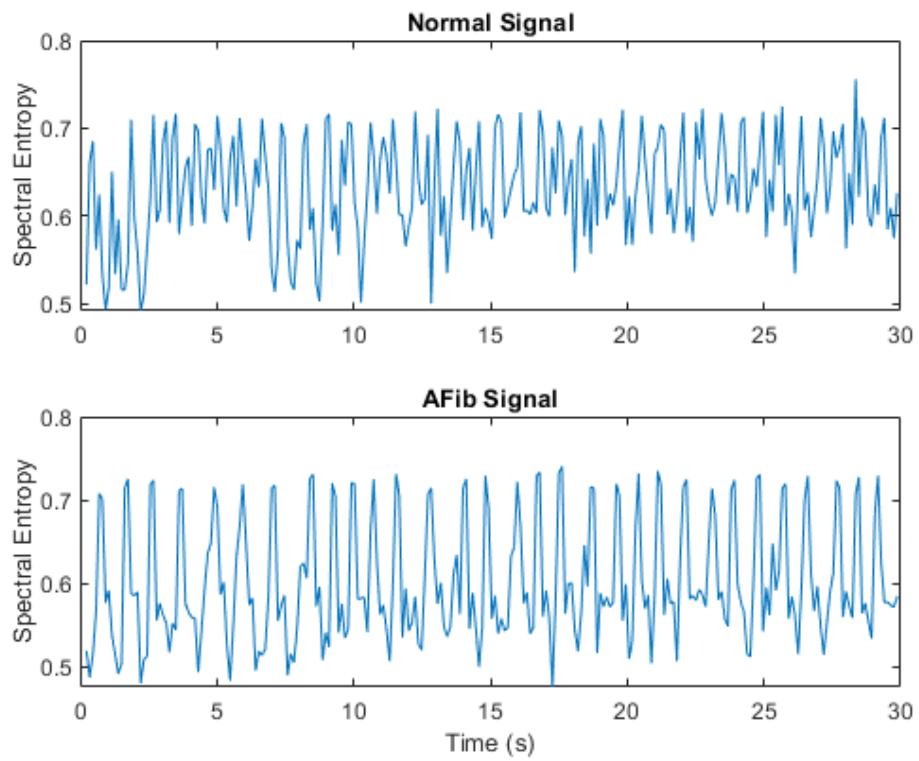


Figura 49 Spectral Entropy di un record di esempio (classe N e A)

L'applicazione di queste due funzioni genera un segnale a due canali. Viene utilizzata la funzione `cellfun` per l'applicazione della funzione ad ogni record.

```
instfreqTrain = cellfun(@(x)instfreq(x,fs)',XTrain,'UniformOutput',false);
instfreqTest = cellfun(@(x)instfreq(x,fs)',XTest,'UniformOutput',false);
pentropyTrain = cellfun(@(x)pentropy(x,fs)',XTrain,'UniformOutput',false);
pentropyTest = cellfun(@(x)pentropy(x,fs)',XTest,'UniformOutput',false);
XTrain2 = cellfun(@(x,y)[x;y],instfreqTrain,pentropyTrain,'UniformOutput',false);
XTest2 = cellfun(@(x,y)[x;y],instfreqTest,pentropyTest,'UniformOutput',false);
```

Ogni elemento del dataset di training passa da avere una dimensione di 1x9000 ad una dimensione di 2x255. La riduzione è motivata dal fatto che entrambe le funzioni sopra descritte campionano il segnale in 255 intervalli temporali, restituendo il valore centrale.

```
XTrain2(1:5)
ans = 5×1 cell array
{2×255 double}
{2×255 double}
```

```
{2×255 double}  
{2×255 double}  
{2×255 double}
```

A questo punto come in tutti gli scenari di rete neurale con canali di input multipli, si deve procedere alla standardizzazione dei valori per evitare che il canale con la magnitudo più grande degli altri copra gli altri.

Difatti come si può osservare di seguito la media della frequenza istantanea è molto maggiore di quella dell'entropia spettrale.

```
mean(instFreqN)  
ans = 5.5615  
mean(pentropyN)  
ans = 0.6326
```

La standardizzazione avviene sottraendo da ciascun elemento la media e poi dividendo per la deviazione standard (radice quadrata della varianza).

```
XV = [XTrain2{:}];  
mu = mean(XV,2);  
sg = std(XV,[],2);  
  
XTrainSD = XTrain2;  
XTrainSD = cellfun(@(x)(x-mu)./sg,XTrainSD,'UniformOutput',false);  
  
XTestSD = XTest2;  
XTestSD = cellfun(@(x)(x-mu)./sg,XTestSD,'UniformOutput',false);
```

Con l'applicazione della standardizzazione i due segnali assumono media pari a circa -0,32 e -0,24.

I due segnali ricavati sono stati salvati su file MATLAB insieme ai tracciati di target Ytrain e Ytest per poter essere utilizzati nel modello Keras descritto nella sezione di livello superiore successiva.

```
save PhysionetDataNew4.mat XTrainSD YTrainStr XTestSD YTestStr
```

5.3.5. Esecuzione modello LSTM con estrazione feature

Per poter essere eseguire il modello LSTM è necessario eseguire una modifica al livello di input indicando la dimensione di due.

```
layers = [ ...
    sequenceInputLayer(2)
    biLSTMLayer(100,'OutputMode','last')
    fullyConnectedLayer(2)
    softmaxLayer
    classificationLayer
]
```

L'addestramento della rete neurale avviene con il comando seguente.

```
net2 = trainNetwork(XTrainSD,YTrain,layers,options);
```

Il risultato di questo addestramento è notevolmente più positivo, con il valore dell'accuratezza che cresce nel tempo fino a superare il 90%. Il valore del loss si avvicina allo zero. Inoltre grazie alla riduzione del numero di campioni è stato possibile rieseguire questo modello nel mio portatile in 152 minuti. Osservando la Figura 50 si può avere subito un'idea come l'accuratezza, rappresentata dalla linea azzurra, tenda verso il 100%, e la loss function tenda verso lo zero. Per una maggiore comprensione si può confrontare la figura seguente con la Figura 45.

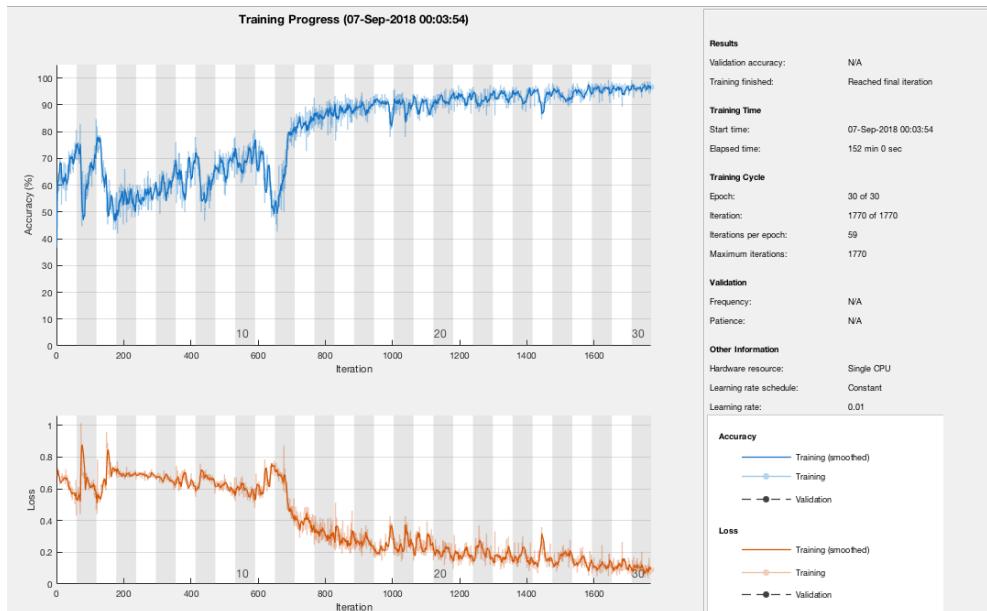


Figura 50 Grafico dell'addestramento del modello LSTM MATLAB con feature extraction

Le matrici di accuratezza riportate in Figura 51 mostrano un'accuratezza complessiva pari a circa il 97% in fase di training e il 93% in fase di test.

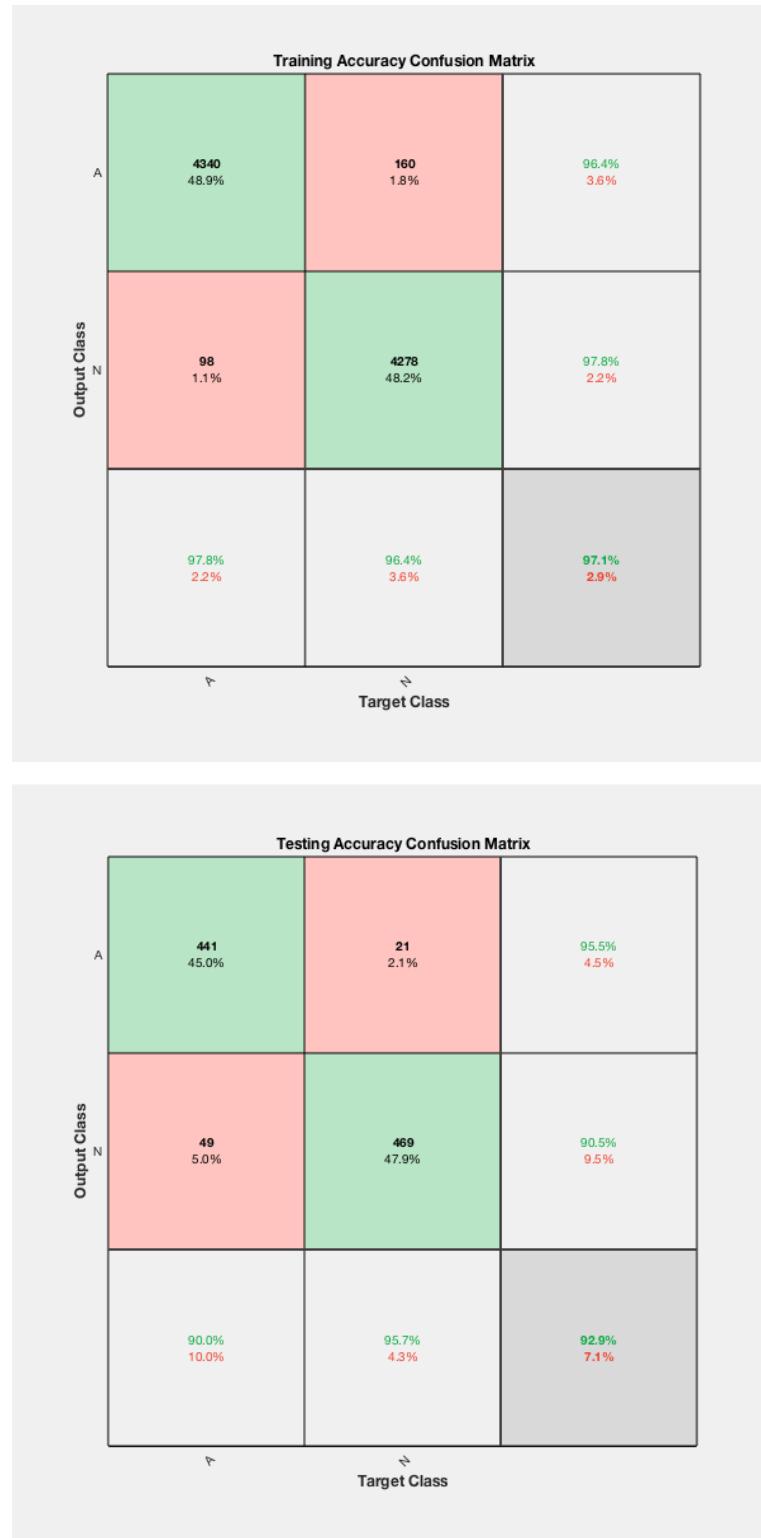


Figura 51 Matrici di confusione del modello LSTM MATLAB con feature extraction

Per concludere l'esame di questo modello si è provveduto a implementare la funzione di attribuzione del punteggio della challenge 2017, ovvero la F1-measure, che non era stata calcolata dagli autori dell'esempio.

Il risultato mostra un valore pari a circa 0.93, ma è doveroso ricordare come questo modello ha provveduto all'eliminazione delle classi Other e Noisy che erano quelle in cui anche i modelli vincitori della challenge ottenevano le performance peggiori.

```
Cm = confusionmat(YTest,testPred2')
F1=zeros([2 1],'double')
F1(1,1)= 2*Cm(1,1)/((Cm(1,1)+Cm(1,2))+(Cm(1,1)+Cm(2,1)))
F1(2,1)= 2*Cm(2,2)/((Cm(2,1)+Cm(2,2))+(Cm(1,2)+Cm(2,2)))
F1mean=mean(F1)

F1mean=0.9285
```

5.4. Modello Bidirectional LSTM - Keras

Sulla base dell'esempio precedente si è sperimentata l'implementazione della soluzione descritta in linguaggio Python con le librerie Keras e Tensorflow. Alcune funzioni non fondamentali del modello, come il disegno della matrice di confusione e l'acquisizione del dataset da file MATLAB .mat, sono state derivate dal progetto di Fernando Andreotti¹⁰ per la challenge 2017

5.4.1. *funzione di acquisizione dataset*

La funzione si occupa di leggere il dataset di training e di test secondo quanto esportato dal precedente modello MATLAB.

```
#####
## Function to load data ##
#####

def loaddata():
    file_name='PhysionetDataNew4.mat'
    matfile = scipy.io.loadmat(file_name)
    x_train_sd = matfile['XTrainSD']
    y_train_str = matfile['YTrainStr']
```

¹⁰ Una copia dei sorgenti è disponibile in <https://github.com/fernandoandreotti/cinc-challenge2017>

```

x_test_sd = matfile['XTestSD']
y_test_str = matfile['YTestStr']
#trasformo il tracciato x_train in 3d per lstm
X_train = np.zeros((len(x_train_sd), 255, 2))
i = 0
for item in x_train_sd:
    X_train[i] = np.transpose(item[0])
    i += 1

# trasformo il tracciato x_test in 3d per lstm
X_test = np.zeros((len(x_test_sd), 255, 2))
i = 0
for item in x_test_sd:
    X_test[i] = np.transpose(item[0])
    i += 1

# trasformo il tracciato delle classificazioni in matrice
classes = ['A', 'N']
y_train_classes = np.zeros((len(y_train_str), len(classes)))
i=0;
for item in y_train_str:
    if(item=='A'):
        y_train_classes[i][0]=1
    if (item == 'N'):
        y_train_classes[i][1]=1
    i+=1

# trasformo il tracciato delle classificazioni in matrice
y_test_classes = np.zeros((len(y_test_str), len(classes)))
i = 0;
for item in y_test_str:
    if (item == 'A'):
        y_test_classes[i][0] = 1
    if (item == 'N'):
        y_test_classes[i][1] = 1
    i += 1

return (X_train, y_train_classes, X_test, y_test_classes)

```

5.4.2. definizione del modello

La definizione del modello è analoga a quella del modello MATLAB. Il numero di livelli nascosti della LSTM è incrementato da 100 a 256, e per la limitazione del overfitting viene inserito un dropout di 0.2.

A seguito della bidirectional LSTM layer vengono aggiunti due livelli Dense e una funzione di attivazione softmax. La funzione di calcolo di costo è categorical_crossentropy e l'optimizer scelto è ADAM con learning rate fisso al valore 0.01.

```
#####
## Model definition
## Bidirectional LSTM
## LAYERS:
## 1 " Sequence Input      Sequence input with 2 dimensions
## 2 " BiLSTM              BiLSTM with 100 hidden units
## 3 " Fully Connected    2 fully connected layer
## 4 " Softmax             softmax
## 5 " Classification Output crossentropyex
#####

def LSTM_model(WINDOW_SIZE):
    INPUT_FEAT = 2    # input features: instantaneous frequency, spectral entropy
    OUTPUT_CLASS = 2  # output classes (Normal and AFib classes)

    model = Sequential()
    model.add(Bidirectional(LSTM(256, dropout=0.2,
                                recurrent_dropout=0.2, return_sequences=False), input_shape=(None, INPUT_FEAT),
                            merge_mode='mul'))
    model.add(Dense(OUTPUT_CLASS))
    model.add(Dense(OUTPUT_CLASS, activation='softmax'))
    model.compile(optimizer=keras.optimizers.Adam(lr=0.01),
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
    plot_model(model, to_file='model.png')
    return model
```

5.4.3. Esecuzione del modello

L'esecuzione del modello avviene attraverso il codice riportato di seguito, in cui viene invocata la funzione *loaddata()* per recuperare i dataset e poi passarli alla funzione *model_eval()*. Dopo l'esecuzione del modello vengono recuperate le predizioni dal file *xval_result.mat* ed eseguito il calcolo della F1-measure.

```
# Loading data
(X_train,y_train,X_test, y_test) = loaddata()
# Training model
model = model_eval(X_train,y_train,X_test, y_test)
# Outputting results of cross validation
matfile = scipy.io.loadmat('xval_results.mat')
cv = matfile['cvconfusion']
F1mean = np.zeros(cv.shape[2])
for j in range(cv.shape[2]):
    classes = ['A', 'N']
    F1 = np.zeros((2,1))
    for i in range(2):
        F1[i]=2*cv[i,i,j]/(np.sum(cv[i,:,j])+np.sum(cv[:,i,j]))
        print("F1 measure for {} rhythm: {:.1f}".format(classes[i],F1[i,0]))
    F1mean[j] = np.mean(F1)
    print("mean F1 measure for: {:.1f}".format(F1mean[j]))
print("Overall F1 : {:.1f}".format(np.mean(F1mean)))
# Plotting confusion matrix
cvsum = np.sum(cv, axis=2)
for i in range(2):
    F1[i]=2*cvsum[i,i]/(np.sum(cvsum[i,:])+np.sum(cvsum[:,i]))
    print("F1 measure for {} rhythm: {:.1f}".format(classes[i],F1[i,0]))
F1mean = np.mean(F1)
```

La funzione *model_eval()* si occupa di istanziare il modello e avviare l'addestramento con il metodo *fit*. I dati relativi alla matrice di confusione vengono salvati nel file *xval_result.mat* per il calcolo finale della F1-measure media.

```
def model_eval(Xtrain,ytrain,Xtest, ytest):
    batch =250
    epochs = 50
```

```

rep = 100      # procedure can be repeated multiple times
classes = ['A', 'N']
Nclass = len(classes)
cvconfusion = np.zeros((Nclass,Nclass,rep))
cvscores = []
counter = 0
# repetitions loop
for r in range(rep):
    print("Rep %d"%(r+1))
    # Callbacks definition
    callbacks = [
        # Early stopping definition
        EarlyStopping(monitor='val_loss', patience=8, verbose=1),
        # Saving best model
        ModelCheckpoint('weights-best_r{}.hdf5'.format(r), monitor='val_loss',
        save_best_only=True, verbose=1),
    ]
    # Load model
    model = LSTM_model()
    # Train model
    model.fit(Xtrain, ytrain,
               validation_data=(Xtest, ytest),
               epochs=epochs, batch_size=batch, callbacks=callbacks)
    # Evaluate best trained model
    model.load_weights('weights-best_r{}.hdf5'.format(r))
    ypred = model.predict(Xtest)
    ypred = np.argmax(ypred, axis=1)
    ytrue = np.argmax(ytest, axis=1)
    cvconfusion[:, :, counter] = confusion_matrix(ytrue, ypred)
    F1 = np.zeros((2,1))
    for i in range(2):

        F1[i]=2*cvconfusion[i,i,counter]/(np.sum(cvconfusion[:, :, counter])+np.sum(cvconfusion[:, i, counter]))
        print("F1 measure for {} rhythm: {:.1.4f}".format(classes[i],F1[i,0]))
        cvscores.append(np.mean(F1)* 100)
    print("Overall F1 measure: {:.1.4f}".format(np.mean(F1)))
    K.clear_session()
    gc.collect()

```

```
config = tf.ConfigProto()
config.gpu_options.allow_growth=True
sess = tf.Session(config=config)
K.set_session(sess)
counter += 1
# Saving validation results
scipy.io.savemat('xval_results.mat',mdict={'cvconfusion': cvconfusion.tolist()})
return model
```

I risultati dei modelli mostrano una F1-Measure media pari a circa 0.80, con i migliori modelli che raggiungono una valutazione di circa 0.90.

I valori di accuratezza raggiungono il 90% e il loss si riduce fino a circa 0.25, senza però tendere definitivamente a zero.

6. Modelli della PhysioNet / CinC Challenge 2018

In questa sezione vengono presentati il modello di esempio attualmente pubblicato nella pagina della Challenge 2018 del sito PhysioNet, e alcuni delle più significative sperimentazioni da me affrontate in questi ultimi mesi di studio.

6.1. Moduli comuni

6.1.1. Prepare Entry

Il modulo “PrepareEntry.py” si occupa di invocare gli altri moduli e definisce lo scheletro principale del progetto.

Questo componente stampa su console gli istanti di inizio e fine e monitora la durata complessiva dell’elaborazione.

Nell’ordine vengono chiamate le seguenti funzioni:

- train: funzione che si occupa di allenare il modello attraverso lo scorrimento di tutto il dataset di training.
- score_training_set: una volta ultimata la fase di addestramento il dataset di training viene ripercorso in modalità predizione, viene prodotto un tracciato di probabilità di apnea per ogni istante del record e infine viene calcolato il valore complessivo dell’indicatore AUPRC utilizzato per l’attribuzione del punteggio.
- evaluate_test_set: questa funzione utilizza il modello in modalità predizione e quindi provvede a generare un file per ciascun record contenente le probabilità di apnea a ciascun istante
- package_entry: la funzione si occupa di produrre una cartella compressa contenente tutto il codice sorgente, i log, i pesi del modello, i file di probabilità elaborati sia per il training set che per il test set

Infine prima di concludere l’elaborazione questa funzione si occupa di avviare la console grafica Tensorboard.

```
if __name__ == '__main__':
    os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
    L.init_logger(T.p_LOG_FILE)
    dtInizioElaborazione = datetime.datetime.now()
```

```

L.log_info("Execution starts at: " + str(dtInizioElaborazione))

try:
    model=None
    L.log_info("##### TRAIN #####")
    model= train(model)
    L.log_info("##### SCORE TRAINING SET #####")
    model = score_training_set(model) ("#####")
    L.log_info("##### EVALUATE TEST SET #####")
    evaluate_test_set()
    L.log_info("##### PACKAGE ENTRY #####")
    package_entry()
    dtFineElaborazione = datetime.datetime.now()
    elapsedTime = dtFineElaborazione - dtInizioElaborazione
    L.log_info("Execution stops at: " + str(dtInizioElaborazione))
    L.log_info(" Elapsed Time: " + str(elapsedTime))
    print("Launching Tensorboard...")
    os.system('tensorboard --logdir=tensorboard/' +
str(T.p_TENSORBOARD_LOGDIR))

except:
    dtFineElaborazione = datetime.datetime.now()
    elapsedTime = dtFineElaborazione - dtInizioElaborazione
    L.log_info("Execution stops with errors at: " + str(dtInizioElaborazione))
    L.log_info(" Elapsed Time: " + str(elapsedTime))
    raise

```

La funzione *train()* si occupa di invocare il metodo di inizializzazione del modello T per eliminare i dati dei run precedenti, ed inizializzare i file di log. Quindi con un ciclo for va a leggere l'elenco dei record presenti del dataset e passa ciascuno di essi al classificatore per l'addestramento.

```

# -----
# Generate the data to train the classifier
# -----
def train():
    T.init()
    # Generate a data frame that points to the challenge files
    tr_files, te_files = phyc.get_files()
    # For each subject in the training set...
    for i in range(0, np.size(tr_files, 0)):
```

```

gc.collect()
print('Preprocessing training subject: %d/%d'
      % (i + 1, np.size(tr_files, 0)))
record_name = tr_files.header.values[i][:-4]
T.preprocess_record(record_name)
T.finish()

```

La funzione `score_training_set()` si occupa di prendere in esame uno per volta i record del training set e per ciascuno di essi recuperare il tracciato *arousals*.

Quindi viene recuperato un array *predictions*, della stessa dimensione di *arousals*, contenente le predizioni del modello sullo stesso record. I due array vengono passati al modulo `score` per la valutazione degli indicatori AUROC e AUPRC.

```

#-----
# Run the classifier on each training subject, and compute the mean performance
# ----

def score_training_set():
    # Generate a data frame that points to the challenge files
    tr_files, te_files = phyc.get_files()
    score = Challenge2018Score()
    for i in range(0, np.size(tr_files, 0)):
        gc.collect()
        sys.stdout.write('Evaluating training subject: %d/%d'
                         % (i + 1, np.size(tr_files, 0)))
        sys.stdout.flush()
        record_name = tr_files.header.values[i][:-4]
        predictions = R.classify_record(record_name)
        arousals = phyc.import_arousals(tr_files.arousal.values[i])
        arousals = np.ravel(arousals)
        score.score_record(arousals, predictions, record_name)
        auroc = score.record_auroc(record_name)
        auprc = score.record_auprc(record_name)
        print('AUROC:%f AUPRC:%f' % (auroc, auprc))

    auroc_g = score.gross_auroc()
    auprc_g = score.gross_auprc()
    print('Training AUROC Performance (gross): %f' % auroc_g)
    print('Training AUPRC Performance (gross): %f' % auprc_g)

```

La funzione evaluate_test_set() scorre uno ad uno i record del test set e produce per ciascuno un tracciato testuale con estensione .vec contenente la probabilità di arousal ad ogni istante.

```
# -----
# Run the classifier on each test subject, and save the predictions
# for submission
# -----
def evaluate_test_set():
    # Generate a data frame that points to the challenge files
    tr_files, te_files = phyc.get_files()
    for i in range(0, np.size(te_files, 0)):
        gc.collect()
        print('Evaluating test subject: %d/%d' % (i+1, np.size(te_files, 0)))
        record_name = te_files.header.values[i][-4]
        output_file = os.path.basename(record_name) + '.vec'
        predictions = R.classify_record(record_name)
        np.savetxt(output_file, predictions, fmt='%.3f')
```

La funzione package_entry crea una cartella compressa con tutto il codice sorgente, i modelli, e i file delle predizioni con estensione .vec da sottoporre per la validazione.

```
# -----
# Build a zip file for submission to the Challenge
# -----
def package_entry():
    with ZipFile('entry.zip', 'w', ZIP_DEFLATED) as myzip:
        for dirName, subdirList, fileList in os.walk('.'):
            for fname in fileList:
                if ('.vec' in fname[-4:] or '.py' in fname[-3:]):
                    or '.pkl' in fname[-4:] or '.txt' in fname[-4:]):
                    myzip.write(os.path.join(dirName, fname))
```

6.1.2. Il modulo di acquisizione dei segnali

Il modulo Python *physionetchallenge2018_lib.py*, referenziato negli altri moduli con l'abbreviazione *phyC*, raccoglie tutte le funzioni di accesso ai segnali di input ed al segnale arousals.

Di seguito sono descritte le funzioni principali di questo modulo.

In questo modulo è definito nella variabile *p_DATASET_DIR* il percorso di base del dataset da analizzare.

La funzione *get_files()* si occupa di verificare la directory di base e restituire un elenco ordinato dei record del training set e del test set. La funzione può essere personalizzata in base all'utilizzo del modello su ambiente Unix o Windows, indicando il separatore nella variabile *sep*.

```
# -----
# returns a list of the training and testing file locations for easier import
# -----
def get_files():
    sep = "\\" #windows
    # sep = "/" unix
    header_loc, arousal_loc, signal_loc, is_training = [], [], [], []
    rootDir = p_DATASET_DIR
    print("eseguo phic get files: " + str(rootDir))
    for dirName, subdirList, fileList in sorted(os.walk(rootDir, followlinks=True)): #sorted(
        if dirName != rootDir and dirName != rootDir+sep+'test' and dirName != rootDir+sep+'training':
            if dirName.startswith(rootDir+sep+'training'+sep+"):
                is_training.append(True)
            for fname in fileList:
                if '.hea' in fname:
                    header_loc.append(dirName + sep + fname)
                if '-arousal.mat' in fname:
                    arousal_loc.append(dirName + sep + fname)
                if 'mat' in fname and 'arousal' not in fname:
                    signal_loc.append(dirName + sep + fname)

    elif dirName.startswith(rootDir+sep+'test'+sep+"):
        is_training.append(False)
        arousal_loc.append("")
```

```

for fname in fileList:
    if '.hea' in fname:
        header_loc.append(dirName + sep + fname)
    if 'mat' in fname and 'arousal' not in fname:
        signal_loc.append(dirName + sep + fname)

    # combine into a data frame
    data_locations = {'header': header_loc,
                      'arousal': arousal_loc,
                      'signal': signal_loc,
                      'is_training': is_training
                     }
    # Convert to a data-frame
    df = pd.DataFrame(data=data_locations)

    # Split the data frame into training and testing sets.
    tr_ind = list(find(df.is_training.values))
    te_ind = list(find(df.is_training.values == False))

    training_files = df.loc[tr_ind, :]
    testing_files = df.loc[te_ind, :]

return training_files, testing_files

```

Le funzioni *import_signals* e *import_arousals* prendo in input il percorso di un file corrispondente ad un record e restituisco un oggetto rispettivamente i tracciati di input ed il tracciato delle apnee. I primi dati sono estratti con la libreria scipy, mentre il secondo con libreria h5py, in quanto pur essendo entrambi file in formato MATLAB variano in maniera considerevole le versioni e gli strumenti per la loro acquisizione

```

# -----
# import the outcome vector, given the file name.
# e.g. /training/tr04-0808/tr04-0808-arousal.mat
# -----
def import_arousals(file_name):
    import h5py

```

```

import numpy
f = h5py.File(file_name, 'r')
arousals = numpy.array(f['data']['arousals'])
return arousals

def import_signals(file_name):
    return np.transpose(scipy.io.loadmat(file_name)['val'])

```

6.1.3. Parti comuni: Il modulo di attribuzione del punteggio

Il modulo di attribuzione del punteggio *score2018.py* implementa una classe *Challenge2018Score*. La modalità di calcolo prevede sia il calcolo dei valori AUPRC e AUROC per ciascun record, sia il calcolo degli stessi valori sull'intero dataset. E' opportuno precisare che il calcolo viene effettuato memorizzando tutti i valori di tp, fp, tn, fn (true positive, false positive, true negative, false negative) e poi eseguendo un calcolo complessivo; questo perché i diversi record sono caratterizzati da una lunghezza differente, e quindi la media degli indicatori dei vari record non sarebbe uno stimatore abbastanza preciso.

Vengono descritti di seguito alcuni dei metodi di questa classe, anche al fine di esplicitare meglio la modalità di calcolo del punteggio.

Come si può osservare tutti i valori che nel tracciato arousals (qui riportato come variabile truth) vengono semplicemente esclusi dal calcolo degli indicatori.

I valori predetti in corrispondenza di elementi 0 e 1 del tracciato truth) sono salvati in due sub array pred_pos e pred_neg, che vengono passati come parametri alla funzione *_auc*.

```

def score_record(self, truth, predictions, record_name=None):
    """Add results for a given record to the buffer.
    - 'truth' is a vector of arousal values: zero for non-arousal
    regions, positive for target arousal regions, and negative for
    unscored regions.
    - 'predictions' is a vector of probabilities produced by the
    classification algorithm being tested. This vector must be
    the same length as 'truth', and each value must be between 0
    and 1.
    If 'record_name' is specified, it can be used to obtain

```

```

per-record scores afterwards, by calling record_auroc() and
record_auprc().

"""

# Check if length is correct
if len(predictions) != len(truth):
    raise ValueError("length of 'predictions' does not match 'truth'")

# Compute the histogram of all input probabilities
b = self._scale + 1
r = (-0.5 / self._scale, 1.0 + 0.5 / self._scale)
all_values = numpy.histogram(predictions, bins=b, range=r)[0]

# Check if input contains any out-of-bounds or NaN values
# (which are ignored by numpy.histogram)
if numpy.sum(all_values) != len(predictions):
    raise ValueError("invalid values in 'predictions'")

# Compute the histogram of probabilities within arousal regions
pred_pos = predictions[truth > 0]
pos_values = numpy.histogram(pred_pos, bins=b, range=r)[0]

# Compute the histogram of probabilities within unscored regions
pred_ign = predictions[truth < 0]
ign_values = numpy.histogram(pred_ign, bins=b, range=r)[0]

# Compute the histogram of probabilities in non-arousal regions,
# given the above
neg_values = all_values - pos_values - ign_values
self._pos_values += pos_values
self._neg_values += neg_values

if record_name is not None:
    self._record_auc[record_name] = self._auc(pos_values, neg_values)

```

La funzione `_auc(pos_values, neg_values)` prende in ingresso i gli array delle probabilità indicate in caso di valore 0 e 1 effettivo nel vettore arousals (truth) e restituisce i valori dell'Area Under ROC e PR. La tipologia di calcolo adottata dalla funzione è di tipo iterativo. Per ogni elemento dei due vettori viene calcolata la TPR (frazione dei veri positivi), la TNR (frazione dei veri negativi). La Figura 52 rappresenta graficamente la composizione dei due indicatori, con il riquadro rosso interno al numeratore e il riquadro più grande al denominatore.

		valore vero		totale			valore vero		totale	
		<i>p</i>	<i>n</i>				<i>p</i>	<i>n</i>		
predizione	risultato	<i>p'</i>	Vero Positivo	Falso Positivo			<i>p'</i>	Vero Positivo	Falso Positivo	
		<i>n'</i>	Falso Negativo	Vero Negativo			<i>n'</i>	Falso Negativo	Vero Negativo	
totale		P	N	totale		P	N	totale		

Figura 52 Rappresentazione degli indicatori intermedi TPR e TNR

Il valore di precision è dato $PPV = TP / (tp+fp)$, il valore di Recall corrisponde a TPR definito sopra.

Ad ogni iterazione viene calcolato il valore di questi indicatori e sommato iterativamente a quelli precedenti per definire la curva. Il risultato della sommatoria sono i valori AUROC e AUPRC.

```

def _auc(self, pos_values, neg_values):
    # Calculate areas under the ROC and PR curves by iterating
    # over the possible threshold values.

    # At the minimum threshold value, all samples are classified as
    # positive, and thus TPR = 1 and TNR = 0.
    tp = numpy.sum(pos_values)
    fp = numpy.sum(neg_values)
    tn = fn = 0
    tpr = 1
    tnr = 0
    if tp == 0 or fp == 0:
        # If either class is empty, scores are undefined.
        return (float('nan'), float('nan'))
    ppv = float(tp) / (tp + fp)
    auroc = 0
    auprc = 0

    # As the threshold increases, TP decreases (and FN increases)
    # by pos_values[i], while TN increases (and FP decreases) by
    # neg_values[i].
    for (n_pos, n_neg) in zip(pos_values, neg_values):
        tp -= n_pos
        tn += n_pos
        fn += n_neg
        fp -= n_neg
        tpr = float(tp) / (tp + fn)
        tnr = float(tn) / (tn + fp)
        ppv = float(tp) / (tp + fp)
        auroc += tpr * tnr
        auprc += tpr * ppv
    return (auroc, auprc)

```

```

fn += n_pos
fp -= n_neg
tn += n_neg
tpr_prev = tpr
tnr_prev = tnr
ppv_prev = ppv
tpr = float(tp) / (tp + fn)
tnr = float(tn) / (tn + fp)
if tp + fp > 0:
    ppv = float(tp) / (tp + fp)
else:
    ppv = ppv_prev
auroc += (tpr_prev - tpr) * (tnr + tnr_prev) * 0.5
auprc += (tpr_prev - tpr) * ppv_prev
return (auroc, auprc)

```

6.1.1. Parti comuni: Il modulo di log

Considerata la vasta quantità di log prodotta durante la fase di training, si è scelto di generare un riepilogo del log su file contenente solo i passi principali dell'addestramento, i valori di AUROC e AUPRC rilevati per ciascun record e quelli complessivi.

Questo modulo si occupa di creare la cartella *logs* se non esiste, e quindi di riportare sul file ogni messaggio ricevuto in input ai vari livelli di priorità.

L'implementazione si basa sulla libreria standard Python, ma si è reso necessario centralizzare la funzione di log per poter avere un unico file di log condiviso in tutti i punti dell'applicazione.

Si riporta una parte del codice con la funzione di inizializzazione, che imposta il formato del log, ed il metodo di stampa a livello info, che effettua sia una print su console che una scrittura su file.

```

def init_logger(filename):
    try:
        os.mkdir('logs')
    except OSError:
        pass
    for f in glob.glob("logs/*"):

```

```

os.remove(f)

logging.basicConfig(filename="logs/" + str(filename),
    level=logging.DEBUG,
    format='%(levelname)s: %(asctime)s %(message)s',
    datefmt='%m/%d/%Y %H:%M:%S')

def log_info(message):
    print(message)
    logging.info(message)

```

6.2. Descrizione Modello di esempio

Il modello di esempio¹¹ è costituito da una regressione logistica, un caso specifico di modello lineare caratterizzato dall'utilizzo del logit come funzione di link. Questo modello di regressione si applica a tutti quei casi dove il vettore target y è di tipo dicotomico (vero – falso, 0 -1, ecc.).

La funzione `init()` si occupa di rimuovere i dati delle esecuzioni precedenti dalle cartelle:

- logs: contiene i log testuali dell'esecuzione
- models: contiene i pesi del modello addestrato
- tensorboard: contiene i dati di log nel formato adeguato per la visualizzazione nella console TensorBoard
- training_output: contiene un file con estensione .vec per ogni record del dataset di training in cui viene riportata la probabilità di arousal ad ogni istante
- training_input: (quando utilizzato per debug) contiene una copia in formato testuale dei valori dei segnali letti dal formato MATLAB .mat

¹¹ I sorgenti del modello sono scaricabili dal sito Physionet a link <https://physionet.org/challenge/2018/sample.zip>, ne ho riportato una copia con alcuni miglioramenti nelle funzioni di log nel mio profilo GitHub in <https://github.com/esanna-unimarconi/cinc-challenge2018-sample2>

- `test_output`: contiene un file con estensione `.vec` per ogni record del dataset di test in cui viene riportata la probabilità di arousal ad ogni istante

La funzione `preprocess_record()` sotto riportata si occupa in primo luogo di recuperare il record e il rispettivo tracciato arousals secondo il nome fornito in input.

Quindi per semplicità viene scelto di concentrare l'analisi su uno solo dei segnali presenti nel record, ovvero il segnale SaO₂(saturazione ossigeno). Viene definita una finestra temporale di 60 secondi nella quale viene calcolata la variazione di questo segnale. I valori della varianza vengono quindi passati all'oggetto `sklearn.linear_model.LogisticRegression`. L'implementazione di questo oggetto non supporta l'utilizzo della GPU e quindi questo modello sebbene estremamente semplificato ha avuto un tempo di elaborazione complessiva sul dataset completo di oltre 10 giorni e 9 ore.

Il modello provvede in primo luogo al recupero dei signali e del tracciato di target arousal sotto forma di oggetto Pandas Dataframe. Essendo quest'ultimo gestibile come array associativo, può estrarre due array rispettivamente per il solo segnale di saturazione dell'ossigeno SaO₂ e per il tracciato arousals delle apnee.

A questo punto per ridurre la mole di dati da passare al modello `LogisticRegression` viene suddiviso il segnale, che contiene approssimativamente la registrazione di una notte di sonno, in intervalli di 60 secondi. Per ciascun intervallo vengono estratte le seguenti caratteristiche:

- per il segnale SaO₂ viene calcolata la varianza in ogni intervallo di 60 secondi
- per il tracciato arousals viene riportato il valore massimo.

La scelta di selezionare il valore massimo per il tracciato di target potrebbe a prima vista sembrare molto approssimativa, ma così non è alla luce delle seguenti considerazioni:

- i valori -1 possono essere classificati indifferentemente come 0 o 1 ai fini dell'attribuzione del punteggio finale
- i valori 1 costituiscono una porzione molto ridotta del totale del record (in media sotto il 10%)
- trattandosi di registrazione temporale di fenomeni fisiologici, le regioni di arousal non sono molto sparse, ovvero i valori 1 non sono distribuiti in maniera omogenea nel record ma quando si presenta un valore 1, il tracciato arousal mantiene il valore 1 per un numero di secondi abbastanza alto.

I nuovi array predisposti con questa operazioni saranno quindi ridotti a 1/12000 della dimensione precedente, passando da una numerosità di campioni di circa 4-5 milioni a quella indicativa di 500 campioni.

Completata questa operazione di preprocessazione viene preso in esame il nuovo array arousals ridotto, per verificare se questo contiene almeno un valore a 1. Se questa condizione non è verificata, si salta l'addestramento per questo record, in quanto non ha nessuna utilità ai fini dell'apprendimento.

Viene quindi istanziato un nuovo oggetto *LogisticRegression* a cui vengono passati i valori di X (SaO₂) e y (arousals) tramite il metodo fit. Il risultato dei pesi viene salvato all'interno della cartella models tramite del metodo joblib.dump.

Si osservi come in questo modello di esempio si sceglie di far ripartire il classificatore da zero per ogni record, e produrre una versione dei pesi per ciascuno di essi.

```
def preprocess_record(record_name):
    header_file = record_name + '.hea'
    signal_file = record_name + '.mat'
    arousal_file = record_name + '-arousal.mat'

    # Get the signal names from the header file
    signal_names, Fs, n_samples = phyc.import_signal_names(header_file)
    signal_names = list(np.append(signal_names, 'arousals'))

    # Convert this subject's data into a pandas dataframe
```

```

this_data = phyc.get_subject_data(arousal_file, signal_file, signal_names)

# -----
# Generate the Features for the classification model - variance of SaO2
# -----
# For the baseline, let's only look at how SaO2 might predict arousals
SaO2 = this_data.get(['SaO2']).values
arousals = this_data.get(['arousals']).values

# We select a window size of 60 seconds with no overlap to compute
# the features
step      = Fs * 60
window_size = Fs * 60

# Initialize the matrices that store our training data
X_subj = np.zeros([(n_samples) // step], 1)
Y_subj = np.zeros([(n_samples) // step], 1)

# Extract the variance of the SaO2 in 60 second windows as a feature
for idx, k in enumerate(range(0, (n_samples-step+1), step)):
    X_subj[idx, 0] = np.var(np.transpose(SaO2[k:k+window_size]), axis=1)
    Y_subj[idx]   = np.max(arousals[k:k+window_size])

# Ignore records that do not contain any arousals
if not np.any(Y_subj):
    sys.stderr.write('no arousals found in %s\n' % record_name)
    return
# -----
# Train a (multi-class) Logistic Regression classifier
# -----
my_classifier = LogisticRegression()
my_classifier.fit(X_subj, np.ravel(Y_subj))

# !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
# Save this algorithm for submission to Physionet Challenge:
# !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
model_file = 'models/%s_model.pkl' % os.path.basename(record_name)
joblib.dump(my_classifier, model_file)

```

Si riporta ora un'analisi dettagliata della modalità di predizione del modello una volta addestrato, che viene invocata dal metodo `PrepareEntry` prima su tutti i record del training set e poi su quelli del test set attraverso l'invocazione del metodo `classify_record`.

Il processo di classificazione è implementato in modo iterativo: la funzione recupera tutti i file di salvataggio dei modelli di ciascun record su cui è stato effettuato l'addestramento, e li carica nella lista `model_list`.

A questo punto itera questa lista e invoca il metodo `run_classifier` per recuperare un array con le predizioni di ciascun istante del record; la predizione finale è quindi la media delle predizioni, calcolata elemento per elemento.

```
def classify_record(record_name):
    header_file = record_name + '.hea'
    signal_file = record_name + '.mat'

    # Read model files from the 'models' subdirectory, which are
    # generated by 'train_classifier.py'
    model_list = []
    for f in glob.glob('models/*_model.pkl'):
        model_list.append(f)

    # Use the average predictions from the models generated on the
    # training set
    predictions_mean = 0.
    for j in range(0, len(model_list)):
        this_model = model_list[j]
        predictions = run_classifier(header_file, signal_file, this_model)
        print("Predictions shape before: " + str(predictions.shape))
        predictions_mean += predictions

    print("predictions_mean" + str(predictions_mean))
    predictions_mean /= len(model_list)
    print("predictions_mean" + str(predictions_mean))
    # !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

    # Return a vector of per-sample predictions, as per challenge requirements
    # !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

    return predictions_mean
```

I risultati complessivi del modello di esempio sul dataset di training sono i seguenti:

- Tempo di elaborazione sul dataset completo pari a circa 10 giorni e 9 ore
- AUPRC = 0.125
- AUROC = 0.718

I risultati complessivi pubblicati¹² misurati sul dataset di test e resi disponibili sul sito Physionet per il modello sample sono i seguenti:

- AUPRC = 0.088
- AUROC = 0.700

Si ricorda che al momento non è ancora possibile effettuare misurazioni in autonomia sul test set in quanto il tracciato arousal non è ancora disponibile fino al termine della challenge. Tuttavia come prevedibile il risultato ottenuto sul dataset di test è inferiore rispetto a quello ottenuto sul dataset di training.

6.3. Descrizione Modello Keras con Dense

Il primo approccio della sperimentazione è stato quello di cercare di riprodurre il modello di esempio con le librerie TensorFlow e Keras. La regressione logistica può essere vista come un caso particolare una di un layer fully connected con una sola classe.

Il modello è riportato nel modulo *train_model_LR.py* del progetto cinc-challenge2018-sannae-logisticRegression¹³.

Il metodo init si occupa di impostare i seguenti parametri:

- Fs= 200, frequenza di campionamento dei segnali a 200Hz;
- p_WINDOW_SIZE =x*Fs = 600, finestra di dimensione 60 secondi del segnale passato al modello;

¹² Il tracciato target arousal non sarà disponibile fino al termine della competizione, pertanto la performance sul dataset di test non può essere autonomamente rilevata.

¹³ copia del progetto è stata resa disponibile sul mio account GitHub in <https://github.com/esanna-unimarconi/cinc-challenge2018-sannae-logisticRegression>

- p_INPUT_FEAT = 1, numero di segnali in input (solo SaO₂, come nel modello di esempio calcolo come feature il valore della varianza in 60 secondi);
- p_OUTPUT_CLASS = 1 vengono eliminati tutti i valori -1 dal training, in quanto non utili alla predizione, e viene passata un'unica classe con valori compresi tra 0,1;
- p_BATCH_SIZE= 400, numero di campioni per volta passati al modello;
- p_EPOCHS=120, epoche ovvero numero di volte che lo stesso report viene fatto ripassare durante il training per migliorare l'addestramento;
- p_MODEL_FILE = 'LOG_REGR_input_1_w600_b400_e120.hdf5': nome del file dove salvo i pesi del modello;
- p_DATASET_DIR = '/opt/PHYSIONET/ch2018', nome della directory in cui è memorizzato il dataset;
- p_LOG_FILE = 'LOG_REGR_input_1_w600_b400_e120.log' il nome del file di log testuale;
- p_KERAS_LOG_FILE = 'LOG_REGR_input_1_w600_b400_e120_Keras.log', nome del file temporaneo dove viene salvato il log interno di Keras;
- p_TENSORBOARD_LOGDIR = 'LOG_REGR_input_1_w600_b400_e120', directory di scrittura dei dati per la visualizzazione nella console Tensorboard.

Inoltre si occupa di svuotare le seguenti cartelle dai dati delle esecuzioni precedenti come già descritto per il modello sample.

Analizziamo la funzione preprocess_record() che si occupa di provvedere all'addestramento del modello su ogni record.

Non si è scelto di unire tutti i record campionati in un unico file a causa della grande dimensione dei dati, si ricorda che il solo dataset di test ha una dimensione complessiva su Disco di 144GB. La funzione viene invocata in

maniera distinta per ogni record, la cui dimensione indicativa è di circa 130-150MB.

In primo luogo vengono estratti gli array dei segnali e degli arousals con il metodo *loaddata*

In maniera analoga al modello di esempio viene calcolata la varianza in 60 secondi per il solo segnale SaO2 e il valore massimo in 60 secondi per il tracciato arousals.

Quindi viene estratto un sottoinsieme dei valori X_subj, Y_subj da passare per la validazione dei dati tra un batch e l'altro nelle variabili X_val, Y_val.

Di seguito vengono definite alcune funzioni di callback del modello:

- EarlyStopping: funzione che provvede all'arresto anticipato dell'iterazione in corso quando il modello si accorge che non ci sono più miglioramenti nel loss. L'attributo patience uguale a tre corrisponde al numero di epoche consecutive che vengono valutate prima di decidere lo stop.
- ModelCheckpoint: funzione che si occupa di aggiornare il file dei pesi del modello quando si verifica un miglioramento nel loss

A questo punto si provvede ad istanziare il modello, se non già esistente. In questo caso si è optato per una scelta differente rispetto al modello di sample dove per ogni record viene istanziato un modello diverso con pesi differenti. Al contrario in questo caso, il modello viene restituito e condiviso per valore tra le funzioni, in modo da evitare di doverlo ricreare da zero per ogni record e di dover ricaricare i suoi pesi. Questa scelta è motivata dal fatto che l'operazione di creazione del modello era molto onerosa, ed essendo un'operazione che non viene gestita dalla GPU si caratterizzava come un collo di bottiglia nell'elaborazione.

L'addestramento vero e proprio del modello avviene anche in questo caso con l'invocazione del metodo fit, al quali vengono passati oltre ai valori X, y, alcuni parametri relativi a:

- dimensione del batch, ovvero intervallo di elementi da processare prima di aggiornare le varie metriche
- epoche: numero di volte in cui lo stesso record deve essere analizzato.
- l'elenco delle callback da eseguire

Alla fine della processazione di ogni record salvo i pesi del modello.

```
def preprocess_record(record_name, model):
    SaO2, arousals, recordLength = loaddata(record_name)

    # Ignore records that do not contain any arousals
    if 1 not in arousals:
        L.log_info('no arousals found in %s\n' % record_name)
        return

    # We select a window size of 60 seconds with no overlap to compute
    # the features
    step=p_WINDOW_SIZE
    window_size=p_WINDOW_SIZE
    n_samples = recordLength

    # Initialize the matrices that store our training data
    X_subj = np.zeros([(n_samples) // step, 1])
    Y_subj = np.zeros([(n_samples) // step, 1])

    for idx, k in enumerate(range(0, (n_samples-step+1), step)):
        X_subj[idx, 0] = np.var(np.transpose(SaO2[k:k+window_size]), axis=1)
        Y_subj[idx] = np.max(arousals[k:k+window_size])

    #saving signals and output array to file for debug purpose
    #output_fileX_subj = "training_input/X_subj" +
    #                      os.path.basename(record_name) + '.vec'
    #output_fileY_subj = "training_input/Y_subj" +
    #                      os.path.basename(record_name) + '.vec'
    #np.savetxt(output_fileX_subj, X_subj, fmt='%.3f')
    #np.savetxt(output_fileY_subj, np.ravel(Y_subj), fmt='%.3f')

    # split train and validation sets
    idxval_start = np.random.randint(np.trunc(recordLength/2), size=1)[0]
```

```

idxval_size = np.random.randint(np.trunc(recordLength / 2), size=1)[0]
Xval = X_subj[idxval_start:idxval_start+idxval_size, :, :]
Yval = Y_subj[idxval_start:idxval_start + idxval_size, :]

#callbacks
callbacks = [
    # Early stopping definition
    EarlyStopping(monitor='val_loss', patience=3, verbose=1),
    # Saving best model
    ModelCheckpoint('models/'+str(os.path.basename(record_name))+str(p_MODEL_FILE),
    monitor='val_loss', save_best_only=True,
        verbose=1),
    CSVLogger('logs/'+p_KERAS_LOG_FILE, separator=',', append=False),
    TensorBoard(log_dir='tensorboard/'+str(p_TENSORBOARD_LOGDIR),
    histogram_freq=0, batch_size=32, write_graph=True,
        write_grads=False, write_images=False, embeddings_freq=0,
        embeddings_layer_names=None, embeddings_metadata=None,
    embeddings_data=None)
]
if (model is None):
    model =LogisticRegressionKeras()

#fit model
model.fit(X_subj, np.ravel(Y_subj), validation_data=(X_subj, np.ravel(Y_subj)),
epochs=p_EPOCHS,
    batch_size=p_BATCH_SIZE, callbacks=callbacks)

#saving model wieghts.
try:
model.save_weights('models/'+str(os.path.basename(record_name))+str(p_MODEL_FIL
E))
except:
    L.log_info("Errore imprevisto: non riesco a salvare i pesi.")
return model

```

Dopo aver descritto il comportamento generale della funzione di training andiamo ora a descrivere in modo più dettagliato la struttura del modello, che viene creato dalla funzione *LogisticRegressionKeras()*.

La funzione istanzia un modello sequenziale con la chiamata della funzione *Sequential()*, e quindi aggiunge un layer densamente connesso con *Dense()*.

La funzione di attivazione selezionata è *hard_sigmoid*, in quanto quella che sembra insieme a Rectified Linear Unit (ReLU) dare le migliori prestazioni.

L'optimizer utilizzato è il SGD (Stochastic gradient descent) e la funzione di costo è la binary_crossentropy.

```
"""
* Model Definition
"""

def LogisticRegressionKeras():
    print('Build model LogisticRegression...')
    model = Sequential()
    model.add(Dense(p_OUTPUT_SIZE, kernel_regularizer=l2(0.), input_shape=(1,),
                   activation='hard_sigmoid')) # better perform relu or hard_sigmoid

    model.compile(optimizer='sgd', loss='binary_crossentropy', metrics=[auc_pr, auc_roc,
    'accuracy'])

    return model
```

In relazione alle metriche misurate, affianco alla standard accuracy sono state aggiunte le metriche per AUROC e AUPRC. Queste metriche non sono ancora supportate nativamente in Keras pertanto è necessario inserire il seguente codice di wrapping che permette l'utilizzo di metriche definite nel modulo TensorFlow.metrics.

```
"""
Function for porting Tensorflowmetrics on Keras
"""

def as_keras_metric(method):
    import functools
    from keras import backend as K
    import tensorflow as tf
    @functools.wraps(method)
    def wrapper(self, args, **kwargs):
        """ Wrapper for turning tensorflow metrics into keras metrics """
        return method(self, args, **kwargs)
```

```

value, update_op = method(self, args, **kwargs)
K.get_session().run(tf.local_variables_initializer())
with tf.control_dependencies([update_op]):
    value = tf.identity(value)
return value
return wrapper

@as_keras_metric
def auc_pr(y_true, y_pred, curve='PR'):
    return tf.metrics.auc(y_true, y_pred, curve=curve),
summation_method='careful_interpolation')

@as_keras_metric
def auc_roc(y_true, y_pred, curve='ROC'):
    return tf.metrics.auc(y_true, y_pred, curve=curve),
summation_method='careful_interpolation')

```

Il risultato raggiunto da questo modello sul training set sono i seguenti:

- AUPRC: 0.076
- AUROC: 0.539

Non è possibile provvedere alla verifica sul test set in quanto il tracciato arousal non è stato ancora reso disponibile.

6.4. Descrizione Modello Keras con ResNet

Il modello ResNet è derivato dal modello prodotto dal team di Fernando Andreotti per challenge 2017. Il modello è stato adattato per gestire i 13 canali. Si riporta di seguito la definizione del modello. La descrizione è definita nella sezione 5.2 di questa tesi, con la variazione che nella dimensione delle feature sono inseriti 13 canali.

```

#####
## Model definition      ##
## ResNet based on Rajpurkar  ##
#####
def ResNet_model():
    # Add CNN layers left branch (higher frequencies)
```

```

# Parameters from paper
WINDOW_SIZE=p_WINDOW_SIZE #12000, 60 secondi * 200
INPUT_FEAT = p_INPUT_FEAT # 13 enrico
OUTPUT_CLASS = p_OUTPUT_CLASS # 3 (1,0,-1)

k = 1 # increment every 4th residual block
p = True # pool toggle every other residual block (end with 2^8)
convfilt = 64
convstr = 1
ksize = 16
poolsize = 2
poolstr = 2
drop = 0.5

# Modelling with Functional API
input1 = Input(shape=(WINDOW_SIZE,INPUT_FEAT), name='input')

# instantiate model
## First convolutional block (conv,BN, relu)
x = Conv1D(filters=convfilt,
            kernel_size=ksize,
            padding='same',
            strides=convstr,
            kernel_initializer='he_normal')(input1)
x = BatchNormalization()(x)
x = Activation('relu')(x)

## Second convolutional block (conv, BN, relu, dropout, conv) with residual net
# Left branch (convolutions)
x1 = Conv1D(filters=convfilt,
            kernel_size=ksize,
            padding='same',
            strides=convstr,
            kernel_initializer='he_normal')(x)
x1 = BatchNormalization()(x1)
x1 = Activation('relu')(x1)
x1 = Dropout(drop)(x1)
x1 = Conv1D(filters=convfilt,
            kernel_size=ksize,

```

```

padding='same',
strides=convstr,
kernel_initializer='he_normal')(x1)
x1 = MaxPooling1D(pool_size=poolsizes,
                  strides=poolstr)(x1)
# Right branch, shortcut branch pooling
x2 = MaxPooling1D(pool_size=poolsizes,
                  strides=poolstr)(x)
# Merge both branches
x = keras.layers.add([x1, x2])
del x1,x2

## Main loop
p = not p
for l in range(15):

    if (l%4 == 0) and (l>0): # increment k on every fourth residual block
        k += 1
        # increase depth by 1x1 Convolution case dimension shall change
        xshort = Conv1D(filters=convfilt*k,kernel_size=1)(x)
    else:
        xshort = x
    # Left branch (convolutions)
    # notice the ordering of the operations has changed
    x1 = BatchNormalization()(x)
    x1 = Activation('relu')(x1)
    x1 = Dropout(drop)(x1)
    x1 = Conv1D(filters=convfilt*k,
                 kernel_size=ksize,
                 padding='same',
                 strides=convstr,
                 kernel_initializer='he_normal')(x1)
    x1 = BatchNormalization()(x1)
    x1 = Activation('relu')(x1)
    x1 = Dropout(drop)(x1)
    x1 = Conv1D(filters=convfilt*k,
                 kernel_size=ksize,
                 padding='same',
                 strides=convstr,

```

```

    kernel_initializer='he_normal')(x1)

    if p:
        x1 = MaxPooling1D(pool_size=poolsize,strides=poolstr)(x1)
    # Right branch: shortcut connection

    if p:
        x2 = MaxPooling1D(pool_size=poolsize,strides=poolstr)(xshort)
    else:
        x2 = xshort # pool or identity

    # Merging branches
    x = keras.layers.add([x1, x2])
    # change parameters
    p = not p # toggle pooling

    # Final bit
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = Flatten()(x)
    #x = Dense(1000)(x)
    #x = Dense(1000)(x)
    out = Dense(OUTPUT_CLASS, activation='softmax')(x)
    model = Model(inputs=input1, outputs=out)
    model.compile(optimizer='adam',
                  loss='categorical_crossentropy',
                  metrics=[auc_pr, auc_roc, 'accuracy'])

return model

```

6.5. Descrizione Modello Keras con bidirectional LSTM

Un ulteriore modello sperimentato è stato quello basato su Long Short Term Memory. Nei tentativi di addestramento con porzioni ridotte del dataset si è osservato come ci sia una miglior performance inserendo una pila di due bidirectional LSTM in sequenza. La possibilità di utilizzare più di due livelli si è dimostrata proibitiva in quanto porta a tempi di addestramento troppo lunghi, nell'ordine dei mesi, e non sembra fornire un valore aggiunto secondo quanto sperimentato sui dataset parziali.

Si riporta di seguito una descrizione del modello di base con una sola LSTM non bidirezionale.

```

def LSTM_model():
    print('Build model LSTM...')
    model = Sequential()
    model.add(LSTM(32, dropout=0.2, kernel_regularizer=l2(0.), recurrent_dropout=0.2,
input_shape=(None, p_INPUT_FEAT),return_sequences=True, go_backwards=True))
    model.add(Dense(p_OUTPUT_CLASS, activation='hard_sigmoid')) #better perform
relu or hard_sigmoid
    #model.add(TimeDistributed(Dense(p_OUTPUT_CLASS, activation='sigmoid')))
    # try using different optimizers and different optimizer configs
    model.compile(loss='binary_crossentropy',
                  optimizer='adam',
                  metrics=[auc_pr,auc_roc,'accuracy'])
try:
    model.load_weights('models/'+str(p_MODEL_FILE))
except:
    print("non ho trovato i pesi. parto a zero")
return model

```

Si riporta di seguito una versione del modello con due livelli consecutivi di bidirectional LSTM. La pila è resa possibile dal parametro return_sequences=True che restituisce tutta la serie e non solo il valore dell'ultimo elemento.

```

def Bi_Layer_LSTM_model():
    print('Build model Bidirectional Bi layer LSTM...')
    model = Sequential()
    model.add(
        Bidirectional(LSTM(32, dropout=0.2, kernel_regularizer=l2(0.),
recurrent_dropout=0.2,return_sequences=True,implementation=2),
                     input_shape=(None, p_INPUT_FEAT), merge_mode='concat',))
    model.add(Bidirectional(LSTM(32, dropout=0.2, recurrent_dropout=0.2,
return_sequences=False, implementation=2)))
    model.add(Dense(p_OUTPUT_CLASS, activation='hard_sigmoid')) #better perform
relu or hard_sigmoid
    #model.add(TimeDistributed(Dense(p_OUTPUT_CLASS, activation='sigmoid')))
    # try using different optimizers and different optimizer configs
    model.compile(loss='binary_crossentropy',
                  optimizer='adam',

```

```

metrics=[auc_pr,auc_roc,'accuracy']) #'accuracy'

try:
    model.load_weights('models/'+str(p_MODEL_FILE))
except:
    print("non ho trovato i pesi. parto a zero")
return model

```

6.6. Descrizione Modello MATLAB con bidirectional LSTM

SI è infine prodotto un modello analogo al precedente in linguaggio MATLAB, al fine di verificare se l'utilizzo di funzioni applicate ai segnali possa essere un utile strumento di miglioramento delle performance, così come verificato per il dataset della challenge 2017.

La necessità di utilizzare MATLAB è motivata dall'assenza di una implementazione affidabile delle funzioni instfreq e pentropy. Difatti dopo diversi giorni di lavoro sia alcune implementazioni recuperate in rete che alcune prove di implementazione in Python sulla base delle descrizioni teoriche si sono dimostrate non adeguate rispetto alla versione MATLAB.

I passaggi eseguiti sono stati simili a quelli presentati in precedenza nella analoga soluzione della challenge 2017:

- oversampling per ridurre lo sbilanciamento tra valori 1 e 0;
- applicazione delle funzioni instfreq e pentropy e passaggio da 13 a 26 canali;
- standardizzazione.

Si è inoltre deciso di eliminare tutti i campioni etichettati come -1 (indefinito per rumore), al contrario di quanto viene fatto nel modello di esempio dove questi valori vengono convertiti in arousal (valore = 1).

Per brevità si riporta solo la definizione del modello, si può osservare nello pseudocodice seguente come il livello bilstmLayer attenda 26 canali.

```

layers = [
    sequenceInputLayer(26)
    bilstmLayer(100,'OutputMode','last')
    fullyConnectedLayer(2)
]

```

```

softmaxLayer
classificationLayer
]
options = trainingOptions('adam', ...
    'MaxEpochs',20, ...
    'MiniBatchSize', 150, ...
    'InitialLearnRate', 0.01, ...
    'GradientThreshold', 1, ...
    'plots','none', ... %'training-progress', ...
    'Verbose',true);

```

Per questioni di complessità si è riusciti a testare il modello solo sul dataset ridotto chmini2, che comunque ha dimostrato di essere un sottoinsieme sufficiente per la stima delle prestazioni. Si consideri che il modello MATLAB è stato eseguito su un normale computer non dotati di GPU in quanto parallelamente la workstation era dedicata al 100% delle sue risorse nella elaborazione dei modelli precedenti.

Il primo test effettuato su questo modello ha preso in esame sul dataset chmini (4 record) il solo canale ECG, per verificare la differenza tra l'utilizzo o meno delle funzioni pentropy e instfreq.

Il risultato di questo primo esperimento ha mostrato un miglioramento, con un valore di AUPRC pari a 0,41, rispetto a circa 0,3 dello stesso modello senza l'applicazione delle due funzioni, e rispetto a circa 0,31 del modello di esempio MATLAB.

Si è quindi effettuato un secondo test con i canali SaO2 e ECG ai quali sono state applicati in ordine le due funzioni, l'oversampling e la standardizzazione. Il risultato di questo modello sul dataset chmini è stato pari a 0,66, ovvero un valore più che doppio rispetto al modello di esempio.

Il terzo test ha applicato lo stesso modello al dataset chmini2, ottenendo una performance pari a circa 0,87.

Si è quindi eseguito un quarto e ultimo test con tutti e 26 i canali (13 canali sdoppiati dall'applicazione delle funzioni `instfreq` e `pentropy`) sul dataset `chmini2`, ottenendo un valore di AUPRC pari a circa, 0,7

7. Analisi dei risultati

In questo capitolo vengono riepilogati e confrontati i risultati dei differenti modelli descritti nelle sezioni precedenti.

7.1. Risultati modelli CinC challenge 2017

Si riportano di seguito nella Tabella 11 i risultati ufficiali della challenge 2017:

Tabella 11 Risultati ufficiali CinC Challenge 2017

Ran k	Score (rounded)	Authors
1	0.83	Tomás Teijeiro et al.
1	0.83	Shreyasi Datta et. al.
1	0.83	Morteza Zabihi et al.
1	0.83	Shenda Hong et al.
5	0.82	Mohammed Baydoun et al.
5	0.82	Martin Zihlmann et al.
5	0.82	Guangyu Bin et al.
5	0.82	Zhaohan Xiong et al.
9	0.81	Martin Kropf et al.
9	0.81	Filip Plesinger et al.
9	0.81	Sebastian D. Goodfellow et al.
9	0.81	Maurizio Varanini et al.
9	0.81	Ashish Sharma et al.
9	0.81	Radovan Smíšek et al.
9	0.81	Chen Yao et al.
9	0.81	Marcus Vollmer et al.
9	0.81	David Smoleň
18	0.80	Chen Jiayu et al.
18	0.80	Joachim A. Behar et al.
18	0.80	Ivaylo Christov et al.
18	0.80	Dionisije Sopic et al.
18	0.80	Philip A. Warrick et al.
18	0.80	Jonathan Rubin et al.
18	0.80	Gliner Vadim et al.
25	0.79	Sasan Yazdani et al.
25	0.79	Marco Delai et al.
25	0.79	Csaba Botos et al.
25	0.79	Fernando Andreotti et al.
29	0.78	Vykintas Maknickas et al.
29	0.78	Christoph Hoog Antink et al.
29	0.78	Nadi Sadr et al.
29	0.78	Maximilian Oremek et al.
29	0.78	Runnan He et al.
34	0.77	Bradley M Whitaker et al.
34	0.77	Elena Simarro Mondéjar et al.
34	0.77	Mohamed Limam et al.
34	0.77	Miguel Lozano et al.
38	0.76	Teo Soo-Kng et al.
39	0.75	Zhenning Mei et al.
39	0.75	Babak Afshin-Pour et al.
39	0.75	Joel Karel et al.
39	0.75	Rymko et al.

39	0.75	Katarzyna Stepien et al.
44	0.74	Griet Goovaerts et al.
45	0.73	Javier de la Torre Costa et al.
45	0.73	Heikki Väänänen et al.
45	0.73	Pedro Alvarez et al.
45	0.73	Victor Manuel José Ocoa
45	0.73	Manuel García et al.
50	0.72	Vignesh Kalidas
51	0.71	Jos van der Westhuizen
51	0.71	Kamran Kiani et al.
53	0.69	Ahmad B. A. Hassanat et al.
54	0.68	Philip Aston et al.
55	0.64	Irena Jekova et al.
56	0.63	Lluís Borràs Ferrís et al.
57	0.61	Ilya Potapov et al.
57	0.61	Ruhallah Amandi M et al.
59	0.58	DA SILVA—FILARDER Matthieu et al.
60	0.56	María Rebeca Lliguin León et al.
61	0.55	Erin Coppola
62	0.53	Carlos Fambuena Santos et al.
62	0.53	Octavian-Lucian Hasna et al.
64	0.51	Ines Chavarria Marques; et al.
65	0.50	Mihalis Nicolaou et al.
66	0.48	Sergey S. Krivenko
	0.41	(Sample entry)
67	0.25	Raviteja Mullapudi et al.

7.1.1. Risultati modelli LSTM MATLAB e Keras

I due modelli basati su bidirectional LSTM descritti nelle sezioni precedenti hanno dimostrato come l'estrazione di alcune feature (momenti tempo-frequenza come frequenza istantanea e entropia spettrale) possono costituire un dataset più caratterizzante rispetto al segnale originario.

Entrambi i modelli hanno preso in considerazione solo la porzione di dataset afferente alle classi Norma (N) e Atrial Fibrillation (A). Il modello MATLAB ha ottenuto un F1-measure pari a circa il 56%, valore che è stato incrementato fino al 93%. La reimplementazione dello stesso modello con Keras e Tensorflow ha prodotto una F1-measure che durante 100 ripetizioni ha ottenuto una F1-measure media pari a circa l'80%, con picchi vicini al 90% in una parte delle esecuzioni. L'approccio di sostituire il segnale con alcune sue funzioni in questo caso si è dimostrato vincente.

7.2. Risultati modelli CinC challenge 2018

In questa sezione vengono riepilogati e analizzati i risultati della challenge 2018.

Per ridurre i tempi di elaborazione ed avere dei risultati parziali sui modelli si sono definiti due sottoinsiemi del dataset:

- ch2018_mini: costituito da 4 record del training set e 2 record del test set, pari a circa lo 0,4% del totale
- ch2018_mini2: costituito da 50 record del training set e 10 record del test set, pari a circa il 5% del totale.

Nella Tabella 12 vengono riepilogati i risultati ottenuti dai diversi modelli descritti nei capitoli precedenti.

Tabella 12 CinC Challenge 2018, risultati dei modelli sperimentati

DATASET	MODELLO	TRAINING AUROC	TRAINING AUPRC
ch2018	1 - sample	0,718	0,125
ch2018	2 - LRegressionK	0,500	0,076
ch2018	3 - ReNet	0,490	0,068
ch2018	4 - LSTM	0,500	0,070
ch2018_mini2	1 - sample	0,688	0,120
ch2018_mini2	2 - LRegressionK	0,534	0,082
ch2018_mini2	3 - ReNet	0,570	0,093
ch2018_mini2	4 - LSTM	0,670	0,109
ch2018_mini	1 - sample	0,219	0,025
ch2018_mini	2 - LRegressionK	0,474	0,041
ch2018_mini	3 - ReNet	0,617	0,078
ch2018_mini	4 - LSTM	0,670	0,109

I valori ufficiali provvisori della challenge sono descritti nella Tabella 13:

Tabella 13 CinC Challenge 2018, risultati ufficiali provvisori

AUPRC	AUROC	authors
0.439	0.902	Matthew HP et al.
0.244	0.852	Yang Liu et al.
0.228	0.822	Márton Görög et al.
0.214	0.815	Andrea Patane' et al.
0.160	0.765	Sardar Ansari
0.129	0.735	Filip Plesinger et al.
0.119	0.739	Philip de Chazal et al.
0.088	0.700	Wang Chao
0.088	0.700	Sample entry (Python)
0.082	0.652	Rohan Banerjee et al.
0.081	0.605	Ivan Lazić et al.
0.075	0.618	Alexander Yee
0.067	0.513	Sven Schellenberger et al.
0.063	0.596	Saman Parvaneh et al.
0.063	0.550	Guðni Fannar Kristjánsson et al.
0.063	0.544	Ruhallah Amandi
0.056	0.547	Haozhu Wang et al.
0.056	0.511	Fernando Andreotti et al.

0.055	0.500	Yinghua (Kelly) Shen
0.055	0.500	Hongyang Li et al.
0.051	0.488	Daniel Miller et al.
0.043	0.500	Yizhou Zhong et al.
0.037	0.315	Romeo Cabanban et al.
0.027	0.500	Shubha Majumder et al.
0.027	0.500	Jia Dongya et al.

In modelli di previsione prodotti nel corso di questi mesi di studio hanno raggiunto performance complessive nel dataset di training in un intervallo tra 0,68 e 0,76. Nella classifica riportata sopra si sarebbero classificati un po' meglio che a metà classifica (13/25). Il solo modello MATALB, testato solo su dataset ridotto chmini2 ha raggiunto un valore di AUPRC pari a circa 0,87 un valore che gli avrebbe permesso di classificarsi tra i primi 10 modelli se confermato sul dataset di test.

Tuttavia la presenza di numerosi ricercatori al di sotto del modello di sample e dei risultati prodotti dal candidato nel corso di questi mesi di studio fanno emergere una generale difficoltà per costruire modelli in grado di apprendere in maniera accettabile, sia per la complessità e la dimensione dei segnali di input, sia per la particolarità del tracciato di target delle arousal che si presenta molto sbilanciato a favore della classe 0.

Prima di approfondire i dettagli delle performance dei singoli modelli è necessario precisare come la modalità di calcolo dell'indicatore AUPRC è tale per cui viene considerata solo la quota di arousals correttamente predette. Questo perché in un dataset così fortemente sbilanciato sarebbe fin troppo facile raggiungere valori superiori al 90%, impostando un output fisso a 0.

7.2.1. *Risultati modello Logistic Regression Keras*

Il risultato di questo modello si posiziona poco sopra la metà della classifica provvisoria, anche se purtroppo è leggermente inferiore al modello di esempio. La Figura 53 mostra la distribuzione dei record per valore dell'indicatore AUPRC arrotondato ad una cifra decimale. Si noti come il modello sviluppato con le librerie Keras e Tensorflow risulti avere un 45% di casi in cui il valore dell'indicatore è stato inferiore a 0,05 a indicare una difficoltà nella classificazione.

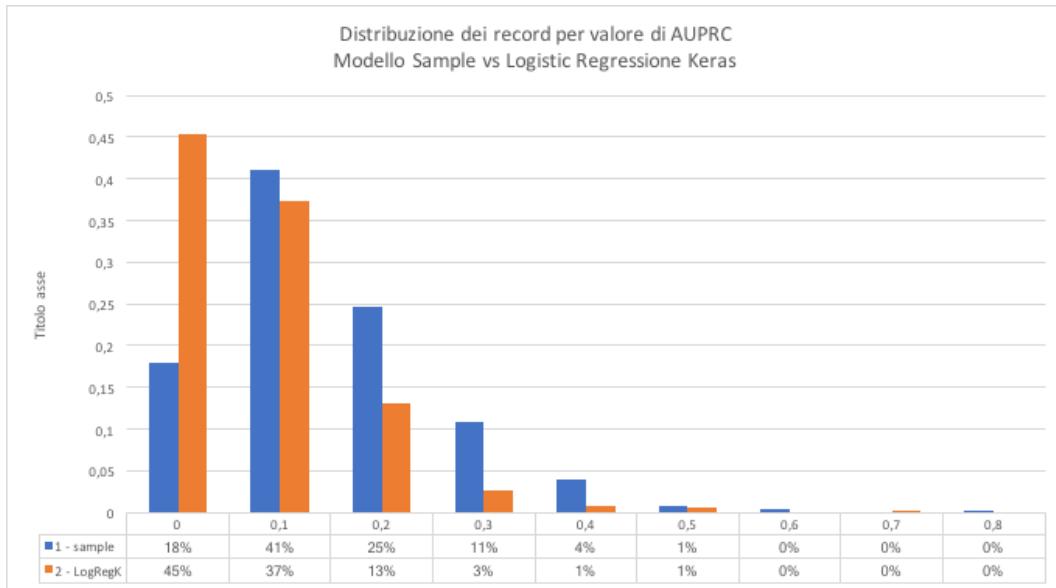
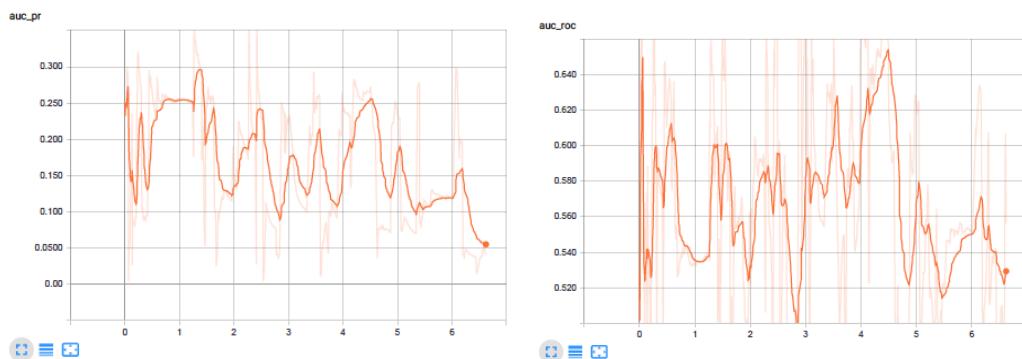


Figura 53 Modello sample vs LogRegKeras, distribuzione dei record per valore di AUPRC

Analizzando generici di AUPRC e AUROC calcolati dalle metriche TensorFlow e visualizzati tramite console grafica TensorBoard, con una percentuale di smooth del 95%, si osserva come queste due metriche assumono nel corso del training valori molto altalenanti senza mai convergere, con una AUPRC compresa tra 0,300 e 0,100 e addirittura in progressivo peggioramento, ed una AUROC che varia nell'intervallo 0,500 - 0540.

La Figura 54 mostra in ordine gli indicatori auc_pr, auc_roc, loss, acc.



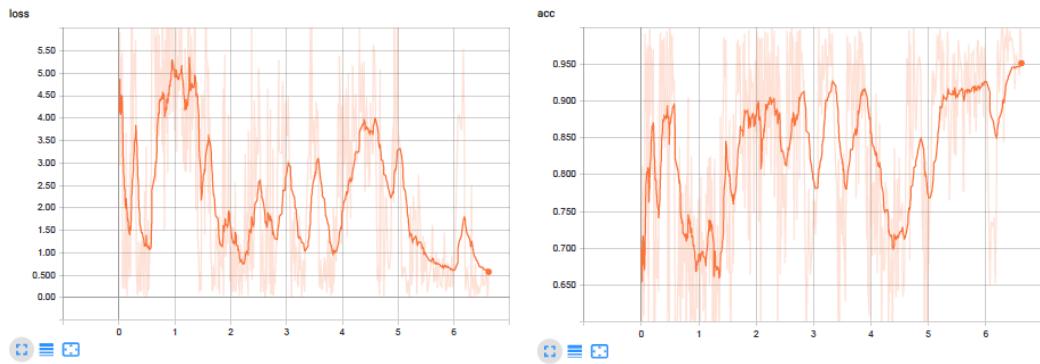


Figura 54 Performance indicate da Tensorflow per il modello LogisticRegressionKeras

7.2.1. *Risultati modello ResNet*

Il risultato di questo modello raggiunge performance simili a quello precedente. L'immagine seguente mostra la distribuzione dei record per valore dell'indicatore AUPRC arrotondato ad una cifra decimale. Si noti come il modello sviluppato con le librerie Keras e Tensorflow risulti avere un 47% di casi in cui il valore dell'indicatore è stato inferiore a 0,05 a indicare una difficoltà nella classificazione.

La performance di questo modello nel classificare gli arousal è peggiore anche del precedente Logistic Regression sviluppato in Keras e Tensorflow, in quanto la percentuale di record per i quali è stato ottenuto un valore inferiore a 0,05 è maggiore.

La Figura 55 mostra la distribuzione dei record per valore di AUPRC predetto.

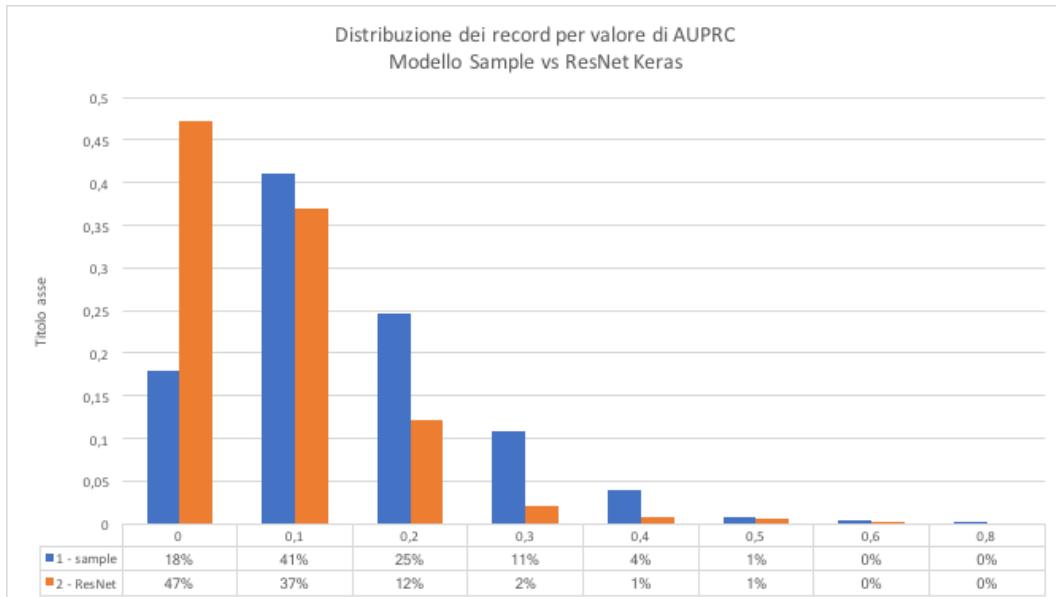


Figura 55 Modello sample vs ResNet, distribuzione dei record per valore di AUPRC

Tuttavia i valori visualizzati nella console TensorBoard mostrano un miglioramento. Il loss rimane contenuto nell'intervallo 0,4 - 0,8 e mostra una lieve discesa di almeno uno 0,1 durante l'apprendimento. L'accuracy inizia ad avere valori molto più accettabili, ricadendo in un intervallo tra 0,65 e 0,85 e mostrando anche in questo caso un trend di lieve miglioramento durante l'addestramento. I valori loss e acc sono stati riportati nella Figura 56.

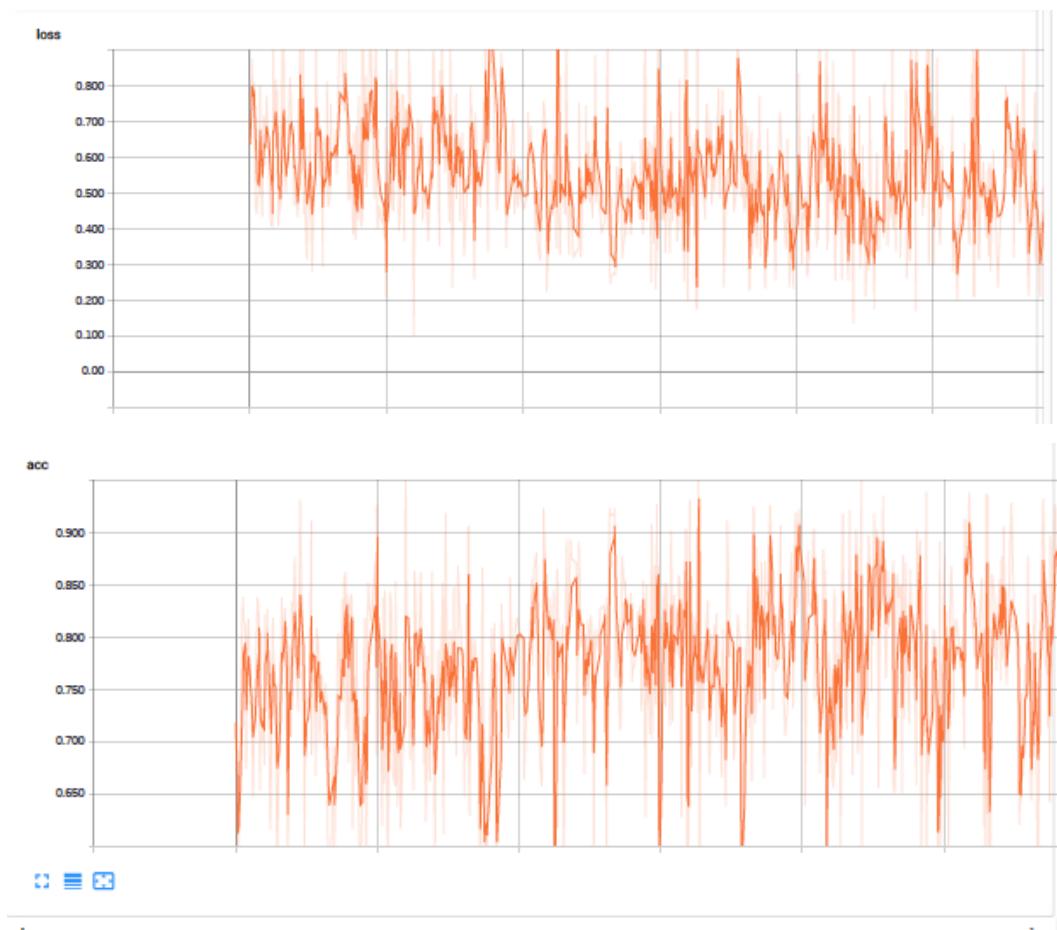


Figura 56 Performance indicate da Tensorflow per il modello ResNet

7.2.1. Risultati modello LSTM Keras e MATLAB

I valori visualizzati nella console Tensorboard mostrano un discreto miglioramento, con valori di AUPRC più stabili e compresi tra 0,935 e 0,955 e valori di AUROC compresi tra 0,94 e 0,96.

Il loss mostra valori compresi tra 0,1 e 0,4 in lieve diminuzione e l'accuratezza riporta valori compresi tra 0,86 e 0,96.

Tuttavia anche in questo caso la funzione si scoring calcola un valore di AUPRC vicino a 0,7.

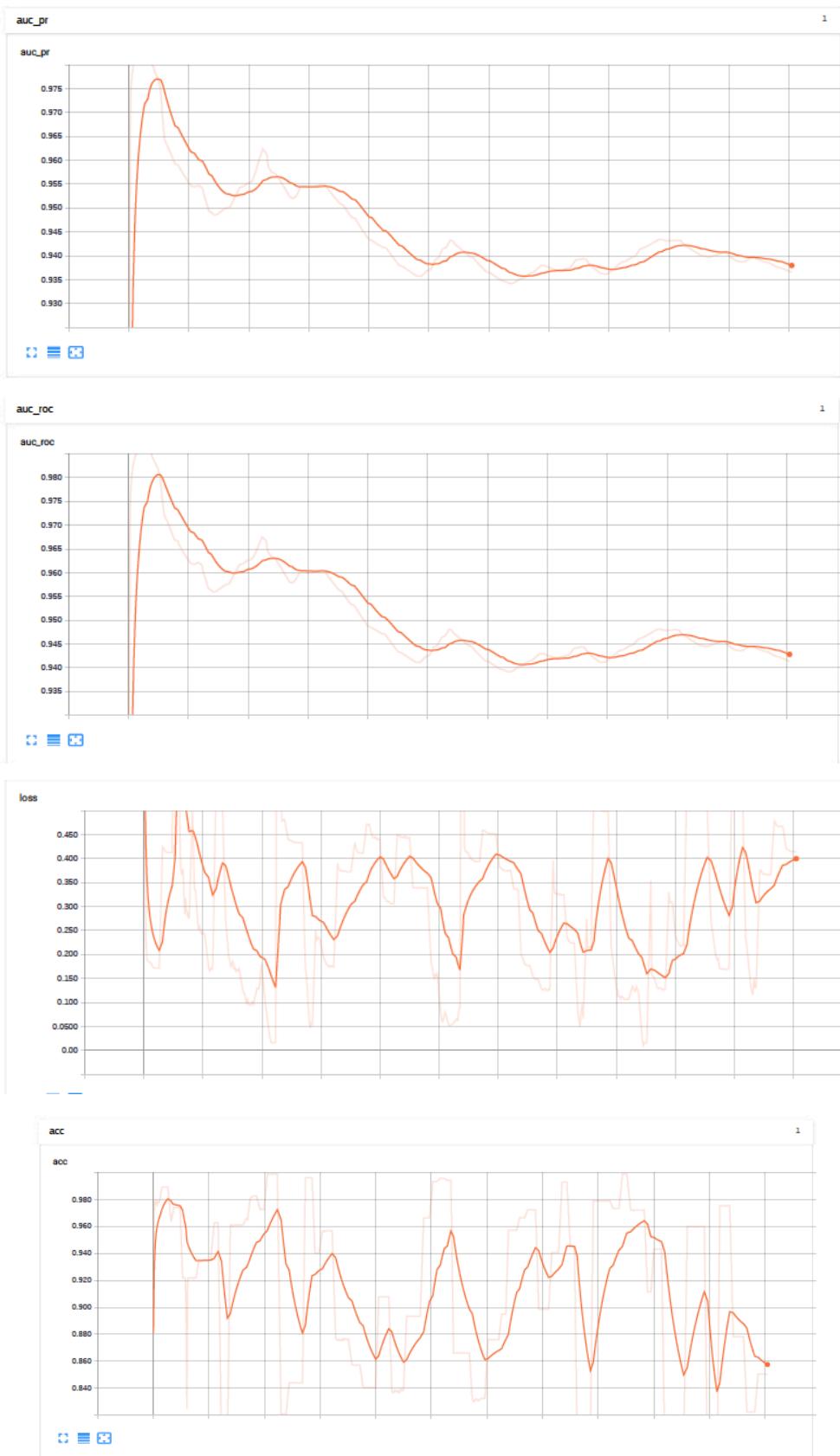


Figura 57 Performance indicate da Tensorflow per il modello LSTM

Il modello LSTM prodotto in linguaggio MATLAB e facente uso dei soli canali ECG e SaO₂, testato solo su dataset ridotto chmini2, ha ottenuto un valore di AUPRC pari a circa 0,87, un valore molto vicino al modello di esempio Python e che si sarebbe classificato alla posizione numero 10 su 25 nella lista provvisoria.

8. Conclusioni

La curva di apprendimento per acquisire un livello di conoscenze accettabili nell'area dell'informatica oggi chiamata Data Science, ed in particolare per tutto quello che ricade sotto il deep learning, è molto elevata.

Questi quasi otto mesi di studio dedicati all'approfondimento di questi argomenti sono stati una difficile ma bellissima avventura, all'inizio piena soltanto di tanta confusione sugli argomenti teorici, per poi passare ad un alternarsi di picchi di entusiasmo e scoraggiamento durante la fase di predisposizione della ambiente di sviluppo e di creazione e test dei modelli, in base a quelli che giorno dopo giorno sembrano essere discreti miglioramenti ovvero ostacoli inaspettati.

La stessa predisposizione della workstation è stato per me uno sforzo notevole sia dal punto di vista economico che tecnico per l'installazione e vari problemi di compatibilità tra versioni di sistema operativo, driver e librerie NVIDIA e poi a livello applicativo.

La scelta come argomento della tesi della Computing Challenge in Cardiology, raccontata nel capitolo 2, è stata motivata dalla volontà di accrescere le mie conoscenze nel mondo del deep learning, ed in particolare nel campo della vision che mi ha sempre appassionato e credo possa avere notevoli sviluppi in campo medico, almeno questo sembra essere un trend emergente per tante grandi corporation. Inoltre la scelta di una tematica di dominio sanitario è originata dalla mia pluriennale esperienza lavorativa nel mondo del Sanità Pubblica Italiana, prima come System Engineer nel colosso IT italiano Engineering Ingegneria Informatica, e poi come tecnico delle aziende sanitarie pubbliche ASL7 Carbonia e AO Brotzu di Cagliari.

Nel capitolo 3 ho cercato fare una sintesi di tutti gli aspetti teorici che è stato necessario approfondire, dalla classificazione teorica di base delle reti neurali fino alle ultime recentissime scoperte. Tante delle pubblicazioni, e degli articoli scientifici citati in questa tesi non erano ancora stati pubblicati nel mese di febbraio, quando ho iniziato questo percorso.

Il capitolo 4 riporta una panoramica delle principali librerie di calcolo numerico e deep learning in linguaggio Python. Non sarebbero bastate un migliaio di pagine per documentare il vastissimo set di funzionalità di basso e alto livello oggi disponibile, con ogni funzione che vede giorno dopo giorno un set crescente di parametri. Ho iniziato il mio lavoro di sperimentazione con la versione 1.5.0 di TensorFlow versione CPU, per concludere con la versione 1.9.0 versione GPU. Negli ultimi giorni di lavoro in cui mi trovo a revisionare questo documento Google annuncia l'uscita della nuova versione TensorFlow 2.0 con la promessa di tantissime novità.

Il capitolo 5 descrive alcuni dei modelli presentati per la competizione CinC challenge 2017, avente come oggetto la classificazione di segmenti di Elettrocardiogrammi per la diagnosi della patologia cardiologica denominata Fibrillazione Atriale. Si è provveduto a prendere in esame alcuni dei modelli di deep learning meglio classificati, e a riprodurre un modello basato su LSTM

Il capitolo 6 descrive i modelli sviluppati per cercare migliorare le performance provvisorie finora raggiunte nella CinC challenge 2018, avente come tema la ricerca dei fenomeni collegati agli arousals (Sindrome delle Apnee Ostruttive del Sonno).

Il capitolo 7 riporta un riepilogo dei risultati raggiunti dai modelli analizzati o prodotti per le due challenge, con una discussione tecnica sulle possibili motivazioni delle performance raggiunte. Sono stati prodotte tre ipotetici modelli di regressione logistica, rete convoluzionale ResNet e rete neurale ricorrente LSTM.

Nel caso della challenge 2017 si è riusciti ad ottenere dei risultati migliori rispetto ai primi classificati, grazie alla trasformazione del segnale tramite le due funzioni instantaneous frequency e spectral entropy.

In relazione ai lavori afferenti alla challenge 2018 si è proposto lo stesso tipo di approccio riadattando i modelli per gestire 13 canali, ottenendo risultati nella media dei lavori presentati per la conferenza. Si è cercato di utilizzare anche le stesse funzioni matematiche in un modello MATLAB, che ha mostrato

miglioramenti solo per il canale ECG ma non per gli altri dodici. Ulteriori test sono al momento in corso con ripetizioni degli stessi test con maggior numero di epoche, al fine di verificare possibili miglioramenti.

Gli obiettivi per il prossimo futuro sono quelli di provare a migliorare ulteriormente i modelli proposti e continuare a seguire lo sviluppo della CinC challenge 2018, e se sarà possibile quello di partecipare con un gruppo di ricerca alla prossima edizione di questa competizione.

Bibliografia

- [1] M. DHANDE, «AI startups are in the money: Tech companies are making massive AI investments,» 2017. [Online]. Available: <https://www.geospatialworld.net/blogs/companies-investing-in-ai-machine-learning-and-deep-learning/>.
- [2] WORLD HEALTH ORGANIZATION - Global Health Expenditure database, «The World Bank Data - Current health expenditure (% of GDP),» 2015. [Online]. Available: <https://data.worldbank.org/indicator/SH.XPD.CHEX.GD.ZS>.
- [3] CENTRO NAZIONALE PER LA PREVENZIONE DELLE MALATTIE E LA PROMOZIONE DELLA SALUTE DELL'ISTITUTO SUPERIORE DI SANITÀ, «Malattie croniche,» [Online]. Available: <http://www.epicentro.iss.it/temi/croniche/croniche.asp>.
- [4] IBM, «IBM Watson for Oncology,» [Online]. Available: <https://www.ibm.com/us-en/marketplace/ibm-watson-for-oncology>.
- [5] ALPHABET - DEEP MIND, «DeepMind Healt - Helping clinicians get patients from test to treatment, faster,» [Online]. Available: <https://deepmind.com/applied/deepmind-health/>.
- [6] MINISTERO DELLA SALUTE ITALIANO, «Sindrome da apnee ostruttive del sonno,» [Online]. Available: http://www.salute.gov.it/portale/salute/p1_5.jsp?lingua=italiano&id=106&area=Malattie_dell_apparato_respiratorio.
- [7] PHYSIONET, «You Snooze, You Win: the PhysioNet/Computing in Cardiology Challenge 2018,» [Online]. Available: <https://physionet.org/challenge/2018/>.
- [8] A. SHARMA, «Understanding Activation Functions in Deep Learning,» 30 Ottobre 2017. [Online]. Available:

<https://www.learnopencv.com/understanding-activation-functions-in-deep-learning/>.

- [9] B. Z. Q. V. L. PRAJIT RAMACHANDRAN*, «SWISH: A SELF-GATED ACTIVATION FUNCTION,» Google Brain, 16 Ottobre 2017. [Online]. Available: <https://arxiv.org/pdf/1710.05941v1.pdf>.
- [10] X. Z. S. R. J. S. KAIMING HE, «Deep Residual Learning for Image Recognition,» Microsoft Research, 10 Dicembre 2015. [Online]. Available: <https://arxiv.org/pdf/1512.03385.pdf>.
- [11] I. BOBRIAKOV, «Top 20 Python libraries for data science in 2018,» Medium, 11 Giugno 2018. [Online]. Available: <https://medium.com/activewizards-machine-learning-company/top-20-python-libraries-for-data-science-in-2018-2ae7d1db8049>.
- [12] V. CHU, «Benchmarking Tensorflow Performance and Cost Across Different GPU Options,» Medium, 20 Aprile 2017. [Online]. Available: <https://medium.com-initialized-capital/benchmarking-tensorflow-performance-and-cost-across-different-gpu-options-69bd85fe5d58>.
- [13] A. LAZORENKO, «TensorFlow performance test: CPU VS GPU,» Medium, 27 Dicembre 2017. [Online]. Available: <https://medium.com/@andriylazorenko/tensorflow-performance-test-cpu-vs-gpu-79fcd39170c>.
- [14] M. K. S. (J. Z. (ZHAOHAN XIONG (1), «Robust ECG Signal Classification for Detection of Atrial Fibrillation Using a Novel Neural Network,» (1) Auckland Bioengineering Institute, The University of Auckland, Zealand - (2) School of Medicine, The University of Auckland, Auckland, New Zealand - (3) Waikato Hospital, Hamilton, New Zealand, [Online]. Available: <http://www.cinc.org/archives/2017/pdf/066-138.pdf>.
- [15] O. C. , M. A. F. P. A. M. M. D. V. FERNANDO ANDREOTTI*, «Comparing Feature-Based Classifiers and Convolutional Neural

- Networks to Detect Arrhythmia from Short Segments of ECG,» Institute of Biomedical Engineering, University of Oxford, Oxford, United Kingdom, [Online]. Available: <http://www.cinc.org/archives/2017/pdf/360-239.pdf>.
- [16] A. Y. H. M. H. C. B. A. Y. N. PRANAV RAJPURKAR, «Cardiologist-Level Arrhythmia Detection with Convolutional Neural Networks,» 06 Luglio 2017. [Online]. Available: <https://arxiv.org/pdf/1707.01836.pdf>.
 - [17] MATHWORKS, «Classify ECG Signals Using Long Short-Term Memory Networks,» MathWorks, [Online]. Available: <https://it.mathworks.com/help/signal/examples/classify-ecg-signals-using-long-short-term-memory-networks.html>.
 - [18] J. T. L. A. X. S. PONS, «Experimenting with Musically Motivated Convolutional Neural Networks,» in *14th International Workshop on Content-Based Multimedia Indexing (CBMI)*,, 2016.
 - [19] MATHWORKS, «tftmoment - Conditional temporal moment of the time-frequency distribution of a signal,» MathWorks, [Online]. Available: <https://it.mathworks.com/help/predmait/ref/tftmoment.html>.
 - [20] MATHWORKS, «instfreq - Estimate instantaneous frequency,» [Online]. Available: https://it.mathworks.com/help/signal/ref/instfreq.html?searchHighlight=instfreq&s_tid=doc_srchtitle.
 - [21] MATHWORKS, «pentropy - Spectral entropy of signal,» MathWorks, [Online]. Available: https://it.mathworks.com/help/signal/ref/pentropy.html?s_tid=doc_ta.
 - [22] GARI D CLIFFORD ET AL., AF Classification from a Short Single Lead ECG Recording: the PhysioNet/Computing in Cardiology Challenge 2017, 2017

- [23] TOMÁS TEIJEIRO ET AL., Arrhythmia Classification from the Abductive Interpretation of Short Single-Lead ECG Records, 2017
- [24] SHREYASI DATTA ET AL, Identifying Normal, AF and other Abnormal ECG Rhythms using a Cascaded Binary Classifier, 2017
- [25] SHENDA HONG ET AL., ENCASE: an ENsemble CLASsifiEr for ECG Classification Using Expert Features and Deep Neural Networks, 2017
- [26] NITISH SRIVASTAVA, UNIVERSTY OF TORONTO, Dropout: A Simple Way to Prevent Neural Networks from Overfitting, 2014
- [27] PRAJIT RAMACHANDRAN, GOOGLE BRAIN, Dropout: SWISH: a self-gated activation function, 2017
- [28] KAIMING HE AT AL., Deep Residual Learning for Image Recognition, 2015
- [29] ZHAOHAN XIONG ET AL., A Machine Learning Aided Systematic Review and Meta-Analysis of the Relative Risk of Atrial Fibrillation in Patients With Diabetes Mellitus, 2018
<https://www.frontiersin.org/articles/10.3389/fphys.2018.00835/full>
- [30] MARTIN GORNER, GOOGLE, Tensorflow and Deep Learing without a PhD, presentazione per Devoxx Conference 2016,
<https://docs.google.com/presentation/d/1TVixw6ItiZ8igjp6U17tcgoFrLSaHWQmMOwjlgQY9co/pub?slide=id.p>
- [31] CHRISTOPHER OLAH, GOOGLE BRAIN, Understanding LSTM Networks, <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- [32] ROHITH GANDHI, Introduction to Sequence Models — RNN, Bidirectional RNN, LSTM, GRU, 2017,
<https://towardsdatascience.com/introduction-to-sequence-models-rnn-bidirectional-rnn-lstm-gru-73927ec9df15>
- [33] JASON BROWNLEE, How to Develop a Bidirectional LSTM For Sequence Classification in Python with Keras, 2017,

- <https://machinelearningmastery.com/develop-bidirectional-lstm-sequence-classification-python-keras/>
- [34] HONGMING CHEN, THE RISE OF DEEP LEARNING IN DRUG DISCOVERY, 2018,
<https://www.sciencedirect.com/science/article/pii/S1359644617303598>
- [35] TUSHAR GUPTA, Deep Learing, 2016,
<https://towardsdatascience.com/deep-learning-d5fe55326e57>
- [36] TUSHAR GUPTA, Deep Learing: Feedforward Neural Network, 2017,
<https://towardsdatascience.com/deep-learning-feedforward-neural-network-26a6705dbdc7>
- [37] DOUKKALI FIRDAOUS, Batch normalization in Neural Networks,
<https://towardsdatascience.com/batch-normalization-in-neural-networks-1ac91516821c>
- [38] ANUJA NAGPAL, L1 and L2 Regularization Methods, 2017
<https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c>
- [39] ANISH SINGH WALIA, Types of Optimization Algorithms used in Neural Networks and Ways to Optimize Gradient Descent, 2017
<https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f>
- [40] VADIM SMOLYAKOV, Neural Network Optimization Algorithms, 2018
<https://towardsdatascience.com/neural-network-optimization-algorithms-1a44c282f61d>
- [41] DAPHNE CORNELISSE, An intuitive guide to Convolutional Neural Networks, 2018, <https://medium.freecodecamp.org/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050>
- [42] JOHN OLAFENWA, Understanding residual networks, 2018,
<https://towardsdatascience.com/understanding-residual-networks-9add4b664b03>

- [43] ADITYA SHARMA, Understanding Activation Functions in Deep Learning, <https://www.learnopencv.com/understanding-activation-functions-in-deep-learning/>
- [44] CORNELL UNIVERSITY, Department of Computer Science, CS1114: Introduction to Computing using Matlab and Robotics, Section 6: https://www.cs.cornell.edu/courses/cs1114/2013sp/sections/S06_convolution.pdf
- [45] ANDREW NG, Coursera and Stanford University, Convolutional Neural Networks - Pooling Layers.
<https://www.coursera.org/lecture/convolutional-neural-networks/pooling-layers-hELHk>
- [46] ANDREW NG ET AL., Coursera and Stanford University, Sequence Models
- [47] PAOLO MEDICI, Università di Parma, Elementi di analisi per Visione Artificiale - Generalizzare l'addestramento, 2017
- [48] ROTA BULÒ, Appunti di reti neurali, 2006
<http://www.dsi.unive.it/~srotabul/files/AppuntiRetiNeurali.pdf>