



UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Triennale in Informatica

TESI DI LAUREA

# ExoHunter: una rete neurale convoluzionale per la classificazione di curve di luce di esopianeti

RELATORE

**Prof. Fabio Palomba**

Università degli studi di Salerno

CANDIDATO

**Giorgio Angelo Esposito**

Matricola: 0512107389

*"Ogni cosa ha le sue meraviglie, anche l'oscurità e il silenzio, e io imparo, in qualunque condizione mi possa trovare, ad esserne contenta"* - Helen Keller

## Sommario

L'astronomia è uno dei campi scientifici più importanti di sempre. Ha vissuto nel corso dei millenni continue evoluzioni che hanno portato a enormi scoperte su tutto ciò che ci circonda. Con l'avanzare della tecnologia, soprattutto in tempi recenti, la mole di dati che gli scienziati hanno a disposizione è sempre maggiore e non sempre riescono ad esaminarli nella loro completezza. Per questo motivo, è stato necessario utilizzare i progressi fatti in campi esterni a quello astronomico per poter velocizzare ed automatizzare i processi di analisi, pulizia, costruzione e simili che sono effettuati su questi dati. Le principali innovazioni esterne introdotte vengono dal campo dell'informatica e hanno a che fare principalmente con l'Intelligenza Artificiale, più precisamente con i sottocampi del Machine Learning e Deep Learning. Una degli studi astronomici in cui si utilizzano queste tecniche è la ricerca di esopianeti, pianeti al di fuori del nostro Sistema Solare. Questa tesi si propone di presentare, dopo aver analizzato il *background* di ricerche proposte su questo tema, una nuovo modello, basato sulle più recenti tecniche di Intelligenza Artificiale, capace di classificare correttamente se una data curva di luce rappresenta un esopianeta oppure si tratta di un falso positivo. Utilizzando i dati presenti nel *Nasa Exoplanet Archive*, si è creato un dataset per il problema in questione, che si compone di curve di luce (diminuzioni periodiche della luminosità della stella attorno al quale orbita l'esopianeta) e di falsi positivi, ed è stato utilizzato per addestrare una rete neurale convoluzionale, a cui è stato dato il nome di ExoHunter. Questa rete ottiene un *accuracy* media dell'89%, riuscendo quindi a distinguere correttamente le curve di luce di esopianeti da falsi positivi.

<b>Indice</b>	<b>ii</b>
<b>Elenco delle figure</b>	<b>iv</b>
<b>Elenco delle tabelle</b>	<b>vii</b>
<b>1 Introduzione</b>	<b>1</b>
1.1 Introduzione . . . . .	1
1.2 L'IA nell'astronomia . . . . .	3
1.3 Struttura della tesi . . . . .	3
<b>2 Esopianeti</b>	<b>4</b>
2.1 Definizione e classificazione . . . . .	4
2.2 Metodi di scoperta . . . . .	5
2.3 Telescopi utilizzati . . . . .	8
<b>3 Stato dell'arte</b>	<b>9</b>
3.1 Reti neurali artificiali . . . . .	10
3.2 Reti neurali convoluzionali . . . . .	11
3.3 Astronet . . . . .	13
3.4 Astronet-K2 . . . . .	19
3.5 Exonet . . . . .	25
3.6 Astronet-Vetting . . . . .	28
3.7 Genesis . . . . .	33

3.8 Identificazione di esopianeti utilizzando dati reali e artificiali . . . . .	39
<b>4 ExoHunter</b>	<b>45</b>
4.1 Formulazione del problema . . . . .	45
4.2 Creazione del dataset . . . . .	46
4.3 ExoHunter . . . . .	50
4.4 Risultati ottenuti . . . . .	54
<b>5 Conclusioni</b>	<b>58</b>
<b>Ringraziamenti</b>	<b>60</b>
<b>Bibliografia</b>	<b>61</b>

---

## Elenco delle figure

---

2.1	Metodo dei transiti, fonte: NASA . . . . .	6
2.2	Numero di esopianeti scoperti con i diversi metodi negli anni (dati aggiornati al 1° settembre 2022) Fonte: NASA. . . . .	7
3.1	L'architettura di una rete neurale artificiale completamente connessa. Nell'architettura sono presenti lo strato di input, due strati nascosti e lo strato di output. Fonte: " <i>Identifying exoplanets with deep learning: A five-planet resonant chain around kepler-80 and an eighth planet around kepler-90</i> ". . . . .	11
3.2	Parte dell'architettura di una rete neurale convoluzionale. Possiamo notare: (a) l'organizzazione dei neuroni nelle tre dimensioni, (b) le operazioni di convoluzione e di <i>pooling</i> . Fonte: " <i>Identifying exoplanets with deep learning: A five-planet resonant chain around kepler-80 and an eighth planet around kepler-90</i> ". . . . .	12
3.3	Esempi di <i>global view</i> e <i>local view</i> . Fonte: " <i>Identifying exoplanets with deep learning: A five-planet resonant chain around kepler-80 and an eighth planet around kepler-90</i> " . . . . .	14
3.4	Le curve del <i>Precision vs Recall</i> calcolate per <i>Astronet</i> . Fonte: " <i>Identifying exoplanets with deep learning: A five-planet resonant chain around kepler-80 and an eighth planet around kepler-90</i> " . . . . .	17

3.5 L'architettura di Astronet, fonte: " <i>Identifying exoplanets with deep learning: A five-planet resonant chain around kepler-80 and an eighth planet around kepler-90</i> ". Gli strati convoluzionali sono indicati come conv<kernel-size>-<number of feature map>, gli strati di max pooling sono indicati come maxpool<window lenght>-<stride lenght> e gli strati completamente connessi sono indicati come FC-<number of units> . . . . .	18
3.6 L'architettura di <i>Astronet-K2</i> , fonte: " <i>Identifying exoplanets with deep learning. ii. two new super-earths uncovered by a neural network in K2 data</i> ". Gli strati convoluzionali sono indicati come conv<kernel size>-<number of feature map>, gli strati di max pooling sono indicati come max<window lenght>-<stride lenght> e gli strati completamente connessi sono indicati come FC-<number of units>. . . . .	22
3.7 Le curve del <i>Precision vs Recall</i> calcolate per <i>Astronet-K2</i> . Nel grafico sono presenti le curve calcolate per ogni sottoinsieme del <i>test set</i> diviso in base al rapporto S/N. Fonte: " <i>Identifying exoplanets with deep learning. ii. two new super-earths uncovered by a neural network in K2 data</i> ". . . . .	24
3.8 L'architettura di <i>Exonet</i> (sinistra) e <i>Exonet-XS</i> (destra), fonte: " <i>Scientific domain knowledge improves exoplanet transit classification with deep learning</i> ". Gli strati convoluzionali sono indicati coem CONV-<kernel size>-<number of feature map>, gli strati di max pooling sono indicati come MAXPOOL-<window lenght>-<stride lenght> e gli strati completamente connessi sono indicati come FC-<number of units>. . . . .	27
3.9 Le curve del <i>Precision vs Recall</i> calcolate sui vari modelli: <i>Exonet-Centroid</i> è il modello con la sola aggiunta delle curve dei centroidi, <i>Exonet-Stellar</i> è il modello con la sola aggiunta dei parametri stellari, <i>Exonet-Argumented</i> è il modello con la sola aggiunta della <i>data-argumentation</i> e <i>Exonet</i> è il modello con l'aggiunta di tutte e tre le caratteristiche. Fonte: " <i>Scientific domain knowledge improves exoplanet transit classification with deep learning</i> ". . . . .	29
3.10 Le curve del <i>Precision vs Recall</i> calcolate sui vari modelli: <i>Vetting-plain</i> è il modello Astronet. Fonte: " <i>Identifying exoplanets with deep learning. III. Automated triage and vetting of TESS candidates</i> ". . . . .	34

3.11 L'architettura di <i>Astronet-Vetting</i> , fonte: "Identifying exoplanets with deep learning. III. Automated triage and vetting of TESS candidates". Gli strati convoluzionali sono indicati come conv<kernel size>-<number of feature map>, gli strati di max pooling sono indicati come maxpool<window lenght>-<stride lenght> e gli strati completamente connessi sono indicati come FC-<number of units>.. .	35
3.12 L'architettura di <i>Genesis</i> , fonte: "A one-armed CNN for exoplanet detection from light curves". CONV-50-64 indica uno strato convoluzionale di dimensione 50 con un numero di filtri di 64. MAXPOOL-32-32 indica uno strato di max pooling di lunghezza 32 e stride di 32. AVGPOOL-64 indica l'average pooling su 49.216 input. DROPOUT-0.25 indica uno strato di dropout con probabilità del 25%. FC-256 indica uno strato completamente connesso con 256 neuroni. .	37
3.13 Istogrammi relativi alle ensambled accuracies ottenute da Genesis. Fonte immagini: "A one-armed CNN for exoplanet detection from light curves" . . . . .	38
3.14 Grafico 3D dei valori di $F_1$ rispetto a $\lambda$ e T. Fonte: "Deep learning exoplanets detection by combining real and synthetic data" . . . . .	42
3.15 Curve ROC per i diversi valori di $\lambda$ . Fonte = "Deep learning exoplanets detection by combining real and synthetic data" . . . . .	44
4.1 La tabella relativa ai KOI del <i>Nasa Exoplanet Archive</i> . . . . .	47
4.2 Curva di luce di <i>Kepler-53 d</i> . . . . .	49
4.3 Esempio di curva di luce di falso positivo . . . . .	50
4.4 Confronto tra la curva di luce originale (blu) e la curva di luce artificiale (giallo)	51
4.5 L'architettura di <i>ExoHunter</i> . Gli strati convoluzionali sono indicati come Conv1D (numero di filtri, dimensione del kernel), gli strati di <i>Maxpooling</i> sono indicati come MaxPool (dimensioni della pool, ampiezza del passo), gli strati completamente connessi come FC-numero di neuroni, lo strato di <i>dropout</i> come Dropout (tasso di <i>dropout</i> ). . . . .	53
4.6 Grafici dell'andamento della funzione di perdita e dell' <i>accuracy</i> rispetto alle epoche per <i>ExoHunter_10</i> . . . . .	55
4.7 Matrice di confusione che riporta la media dei risultati di tutte le matrici di confusione calcolate dai dieci modelli sul test set. . . . .	56
4.8 La curva ROC-AUC calcolata sul test set. . . . .	57

---

## Elenco delle tabelle

---

3.1	Tabella relativa ai valori di <i>accuracy</i> sul <i>test set</i>	16
3.2	Tabella relativa ai valori di AUC sul <i>test set</i>	17
3.3	Confronto dei risultati delle due reti sul <i>test set</i>	28
3.4	I risultati ottenuti sul <i>test set</i> dalle diverse tipologie di reti.	33
3.5	Paragone dei risultati tra Genesis e Astronet	39
3.6	I risultati dell'algoritmo genetico con diversi parametri.	43

# CAPITOLO 1

---

## Introduzione

---

### 1.1 Introduzione

Scutare l'Universo è l'attività umana più antica. Anche quando l'uomo non aveva i mezzi per poterli osservare più da vicino ha studiato i movimenti dei corpi celesti, si è interessato della posizione delle stelle e del ripetersi degli eventi astronomici, come ad esempio le eclissi, arrivando anche a creare i primitivi strumenti, come l'astrolabio (strumento utilizzato per misurare l'altezza delle stelle rispetto all'orizzonte) e i cataloghi; ha fatto supposizioni, ha messo a frutto l'immaginazione, trovando le prime risposte ai suoi interrogativi, attraverso la creazione del mito, ma il continuo osservare ha trovato sempre più conferma nei calcoli matematico-scientifici, tant'è che già nel 40 a.C. si delineano due grandi teorie: quella tolemaica e quella aristotelica. Essi fissano un primo punto fondamentale "la teoria geocentrica". La visione cristiana - di discendenza aristotelica - domina sulle teorie scientifiche durante il Medioevo, generando una battuta d'arresto nelle ricerche scientifiche, viceversa il mondo islamico conobbe la sua massima espansione, dove vennero costruiti i primi osservatori astronomici, istituzioni scientifiche vere e proprie. La vera "Rivoluzione astronomica" arriva solo nella prima età moderna che fissa un importante risultato: il passaggio dal 'modello geocentrico' al 'modello eliocentrico'. Si sostituiscono definitivamente le precedenti teorie tolemaiche/aristoteliche con quelle di Niccolò Copernico e Galileo Galilei. Lo scienziato polacco espone la sua teoria eliocentrica nel libro *De revolutionibus orbium coelestium* e colui che la verificò fu Galilei – puntando semplicemente il cannocchiale verso l'infinito cosmico –

e nel *Sidereus nuncius* lo scienziato pisano conferma tutte le teorie di Copernico<sup>1</sup>.

Intanto, Galileo Galilei aveva introdotto l'uso del telescopio nell'astronomia e grazie ad esso scoprì i satelliti di Giove, le fasi di Venere, le macchie solari e quelli che saranno poi identificati - da parte del matematico olandese Huygens, scopritore anche del satellite più grande del pianeta, Titano - come gli anelli di Saturno. Lo strumento permise a Galilei anche di poter osservare la superficie lunare, scoprendola carettizzata da valli, monti e crateri. Susseguirono poi le leggi di Keplero relative al moto dei pianeti attorno al Sole e Newton, sulla base di queste scoperte, arriverà a formulare nel 1687, le legge di gravitazione universale.

Nel XVIII secolo, Messier compila il primo catalogo relativo alle comete, Herschel scopre Urano e due dei suoi satelliti, Titania e Oberon; è identificato il primo asteroide, scoperti i satelliti di Marte ed è stata ipotizzata l'esistenza di un ottavo pianeta nel Sistema Solare mediante soli calcoli matematici da parte dei matematici Adams e Le Verrier, che porterà alla scoperta, nel 1846, di Nettuno.

Nel XIX secolo viene scoperto Plutone, l'astronomo statunitense Hubble dimostra l'esistenza delle galassie e formula la legge che porta il suo nome, che lega in modo lineare il colore emesso dalle galassie alla loro distanza: maggiore è la distanza della galassia, maggiore è il suo spostamento verso il colore rosso. Questa legge permise di formulare l'idea di espansione dell'Universo e successivamente alla formulazione della teoria del Big Bang, che troverà conferma nel 1965, grazie alla scoperta della radiazione cosmica di fondo da parte di Penzias e Wilson. Inoltre, viene scoperto il centro della Via Lattea da parte di Oort. Il percorso di scoperte astronomiche subisce un'interruzione durante la Seconda Guerra Mondiale, ma sep-pur per scopi bellici, grazie agli studi effettuati dai tedeschi sui lanci missilistici nasce quella che prende il nome di 'Astronautica', che permetterà il lancio dei primi satelliti artificiali attraverso i quali saranno possibili osservazioni dirette degli oggetti astronomici. Vengono lanciate le prime sonde verso i pianeti del Sistema Solare: le sonde Luna 9 e Surveyor 1, lanciate rispettivamente dell'Unione Sovietica e Stati Uniti, sono le prime ad atterrare sulla superficie lunare, Venera 7, sempre sovietica, è la prima ad atterrare su Venere mentre la sonda statunitense Mariner 10 rimane l'unica ad aver sorvolato Mercurio. Le sonde americane Viking 1 e Viking 2 trasmettono le prime immagini della superficie marziana. Le sonde Pioneer 10 e Pioneer 11 furono le prime a sorvolare Giove e Saturno, mentre con la sonda Voyager 2 si ebbe il primo sorvolo di Urano e Nettuno. Altro passo avanti è stato quello dei telescopi spaziali: il più famoso di tutti è Hubble, lanciato in orbita terrestre nel 1990

<sup>1</sup>Solo nel 1687, la teoria eliocentrica di Copernico trova conferma e riconoscimento con la pubblicazione del *Philosophiae naturalis principia mathematica* di Newton (1642-1727).

ed attualmente operativo. Hubble ha permesso di vedere più in là di qualsiasi telescopio terrestre. Il suo successore, il telescopio spaziale Webb, è stato lanciato con successo il 25 dicembre 2021, le cui prime immagini sono state diffuse il 12 luglio 2022. È attualmente il più grande telescopio spaziale mai costruito.

## 1.2 L'IA nell'astronomia

Con il progresso tecnologico, l'astronomia ha subito un' impennata di dati prodotti, che superano le dimensioni degli Exabytes [1]. Questo ha fatto sì che i tradizionali metodi di analisi dei dati utilizzati in questo campo diventassero obsoleti [2] e che nuovi metodi dovessero essere proposti per poter gestire l'enorme quantità di dati, metodi, che devono essere soprattutto automatizzati [3].

Mentre metodi automatizzati di *Data Mining* sono ormai ampiamente utilizzati in tutti i sottocampi dell'astronomia, l'attenzione si è spostata verso l'Intelligenza Artificiale (IA) e i suoi derivati, come il *Machine Learning* e il *Deep Learning*.

Fluke and Jacobs [1] hanno analizzato i progressi fatti nell'utilizzo dell' IA nell'astronomia. In particolare, gli studiosi hanno individuato le tecniche più utilizzate di IA in base al tipo di dato prodotto. L' IA è impiegata nei differenti campi dell'astronomia, anche se non in maniera omogenea, in taluni campi l'uso dell'IA è solo agli inizi, in altri progredisce mentre nella ricerca e classificazione di pianeti extrasolari che l'IA viene utilizzata con successo.

## 1.3 Struttura della tesi

La tesi si divide in cinque capitoli.

Il Capitolo 1 è dedicato ad un excursus storico sugli studi e le scoperte astronomiche, fino ad arrivare all'utilizzo dell'IA e dei metodi applicativi che trovano sempre più conferme per la gestione dei dati.

Nel Capitolo 2 sono definiti gli esopianeti: come si identificano e dei telescopi spaziali che hanno permesso e permetteranno di scoprire la loro presenza nell'Universo.

Nel Capitolo 3 è esposto lo stato dell'arte; sono quindi descritte alcune ricerche fondamentali che hanno aperto il campo all'identificazione di esopianeti usando il Deep Learning.

Nel Capitolo 4 viene descritto come è stato prodotto il dataset utilizzato per risolvere il problema, come è stata costruita la rete neurale convoluzionale e i risultati ottenuti.

Nel Capitolo 5 i risultati ottenuti vengono discussi e si indicano eventuali migliorie.

# CAPITOLO 2

---

## Esopianeti

---

“Da qualche parte, qualcosa di incredibile attende di essere conosciuto.” - Carl Sagan

### 2.1 Definizione e classificazione

Un esopianeta, o pianeta extrasolare, è un pianeta che orbita attorno a una stella diversa dal Sole. Il primo esopianeta, *51 Pegasi b*, venne scoperto nel 1995. Da allora, in oltre 25 anni di ricerca, ne sono stati scoperti 5.084 esopianeti<sup>1</sup>, con oltre 9.000 candidati ancora in attesa di classificazione.

I pianeti variano in dimensioni e composizione, pertanto possono essere classificati nei seguenti modi:

- **Giganti gassosi:** sono pianeti dalle dimensioni maggiori di Giove, composti principalmente da elio e/o idrogeno. Come Giove, non hanno una superficie solida, che invece è composta da gas e da un nucleo solido.

Quando un pianeta di questo tipo orbita molto vicino alla sua stella (a distanze più ravvicinate rispetto a quella di Mercurio) il pianeta prende il nome di *Giove caldo*. Questo nome è dovuto alle elevate temperature raggiunte sul pianeta.

- **Nettuniani:** sono pianeti dalle dimensioni simili a quelle di Nettuno o Urano. Hanno atmosfere composte da idrogeno e un nucleo composto da metalli leggeri.

---

<sup>1</sup>Dato aggiornato all' 11 settembre 2022.

- **Super Terre:** sono pianeti dalle dimensioni superiori alla Terra, ma inferiori a quelle di Nettuno o Urano. La loro composizione può essere gassosa, rocciosa o entrambe. Il nome serve solo a raggruppare esopianeti di queste dimensioni e non indicano la loro abitabilità.
- **Terrestri:** sono pianeti le cui dimensioni vanno da metà di quella della Terra al doppio del suo raggio. Sono composti da materiale roccioso e hanno una superficie solida o liquida.

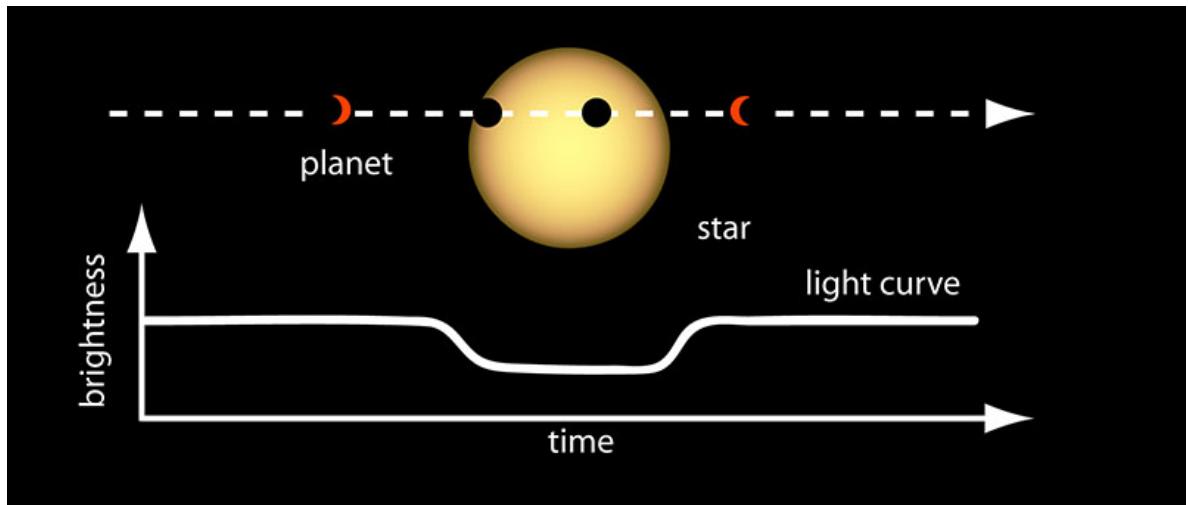
Molti di questi pianeti sono stati trovati nella zona abitabile, ma non è stato ancora possibile stabilire se siano presenti oceani o atmosfere.

## 2.2 Metodi di scoperta

Esistono diversi modi per scoprire un esopianeta:

- **Metodo delle velocità radiali (spettroscopia Doppler):** quando un pianeta orbita intorno a una stella, sembra che la stella sia "immobile", ma in realtà il pianeta esercita una piccola forza gravitazionale che fa sì che la stella "oscilli" leggermente. Più grande è il pianeta, maggiore sarà la forza gravitazionale e, di conseguenza, maggiore l'oscillazione. Queste oscillazioni fanno sì che la luce della stella osservata attraverso la spettroscopia, passi dall'avere onde più distese e di colore rosso, quando la stella si 'allontana' dal punto di vista dell'osservatore, all'avere onde più raggruppate e di colore blu quando la stella si 'avvicina' al punto di vista dell'osservatore (per l'effetto Doppler).  
Questo è stato uno dei primi metodi utilizzati e ancora oggi continua ad avere molto successo.
- **Metodo astrometrico:** le oscillazioni causate dalla presenza di un pianeta attorno a una stella non sono osservabili solamente tramite la spettroscopia, ma anche dai cambiamenti minuscoli della posizione della stella nel cielo. Con questo metodo si effettuano fotografie della stella e delle stelle sue vicine e in ogni foto si effettuano confronti tra le distanze delle stelle rispetto a quella in cui si cercano esopianeti. Se la stella si è mossa, allora il movimento viene analizzato per cercare un pianeta che orbita attorno a tale stella.  
Questo metodo richiede strumenti ottici molto precisi ed è molto difficile da applicare a causa della atmosfera terrestre che distorce la luce ricevuta.

- **Metodo dei transiti:** quando un pianeta passa davanti alla sua stella, quest'ultimo blocca una piccola parte della luce della stella, facendo sì che la luce osservata sia minore per un certo lasso di tempo, da quando entra nel disco della stella fino a quando esce. La Figura 2.1 illustra quanto accade.
- Questo metodo è uno dei più utilizzati ed è quello che ha portato al maggior numero di esopianeti scoperti.



**Figura 2.1:** Metodo dei transiti, fonte: NASA

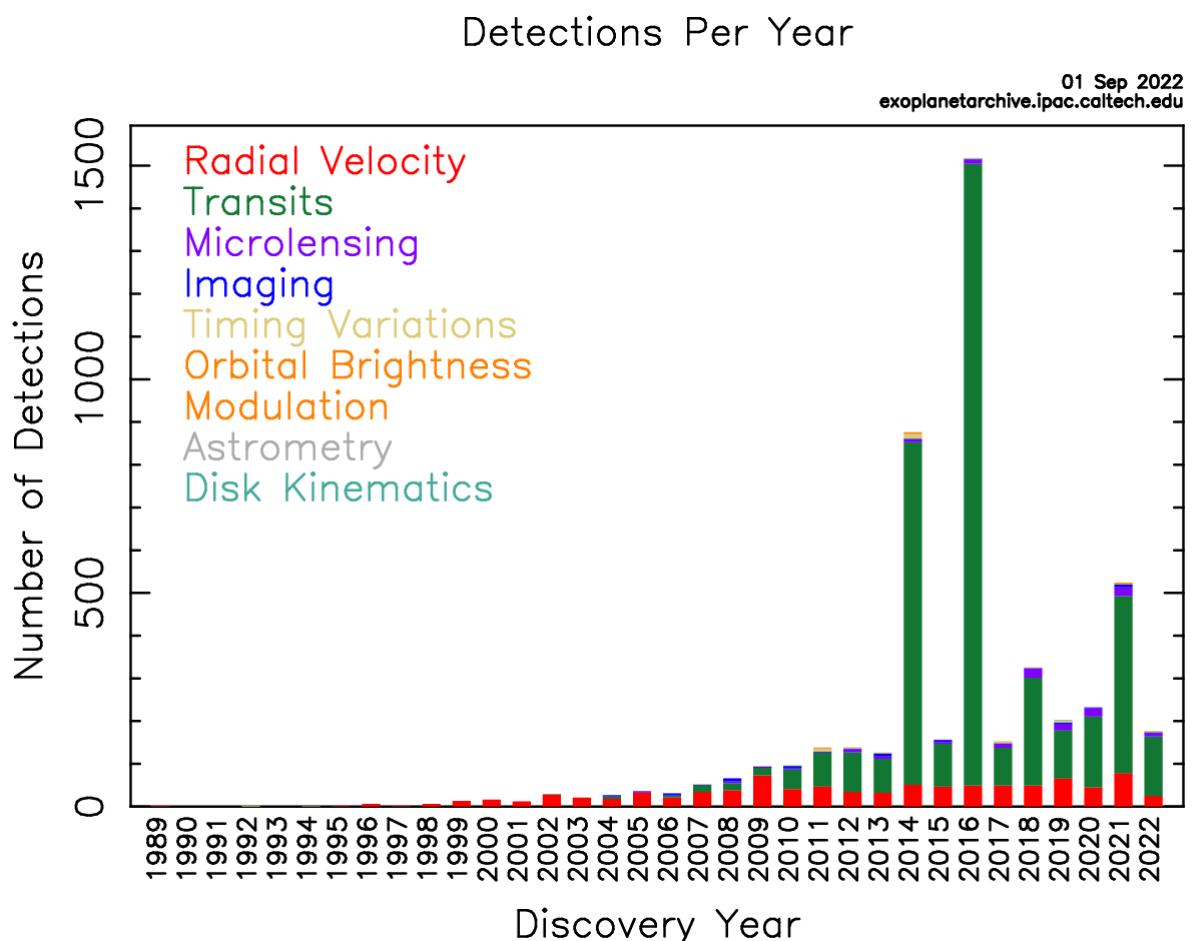
- **Metodo della microlente gravitazionale:** questo metodo si basa sulla teoria della relatività generale di Einstein, secondo cui la massa di un oggetto deforma lo spazio. Questo fa sì che anche la luce venga distorta quando passa vicino ad un oggetto, a causa della gravità dello stesso. Pertanto, quando un oggetto passa tra l'osservatore e la stella più lontana, la luce di quest'ultima appare più luminosa. L'oggetto che causa questo fenomeno è chiamato microlente gravitazionale.

Quando l'oggetto passa davanti alla stella più lontana, la luce osservata di quest'ultima aumenta progressivamente fino a raggiungere un punto massimo per poi diminuire progressivamente. Se tale oggetto avesse un pianeta che gli orbita attorno, allora il momentaneo aumento di luminosità non sarebbe più regolare, ma ci sarebbe un piccolo picco durante l'aumento o la diminuzione.

- **Osservazione diretta:** questo metodo si basa sull'osservazione diretta dell'esopianeta. La difficoltà principale è che la stella attorno cui orbita è molto più luminosa del pianeta stesso. Per questo motivo, sono stati sviluppati strumenti che permettono di bloccare la luce stellare, principalmente ce ne sono due:

1. Il primo modo è la coronografia: si utilizza uno strumento chiamato coronografo che copre il disco solare, producendo un'eclissi artificiale nello strumento.
2. Il secondo modo consiste nell'utilizzare un dispositivo che blocca la luce prima ancora che entri nel telescopio.  
E' un dispositivo separato dal telescopio stesso, che si posizionerebbe alla giusta distanza e alla giusta angolazione.

Ad oggi, il metodo dei transiti è quello che ha portato alla scoperta del maggior numero di esopianeti, seguito dal metodo della velocità radiale, come si può vedere dalla Figura 2.2



**Figura 2.2:** Numero di esopianeti scoperti con i diversi metodi negli anni (dati aggiornati al 1° settembre 2022) Fonte: NASA.

Per un approfondimento matematico sui metodi di scoperta, si consiglia la lettura del paper: "Exoplanet detection methods" di Wright and Gaudi [4].

## 2.3 Telescopi utilizzati

Vengono qui riportati alcuni dei telescopi spaziali che sono stati sviluppati con lo scopo di scoprire esopianeti.

- **CoRoT (Convection, Rotation and planetary Transit):** il telescopio CoRoT, sviluppato dall’Agenzia Spaziale Francese (CNES), in collaborazione con l’Agenzia Spaziale Europea (ESA), aveva come obiettivo secondario quello di individuare esopianeti col metodo dei transiti. È stato il primo telescopio spaziale orbitante progettato per la ricerca di esopianeti.

Lanciato nel 2006, ha scandagliato la regione della costellazione del Serpente e dell’Unicorno, identificando 34 esopianeti in sei anni di attività; il telescopio è stato disattivato nel 2012, l’influenza di radiazioni ne hanno causato un malfunzionamento che impediva la trasmissione dei dati raccolti.

- **Kepler:** il telescopio Kepler dovette osservare una regione della costellazione del Cigno di circa 190.000 stelle, usando il metodo dei transiti. Lanciato nel 2009, con un ciclo vitale previsto di 3 anni e mezzo, subì nel 2012 un guasto al meccanismo di stabilizzazione, il che fece pensare a una fine anticipata della missione. Ma si riuscì a riparare parzialmente il guasto, che portò a un’estensione della missione denominata ‘K2, Second Light’. Questa missione proseguì fino al 2018, anno ufficiale dello spegnimento. Nei suoi 9 anni e mezzo di vita ha scoperto oltre 2600 esopianeti.
- **Tess (Transiting Exoplanet Survey Satellite):** questo telescopio spaziale ha il compito di usare il metodo dei transiti per identificare esopianeti, coprendo un’area 400 volte più grande rispetto a quella di Kepler. Lanciato nel 2018, è ancora operativo ed ha identificato correttamente 221 esopianeti ed oltre 5000 candidati.
- **PLATO (PLAnetary Transits and Oscillations of stars):** PLATO è un telescopio spaziale in fase di sviluppo da parte dell’ ESA, il cui lancio è previsto nel 2026. Esaminerà la volta celeste alla ricerca di esopianeti utilizzando il metodo dei transiti.

# CAPITOLO 3

---

## Stato dell'arte

---

"Io non temo i computer. Temo la loro mancanza." - Isaac Asimov

I dati prodotti dai telescopi presentati nel Capitolo 2.3 sono analizzati dagli scienziati che in base a una serie di regole decidono se il segnale corrisponde a un esopianeta oppure si tratta di un falso positivo causato dagli strumenti stessi.

Inizialmente gli scienziati hanno provato ad automatizzare il processo decisionale che portava alla eliminazione di falsi positivi, passando poi a utilizzare tecniche di *Machine Learning* supervisionato e non supervisionato per classificare le curve di luce prodotte.

Nel 2018 Shallue and Vanderburg hanno utilizzato una Rete Neurale Convoluzionale (CNN, *Convolutional Neural Network*) per identificare all'interno di curve pattern relativi al transito di esopianeti. Da questa rete neurale convoluzionale (*Convolutional Neural Network*, CNN), chiamata *Astronet*, sono state sviluppate una serie di CNN che utilizzano questa come base, ma sono state realizzati anche nuovi modelli convoluzionali.

Tutte le reti si basano sui dati ottenuti tramite il metodo dei transiti (Capitolo 2.2) dai vari telescopi che sono stati progettati nel corso degli anni (Capitolo 2.3).

Si parla in questo caso di TCE, (*Threshold Crossing Event*): una sequenza di simil transiti nella serie temporale che rappresenta il flusso di un dato target che somiglia al transito di un pianeta a tal punto che si eseguono ulteriori analisi.

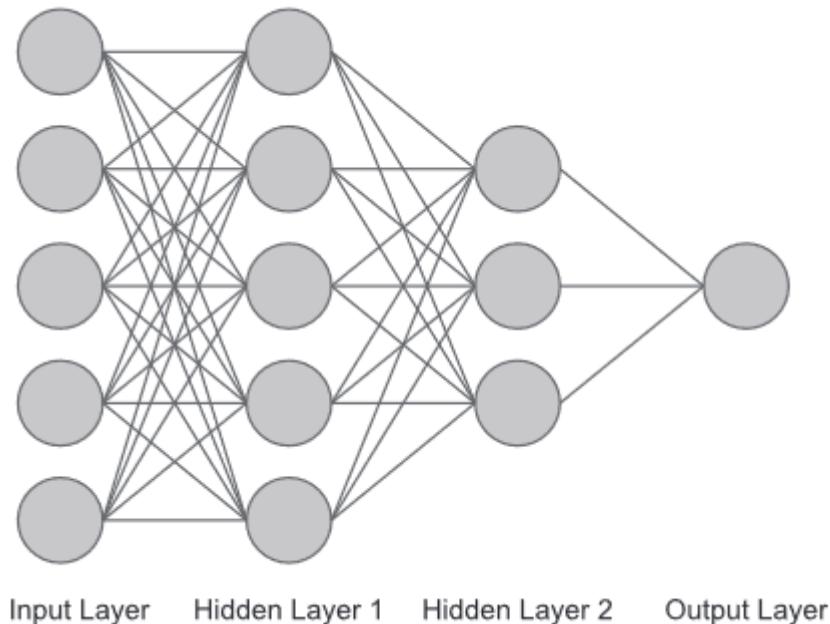
Prima di esaminare lo stato dell'arte, spieghiamo brevemente cosa sono le reti neurali artificiali e le reti neurali convoluzionali.

### 3.1 Reti neurali artificiali

Le reti neurali artificiali (*Artificial Neural Network*, ANN), sono sistemi informatici il cui spunto creativo deriva dall’osservazione del cervello umano. Il cervello è infatti formato da circa  $10^{11}$  cellule chiamate neuroni [6]. I neuroni hanno dei prolungamenti chiamati *dendriti* e *assoni* che hanno il compito di ricevere e trasmettere segnali. In questo modo i neuroni possono comunicare tra di loro. Allo stesso modo, una ANN si compone di neuroni o *unità* o *nodi* interconnessi tra di loro. Ogni nodo ha un input e un output e al suo interno è associata una funzione  $f(x)$  non lineare. Ad ogni connessione è associato un valore numerico chiamato peso. Quando un nodo riceve l’input effettua una somma pesata, ovvero moltiplica ogni input per il peso della connessione su cui lo ha ricevuto e somma tutti i risultati e al risultato di questa somma applica la funzione e il valore prodotto dalla funzione corrisponde al suo output. Un’unità di questo tipo viene chiamata TLU (*Threshold Logic Unit*).

In una rete neurale artificiale i neuroni sono organizzati in strati: in ognuno sono presenti un certo numero di neuroni, connessi con i neuroni dello strato precedente e con quelli dello strato successivo. Possiamo distinguere tre tipologie di strati: il primo è lo strato di input, i cui neuroni hanno il solo compito di ricevere gli input e inviarli ai neuroni nello strato successivo, ci sono poi quelli che sono definiti strati nascosti (*hidden layers*, strati in cui viene effettuata la computazione descritta in precedenza, questi strati hanno connessioni con i neuroni dello strato precedente e quello successivo e uno strato di output, in cui è effettuata solo la somma pesata degli input, restituendo la predizione fatta dalla rete. Quando tutti i neuroni di uno strato hanno connessioni con tutti i neuroni dello strato precedente e con tutti i neuroni dello strato successivo, per tutti gli strati nella rete neurale, si dice che la rete è completamente connessa.

Una rete neurale artificiale viene addestrata usando un algoritmo chiamato *backpropagation*. Questo algoritmo cerca di minimizzare una funzione di perdita in modo iterativo andando a calcolare quali sono i valori ottimali dei pesi delle connessioni. Nello specifico, funziona nel seguente modo: inizialmente viene effettuato il "passo in avanti" (*forward pass*): dallo strato di input gli input viaggiano nella rete passando per gli stati nascosti fino allo strato di output, producendo dei valori. Fatto ciò, la rete calcola l’errore dell’output utilizzando una funzione di perdita (*loss function*) per paragonare l’output giusto rispetto a quello ottenuto e restituisce la misura dell’errore. Comincia ora il "passo all’indietro" (*reverse pass*): partendo dalle connessioni collegate allo strato di output, la rete calcola quanto ognuna di esse ha contribuito all’errore, passando poi alle connessioni collegate allo strato precedente fino ad



**Figura 3.1:** L’architettura di una rete neurale artificiale completamente connessa. Nell’architettura sono presenti lo strato di input, due strati nascosti e lo strato di output. Fonte: "Identifying exoplanets with deep learning: A five-planet resonant chain around kepler-80 and an eighth planet around kepler-90".

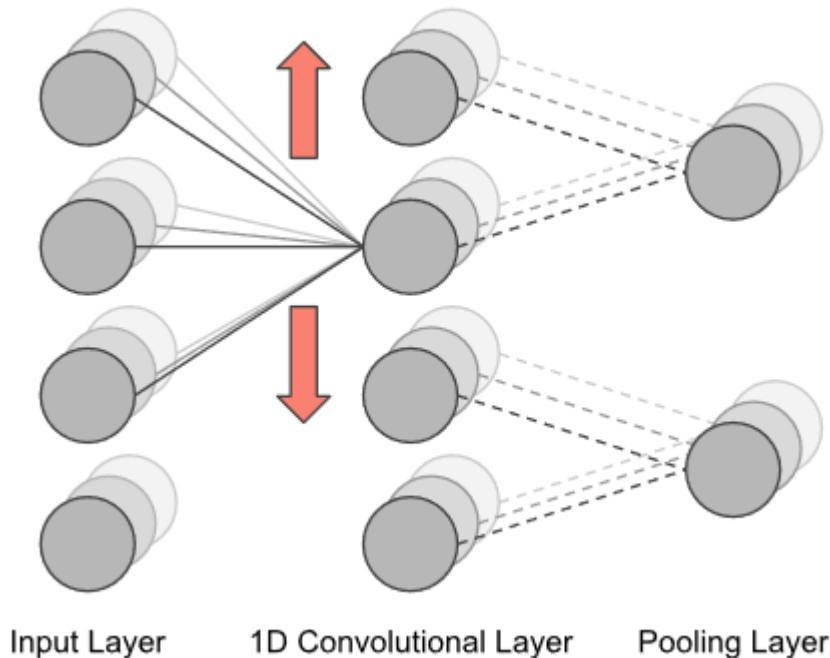
arrivare alle connessioni dello strato di input. Così facendo si calcola l’errore del gradiente attraverso tutte le connessioni della rete andando a propagare all’indietro l’errore attraverso tutta la rete (da qui il nome dell’algoritmo). Calcolati tutti gli errori, l’algoritmo modifica i pesi tendendo di minimizzare la funzione di perdita (per una maggiore comprensione dell’algoritmo si rimanda all’articolo originale del 1986 [7]).

## 3.2 Reti neurali convoluzionali

Le reti neurali convoluzionali (*Convolutional Neural Network*, CNN) sono un’evoluzione dell’architettura delle ANN create specificatamente per risolvere problemi che le ANN non sono in grado di trattare, uno dei quali è il riconoscimento di *pattern* nelle immagini. La migliore architettura delle CNN quindi permette di superare la complessità computazionale richiesta dalle ANN per trattare questo tipo di dato [8].

Nelle CNN sono stati introdotti due nuove tipologie di strati: gli strati convoluzionali e gli strati di *pooling*, mantenendo anche i tipici strati delle ANN. Altra sostanziale differenza sta nel fatto che i neuroni delle CNN sono organizzati in tre dimensioni: altezza, larghezza e profondità. Le prime due fanno riferimento alle dimensioni dell’input, la terza si riferisce alla

profondità di un volume d'attivazione [8].



**Figura 3.2:** Parte dell'architettura di una rete neurale convoluzionale. Possiamo notare: (a) l'organizzazione dei neuroni nelle tre dimensioni, (b) le operazioni di convoluzione e di *pooling*.  
Fonte: "Identifying exoplanets with deep learning: A five-planet resonant chain around kepler-80 and an eighth planet around kepler-90".

Gli strati convoluzionali sono la parte fondamentale delle CNN. In questi strati viene definita una matrice chiamata *kernel* o *filtro* contenente dei pesi, di dimensioni più piccole rispetto all'input. Il compito di questo strato è di effettuare l'operazione di convoluzione: si effettua una moltiplicazione tra i valori nel *kernel* e i valori contenuti nell'input. Poiché si tratta di un'operazione di moltiplicazione tra matrici, entrambe devono avere la stessa dimensione, quindi si prende in considerazione solo una parte dell'input. Questa operazione viene ripetuta per tutta l'immagine: si moltiplicano i valori all'interno del filtro per una regione dell'immagine di uguali dimensioni. La rete riesce così ad apprendere quali filtri si "attivano" quando si scopre una determinata *feature* in una posizione dell'input [8]. L'insieme di tutti gli scalari ottenuti formano a loro volta un'altra matrice, chiamata *feature map*. Ogni *kernel* avrà la sua *feature map* che verrà aggiunta a tutte le altre lungo la profondità per formare il volume di output dello strato convoluzionale. Dopodiché ogni valore nelle *feature maps* verrà dato a una funzione non lineare. In questo modo il numero di connessioni è drasticamente ridotto perché i neuroni nello strato convoluzionale si collegano solamente a una piccola parte dell'input, la cui dimensione è chiamata *dimensione del campo recettivo* [8].

Lo strato di *pooling* ha il compito di ridurre di dimensione l'input, ovvero ridurre il numero di parametri e la complessità del modello. Il *pooling layer* opera su ogni *feature map* ottenuta dallo strato convoluzionale e produce lo stesso identico numero di *feature maps* ma con dimensioni ridotte. L'operazione di *pooling* consiste nel far scorrere lungo le *feature maps* una finestra di dimensioni ridotte e dai valori evidenziati selezionarne uno in base al tipo di *pooling* che si sta effettuando. Tra quelle più utilizzate ci sono il *max pooling* dove si seleziona il valore massimo e l'*average pooling* dove si calcola il valore medio di quelli nella finestra.

In una CNN, tipicamente, sono inseriti due strati convoluzionali seguiti da uno strato di *max pooling* [8]. Infine, dopo aver effettuato le operazioni di convoluzione e di *pooling* si inseriscono strati connessi come nelle tradizionali ANN. Per fare in modo che l'output ottenuto dalle operazioni precedenti possa andare in input agli strati connessi si effettua un'operazione di "appiattimento" - si trasforma l'output da un array a più dimensioni a un array mono-dimensionale - dopodiché la rete funziona esattamente come nelle ANN.

### 3.3 Astronet

Astronet [5] è una CNN sviluppata da Shallue and Vanderburg nel 2018. La rete è stata addestrata per predire se un segnale è un esopianeta oppure un falso positivo causato da fenomeni astrofisici o strumentali.

Le curve di luce sono state scaricate dal Mikulski Archive for Space Telescopes (MAST), prodotte dalla *pipeline* di *Kepler*. Ognuna di queste curve di luce consiste di misurazioni del flusso effettuate in quattro anni a intervalli di 29,4 minuti, per un totale di 70.000 punti.

Shallue and Vanderburg hanno poi effettuato ulteriori operazioni per fare in modo che questi potessero essere utilizzati come input alla rete neurale:

- per ogni TCE nel *training set* sono stati eliminati punti di transito di ulteriori pianeti confermati nel sistema.
- la curva è stata poi "appiattita" ovvero sono state rimosse le irregolarità di bassa frequenza, utilizzando una *B-spline* e dividendo la curva di luce per la migliore spline.

Per preservare il transito, sono stati rimossi i punti di transito dei TCE mentre si montava la spline e si interpolava linearmente sui transiti. Iterativamente è stata montata la spline, rimossi  $3\sigma$  *outliers* e ri-montata la spline mentre si interpolavano sopra gli outliers per far sì che la spline non venisse "tirata" da punti disperdenti.

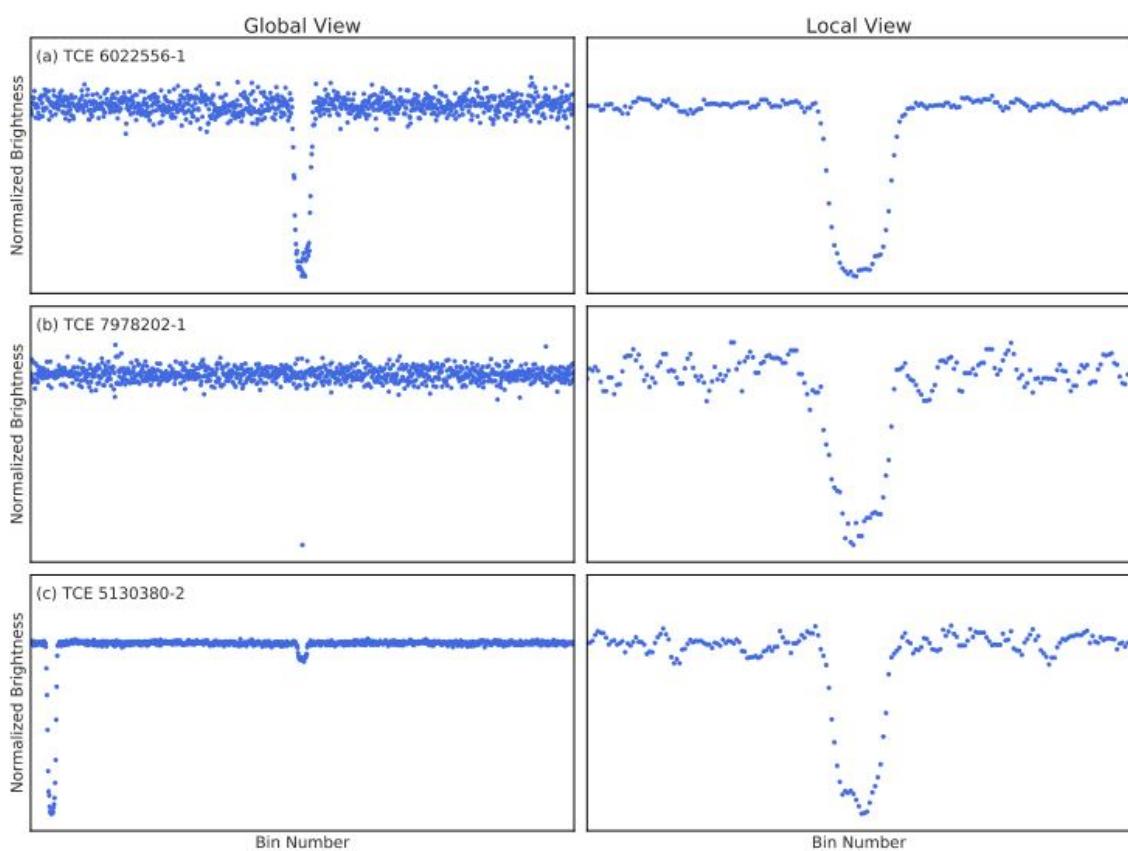
Le curve di luce appiattite sono state poi ripiegate rispetto al periodo del TCE e intervallate

per produrre un array mono-dimensionale.

Per poter creare gli intervalli sulle curve di luce appiattite, Shallue and Vanderburg hanno definito una sequenza di intervalli uniformi sull'asse del tempo con larghezza  $\delta$  e distanza  $\lambda$  dai punti medi e calcolata la media del flusso dei punti che cadono in ogni intervallo.

Sono stati scelti due valori per  $\lambda$ , specifici dei TCE, che generano due diverse "viste" della curva di luce:

- con  $\lambda$  pari a una frazione del periodo del TCE viene generata una *global view*
- con  $\lambda$  pari a una frazione della durata del TCE viene generata una *local view*



**Figura 3.3:** Esempi di *global view* e *local view*. Fonte: "Identifying exoplanets with deep learning: A five-planet resonant chain around kepler-80 and an eighth planet around kepler-90"

Vengono così rappresentati i periodi corti e lunghi del TCE equamente, ma nel caso della *local view* sono visualizzate solo parti della curva di luce e alcune informazioni potrebbero essere perdute. Queste due viste sono date in input separatamente al modello.

Infine, le curve sono state normalizzate per avere mediana 0 e valore minimo -1, in modo tale che ogni TCE ha una profondità di transito "fissa".

Le etichette date alle curve di luce sono prese dal *Autovetter Planet Candidate Catalog for Q1-Q17 DR24*, sulla base dei valori disponibili nella colonna **av\_training\_set**:

- PC (Planet Candidate)
- AFP (Astrophysical False Positive)
- NTP (Non-Transiting Phenomenon)
- UNK (Unknown)

Queste etichette sono state prodotte manualmente. Shallue and Vanderburg hanno ignorato le etichette UNK e hanno deciso di binarizzare le etichette come *Planet* e *Not planet*. Queste etichette sono considerate come *ground truth*, intendendo che sono considerate corrette, anche se hanno riscontrato degli errori nella classificazione. Il dataset è stato diviso randomicamente in tre sottoinsiemi:

- un sottoinsieme da utilizzare per la fase di *training*: il 80% del totale;
- un sottoinsieme da utilizzare per la fase di validazione (*validation*): il 10% del totale. Questa fase viene utilizzata per trovare quali sono i migliori iperparametri del modello;
- un sottoinsieme da utilizzare per la fase di *test*: il 10% del totale. Questa fase viene utilizzata per determinare le prestazioni del modello.

Shallue and Vanderburg hanno utilizzato tre approcci basati sulle reti neurali per classificare le curve di luce prodotte:

- architettura lineare: una rete neurale senza strati nascosti;
- architettura completamente connessa: una rete neurale completamente connessa;
- architettura convoluzionale: una rete neurale convoluzionale.

Tutti gli strati nascosti utilizzano la funzione di attivazione ReLU, mentre lo strato di output usa la funzione sigmoidea. L'output di ogni rete è la probabilità predetta che una curva di luce rappresenta il transito di un pianeta: più il valore è vicino a 1 più il modello è sicuro che l'input è un pianeta, valori vicini allo 0 indicano che l'input è un falso positivo.

Le reti sono state implementate usando TensorFlow [9], una libreria Python usata per il calcolo numerico e il *Machine Learning*. Shallue and Vanderburg hanno utilizzato il sistema Google-Vizier per l'ottimizzazione black-box per poter aggiustare in modo automatico gli iperparametri, inclusi quelli riguardanti la rappresentazione dell'input, l'architettura del

modello e i parametri relativi all’addestramento. Una volta che Vizier ha identificato i migliori iperparametri, sono state allenate dieci copie indipendenti con parametri inizializzati in modo randomico. Per determinare i valori delle predizioni sono stati presi i valori medi di output dei dieci modelli, utilizzando una tecnica conosciuta come "*model averaging*".

Per valutare le *performance* dei modelli sono state usate le seguenti metriche:

- *Precision*: la frazione dei segnali classificati come pianeti che sono veri pianeti.
- *Recall*: la frazione di pianeti veri che sono classificati come pianeti;
- *Accuracy*: la frazione delle classificazioni corrette;
- *AUC (Area Under the receiver operating characteristic Curve)*: equivale alla probabilità che a un pianeta selezionato randomicamente è assegnato un valore più alto rispetto a un falso positivo selezionato randomicamente.

Le prime tre metriche dipendono dalla soglia scelta per il modello. Una soglia naturale per classificare un TCE come pianeta candidato è che la probabilità predetta è maggiore di 0,5, mentre per i falsi positivi, la probabilità predetta è minore di 0,5, ma questo valore può essere modificato come *trade-off* tra *precision* e *recall*. Il valore dell’ AUC è indipendente dalla scelta della soglia di classificazione. Le tabelle 3.1 e 3.2 riportano i valori ottenuti sui vari modelli.

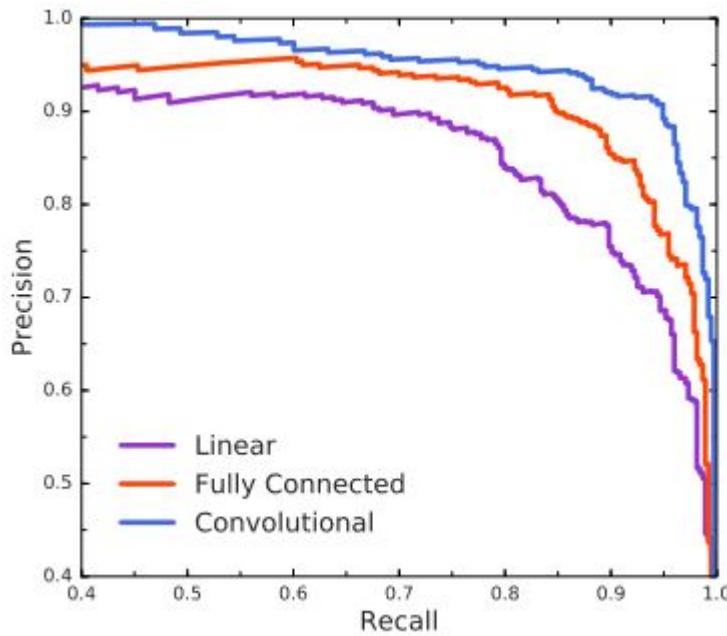
	Vista globale	Vista locale	Vista globale e locale
Modello lineare	0.869	0.879	0.917
Modello completamente connesso	0.902	0.912	0.941
Modello convoluzionale	0.954	0.924	0.960

**Tabella 3.1:** Tabella relativa ai valori di *accuracy* sul test set

La Figura 3.4 mostra le curve di *precision vs recall* calcolate per tutti e tre i modelli.

I valori riportati nella Tabella 3.1 e nella Tabella 3.2 mostrano che utilizzare entrambe le viste porta a migliori risultati in tutte le architetture e che a ottenere i migliori risultati è la rete convoluzionale: Shallue and Vanderburg hanno infatti osservato che sia il modello lineare che il modello completamente connesso hanno difficoltà a distinguere i transiti a "forma di U", (che rappresentano pianeti) rispetto a transiti a "forma di V" (che rappresentano stelle binarie ad eclissi) quando l’ampiezza del transito non è normalizzata nella vista globale. Il modello convoluzionale è capace di distinguere questi transiti con diverse larghezze ed è

	Vista globale	Vista locale	Vista globale e locale
Modello lineare	0.922	0.933	0.963
Modello completamente connesso	0.961	0.965	0.977
Modello convoluzionale	0.985	0.973	0.988

**Tabella 3.2:** Tabella relativa ai valori di AUC sul test set**Figura 3.4:** Le curve del *Precision vs Recall* calcolate per *Astronet*. Fonte: "Identifying exoplanets with deep learning: A five-planet resonant chain around kepler-80 and an eighth planet around kepler-90"

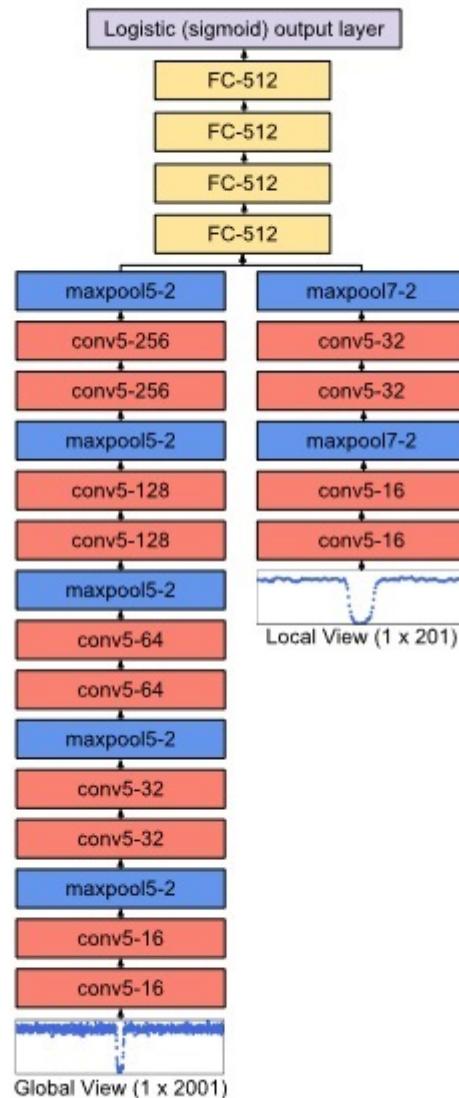
anche capace di identificare eclissi secondarie nella vista globale.

L’architettura migliore della rete convoluzionale è visibile in Figura 3.5

I valori dei parametri sono stati scelti per massimizzare il valore dell’ AUC sul sottoinsieme di validazione. È stata usata una vista globale di lunghezza 2001 con  $\lambda = \delta = p/2001$ , dove  $p$  è il periodo del TCE e una vista locale di lunghezza 201 con durata dei transiti  $k=4$  su entrambi i lati dell’evento,  $\lambda = 2kd/201$  e  $\delta = 0.16d$ , dove  $d$  è la durata del TCE.

Il modello è stato addestrato usando una *batch* di dimensione 64 per 50 epoch. È stato usato l’algoritmo di ottimizzazione Adam con  $\alpha = 10^{-5}$ ,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$  e  $\epsilon = 10^{-8}$ . Non è stata utilizzata la regolarizzazione del *dropout* in questa architettura [5].

Shallue and Vanderburg hanno poi utilizzato la rete creata per classificare nuovi candidati in base alla probabilità di essere veri pianeti, il che ha portato alla scoperta di due nuovi candidati: *Kepler-80 g* e *Kepler-90 i*, che è l’ottavo pianeta del sistema *Kepler-90*.



**Figura 3.5:** L'architettura di Astronet, fonte: "Identifying exoplanets with deep learning: A five-planet resonant chain around kepler-80 and an eighth planet around kepler-90".

Gli strati convoluzionali sono indicati come `conv<kernel-size>-<number of feature map>`, gli strati di max pooling sono indicati come `maxpool<window lenght>-<stride lenght>` e gli strati completamente connessi sono indicati come `FC-<number of units>`

### 3.4 Astronet-K2

Astronet-K2 [10] è l'estensione del lavoro di Shallue and Vanderburg, per classificare segnali dalla missione K2. Le motivazioni legate alla scelta dei dati della missione K2 sono dovute al fatto che durante questa missione il telescopio Kepler non ha osservato una singola regione dello spazio, ma diverse, ottenendo quindi dati da differenti ambienti galattici.

La produzione del dataset utilizzato fa ancora uso dei TCE. Per produrli, sono state scaricate dal Mikulski Archive for Space Telescope (MAST) i *Target Pixel Files* di K2 e da quest'ultimi sono state estratte le curve di luce grezze andando a sommare il flusso contenuto nelle venti diverse aperture fotometriche stazionarie ad ogni marcatura temporale [10]. A causa dell'instabilità di Kepler durante la missione K2, le curve di luce grezze hanno delle grandi caratteristiche sistematiche che impediscono di identificare i transiti. Dattilo et al. hanno corretto queste caratteristiche e prodotto le curve di luce corrette.

Dopodiché, Dattilo et al. hanno cercato i transiti all'interno delle curve di luce, calcolando il periodogramma Box Least Square (BLS), cercando segnali che eccedevano il limite del rapporto segnale/rumore (*signal-to-noise ratio, S/N*) pari a 9, eliminando tali segnali e ricalcolando il BLS nuovamente, fino a quando segnali significativi non rimanevano all'interno della curva di luce. Tutti i segnali più forti della soglia delle S/N sono stati considerati TCE. Il BLS è stato eseguito sulle campagne 0-16 e in totale sono stati realizzati 51.711 TCE. La maggior parte di questi (31.575) sono stati classificati a mano per produrre l'insieme di dati etichettato con cui addestrare la rete. Il *labeling* dei TCE è stato svolto in due passaggi:

1. *Triage*: per le Campagne 0-3 Dattilo et al. inizialmente hanno scrutinato ogni TCE ottenuto dalla ricerca dei transiti. Dalla Campagna 4, hanno considerato solo il primo TCE identificato attorno a ogni stella se e solo se questo era stato classificato come pianeta candidato, poiché il primo TCE è il segnale più forte nella maggior parte dei casi, laddove il segnale è un falso positivo, anche tutti gli altri saranno falsi positivi. Questa operazione ha permesso di velocizzare la classificazione.

Ad ogni TCE è stato assegnato una delle seguenti etichette: "C" per pianeta candidato, "E" per stella binaria a eclissi e "J" per spazzatura (*junk*). Lo scopo di questa fase era quella di organizzare tutti i segnali che rappresentavano non candidati : molti TCE sono causati da artifici strumentali; quindi rimuoverli ha ridotto di oltre il 90% il numero di TCE che devono essere ulteriormente analizzati nel secondo passaggio.

2. *Vetting*: tutti i TCE che nella fase di *Triage* sono stati classificati come "C" in questa fase sono stati ulteriormente analizzati. Dattilo et al. hanno usato una serie di regole per

determinare quale segnale è un candidato:

- (a) Un segnale è un pianeta candidato fino a quando non viene provato il contrario, o per motivi astrofisici (quindi indicando che il segnale è causato da errori sistematici strumentali) o per la violazione delle altre regole.
- (b) Ogni segnale con una profondità di transito superiore al 3% viene classificato come stella binaria a eclissi.
- (c) Se la stella stava ruotando simultaneamente con l'orbita e ha un transito a forma di V, allora è classificata come una stella binaria a eclissi.
- (d) Ogni TCE con modulazione di fase che si crede essere causata da stelle distorte dalla gravità, radiante relativistico o altri fenomeni astrofisici, sono etichettate come stelle binarie a eclissi.
- (e) Tutti i segnali ambigui su cui non è stato possibile stabilire se fossero pianeti candidati o falsi positivi, sono stati rimossi dal *training set*.
- (f) TCE causati da un singolo transito sono rimossi dal *training set*.
- (g) pianeti in disintegrazione, come WD 1145+017 b e K2-22 sono rimossi dal *training set*.

Dattilo et al. hanno rimosso i TCE ottenuti dalle campagne 0 e 11, perché il telescopio puntava in quel momento a una regione del cielo molto affollata. La campagna 0 inoltre ha avuto una durata molto inferiore rispetto alle altre. Anche la campagna 9 è stata rimossa, perché i dati sono stati ottenuti tramite la tecnica della microlente gravitazionale. Infine, sono stati corretti a mano alcuni casi dove si era identificato erroneamente il periodo orbitale dei TCE [10].

Dopo aver etichettato ogni TCE, questi sono stati processati per poter essere utilizzati come input per la rete neurale. La strategia seguita da Dattilo et al. è simile a quella usata da Shallue and Vanderburg: per prima cosa è stata rimossa la variabilità a lunga durata da ogni curva di luce, causata dalla variabilità stellare o dalla sistematicità degli strumenti. E' stata poi *fittata* una spline per rimuovere la bassa variabilità della stella. I dati sono stati poi ripiegati lungo il periodo del TCE, in modo che ogni transito sia allineato e centrato. Successivamente, stati rimossi i dati anomali causati da raggi cosmici o dal passaggio di pianeti del Sistema Solare o di asteroidi vicino alla stella. Seguendo quanto fatto da Shallue and Vanderburg, Dattilo et al. hanno creato due viste, la vista globale (*global view*) e la vista locale (*local view*). La vista globale mostra le caratteristiche della curva di luce su un intero periodo orbitale ed è l'intera

curva di luce ripiegata rispetto al periodo con ogni punto raggruppato all'interno di uno dei 701 intervalli e rispetto ad ogni intervallo è stata calcolata la mediana di tali punti. La vista locale rappresenta il transito più da vicino e di cui è distinguibile la forma. I punti in questa vista sono stati raggruppati in 51 intervalli ed è stata calcolata la mediana di ogni intervallo. In alcune delle viste sono presenti spazi vuoti, poiché alcuni intervalli hanno zero punti al loro interno. I valori mancanti sono stati calcolati andando a interpolare linearmente tra gli intervalli vicini. Per normalizzare le curve di luce, i valori degli intervalli sono stati riscalati in modo tale che il punto mediano è 0 e il punto minimo è -1. Queste differenze rispetto al lavoro svolto da Shallue and Vanderburg sono state causate dal minor numero di punti nelle curve di luce di K2 [10].

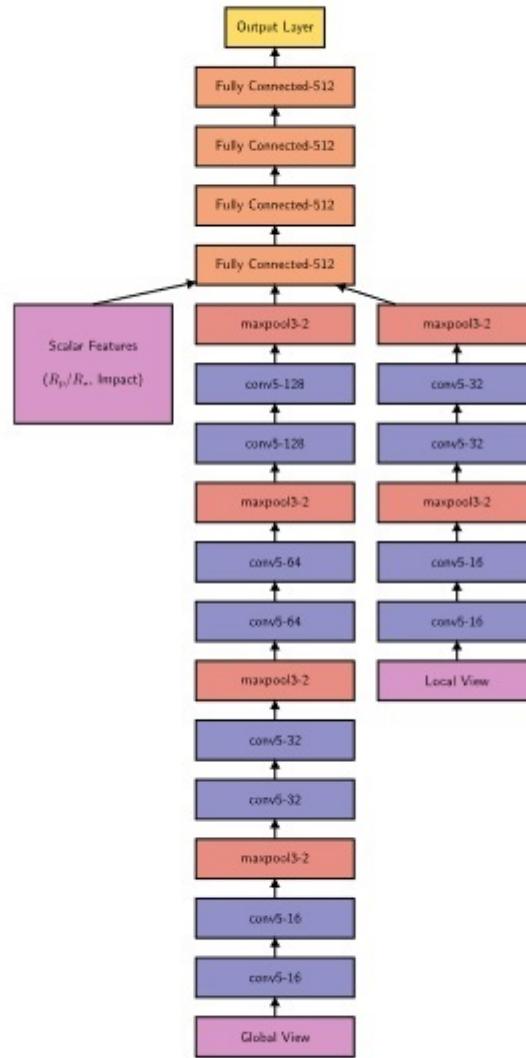
Inoltre, Dattilo et al. hanno deciso di dare in input alla rete anche delle caratteristiche scalari, in particolare i migliori valori del rapporto tra raggio del pianeta e della stella ( $R_p/R_\star$ ) e il parametro dell'impatto del transito da un *least square fit* alla curva di luce del TCE ripiegata rispetto al periodo utilizzando un modello sviluppato da Mandel and Agol [11]. L'uso di queste due features permette alla rete non solo di usare informazioni sulla profondità del transito, ma anche di apprendere più efficientemente la forma del transito e la profondità. Il totale di TCE usati sono 27.634 e questi sono stati divisi randomicamente in tre sottoinsiemi:

- *training*: 80% (22.105 TCE)
- *validation*: 10% (2774 TCE)
- *test*: 10% (2755 TCE)

Il *validation set* viene usato per ottimizzare i parametri della architettura della rete e il *test set* per valutare le performance del miglior modello.

L'architettura della rete si basa sul modello di Astronet (Figura 3.5). Dopo un tentativo iniziale di addestrare Astronet sul dataset creato, che non ha portato a risultati, Dattilo et al. hanno ottimizzato la rete addestrandola sul dataset usato da Shallue and Vanderburg. Le curve di luce utilizzate sono state accorciate a segmenti di 80 giorni per farli combaciare con quelle di K2. Il tasso d'apprendimento ottimale individuato è stato di  $\alpha = 10^{-4}$  sull'architettura in Figura 3.6 (escluse le features). Questa architettura e il tasso d'apprendimento sono state poi usate con il dataset di K2. L'architettura della rete, chiamata *Astronet-K2* è in Figura 3.6.

L'output del modello indica la probabilità che un TCE sia un pianeta candidato: valori vicino a 1 indicano che il TCE è un pianeta candidato, viceversa valori vicino allo 0 indicano che si tratta di un falso positivo. La rete è stata addestrata con la funzione di ottimizzazione



**Figura 3.6:** L'architettura di *Astronet-K2*, fonte: “*Identifying exoplanets with deep learning. ii. two new super-earths uncovered by a neural network in K2 data*”.

Gli strati convoluzionali sono indicati come `conv<kernel size>-<number of feature map>`, gli strati di max pooling sono indicati come `max<window lenght>-<stride lenght>` e gli strati completamente connessi sono indicati come `FC-<number of units>`.

Adam, un tasso d'apprendimento pari a  $\alpha = 10^{-4}$  e una dimensione delle batch di 64. Dattilo et al. hanno poi effettuato *data augmentation* sui dati di training andando a ribaltare randomicamente l'ordine temporale delle curve di luce durante il processo di apprendimento: i veri pianeti sono simmetrici rispetto al tempo; quindi invertire l'ordine temporale produce una nuova curva di luce con la stessa etichetta della curva di luce originale. La dimensione del *training set* è quindi raddoppiata.

Anche Dattilo et al. hanno utilizzato la tecnica del *model averaging* per valutare i risultati della rete: sono stati addestrati dieci modelli indipendenti con gli stessi parametri e calcolato la media dei risultati finali per ogni TCE.

Le metriche di valutazione sono le seguenti:

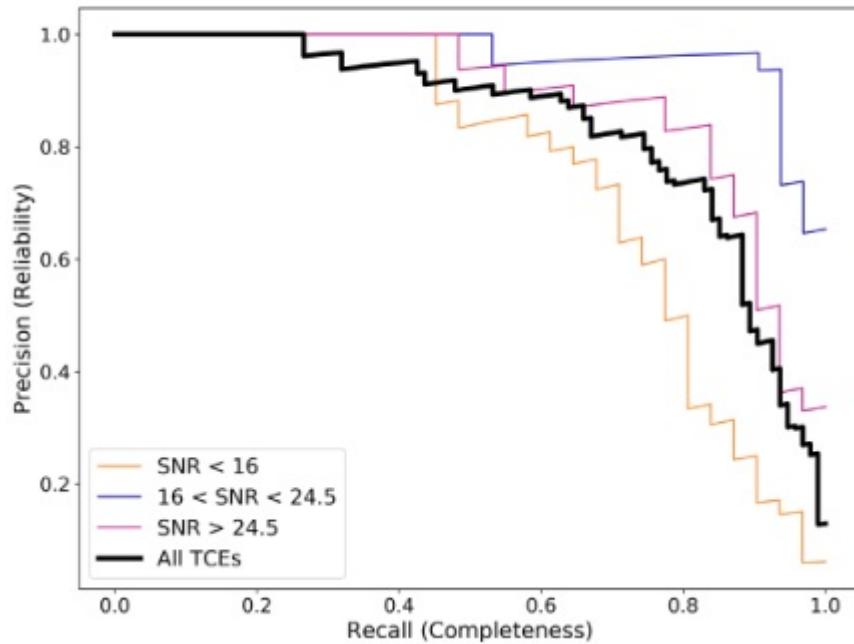
- *Accuracy*: la frazione dei segnali che sono stati classificati correttamente;
- *Precision*: la frazione di tutti i pianeti classificati che erano veri pianeti;
- *Recall*: la frazione di pianeti candidati che la rete ha classificato come pianeti candidati;
- *False-positive rate*: la frazione del totale di falsi positivi che il modello ha classificato come pianeti candidati;
- *AUC (Area Under the receiver-operator characteristic curve)*: la probabilità che un pianeta candidato selezionato randomicamente abbia un valore di predizione superiore a un falso positivo.

Le prime quattro metriche si basano sul valore soglia scelto per il modello: se si considera il valore della predizione superiore a 0,5 come pianeta mentre valori minori di 0,5 come falso positivo, allora il modello raggiunge un *accuracy* di 97.84%. Il valore dell' AUC, indipendente dal valore soglia scelto, sulla media dei modelli è di 98.83%.

La curva del *precision vs. recall* mostra il *trade-off* tra la sensitività e specificità: se il valore soglia è regolato per avere pochi falsi positivi, ci saranno ancora dei falsi positivi.

Se il valore della precisione è 1, e quindi non ci sono falsi positivi, il *recall* è di 0,2, pertanto stiamo perdendo circa l'80% dei pianeti candidati. Questo è dovuto al fatto che il dataset non è bilanciato e perché il rumore strumentale è simile al segnale dei pianeti. Il modello è capace di classificare i pianeti, ma a causa del dataset sbilanciato, anche un'elevata accuratezza può portare a degli errori [10].

Dattilo et al. inoltre hanno valutato le prestazioni del modello su sottoinsiemi del test-set raggruppati in base al valore del rapporto S/N: le curve di *precision vs. recall* sono



**Figura 3.7:** Le curve del *Precision vs Recall* calcolate per *Astronet-K2*. Nel grafico sono presenti le curve calcolate per ogni sottoinsieme del *test set* diviso in base al rapporto S/N. Fonte: “*Identifying exoplanets with deep learning. ii. two new super-earths uncovered by a neural network in K2 data*”.

state calcolate anche per questi insiemi. Per valori del rapporto S/N < 16 il modello ha le performance peggiori, mentre le prestazioni migliori sono con valori tra 16 e 24,5. Per Dattilo et al. questo è dovuto alla maggiore presenza di falsi positivi nella forma di stelle binarie a eclissi che il modello sbaglia a classificare. La Figura 3.7 mostra il grafico delle curve di *precision vs recall* calcolate su ognuno dei sottoinsiemi.

Per questi motivi Dattilo et al. osservano che *Astronet-K2* non è adatta a lavorare in autonomia. Però può essere molto utile se combinata alla supervisione umana: scartare tutti i TCE a cui il modello ha predetto valori inferiori a 0.01, verrebbero eliminati l’85% dei falsi positivi nel dataset al costo di un solo pianeta reale. Si andrebbe così a decrementare di un fattore di 6 il carico di lavoro umano quando si identificano pianeti candidati dai TCE e si ridurrà il numero di TCE non classificati a cui un astronomo dovrebbe lavorare [10].

Infine, Dattilo et al. hanno testato il modello su un nuovo gruppo di TCE di K2. Questi nuovi TCE sono gli altri segnali identificati iterando la rimozione tramite BLS del segnale più forte da ogni target. Questi TCE non classificati provengono da target il cui primo segnale era stato classificato come "J" o "E". Il modello ha classificato 826 segnali sopra l’1%. 229 provenivano

dalla Campagna 12, durante la quale Marte è passato nel campo visivo di *Kepler*, causando effetti sistematici nelle curve di luce, nascondendo il vero transito.

Dattilo et al. si sono poi concentrati su due particolari segnali di questa Campagna, EPIC 246151543 e EPIC 246078672. Dopo ulteriori analisi e osservazioni i TCE si sono rivelati essere due esopianeti del tipo Super-Terra.

### 3.5 Exonet

Exonet [12] è una rete neurale sviluppata da Ansdell et al. presso il 'Nasa Frontier Development Lab' (FDL) edizione 2018, un incubatore di ricerca di 8 settimane, durante il quale si applicano tecniche di Machine Learning alle scienze astronomiche.

Come dataset Ansdell et al. hanno utilizzato le curve di luce del Q1-Q17 Kepler Data Release 24 (DR24). Le operazioni di preparazione dei dati sono molto simili a quelle eseguite da Shallue and Vanderburg: gli scienziati hanno appiattito le curve di luce facendo passare iterativamente una B-spline (escludendo i punti di transito del TCE per preservare il passaggio del pianeta), dividendo poi la curva di luce per la migliore spline mentre si interpola linearmente sui punti di transito [12]. Ansdell et al. hanno utilizzato un diverso procedimento di *spline-fitting*: usando la funzione `LSQUnivariateSpline` della libreria SciPy, velocizzando di cinque volte i tempi di processamento dei dati. Anche loro hanno creato una *global view* e una *local view* di ogni curva di luce ripiegata sul periodo, facendo sì che la profondità massima di transito sia -1, mentre il resto dei valori sia 0.

Inoltre, Ansdell et al. hanno utilizzato le serie temporali della posizione dei pixel del centro della luce (i centroidi) calcolati dalla *pipeline* di Kepler, che permettono di avere informazioni sulla posizione della fonte del segnale simil transito, utile quando si identificano *Background Eclipsing Binary* (BEB): i centroidi si sposteranno nella direzione opposta delle BEB se sia la BEB e la stella sono contenute nella apertura fotometrica usata per misurare il flusso [12].

Le etichette utilizzate da Ansdell et al. sono le stesse utilizzate da Shallue and Vanderburg: quindi, PC, AFP e NTP, che sono poi state rese binarie come *Planet* e *Not planet*. Infine, il dataset è stato poi diviso randomicamente in tre sottoinsiemi:

- *training*: 80%
- *validation*: 10%
- *test*: 10%

L’architettura della rete di Ansdell et al. parte dalla architettura base di Astronet (3.3) a cui sono stati aggiunti diverse *features* oltre ad altre rappresentazioni di input: il modello ottenuto è stato chiamato *Exonet*. A questo modello sono stati aggiunti:

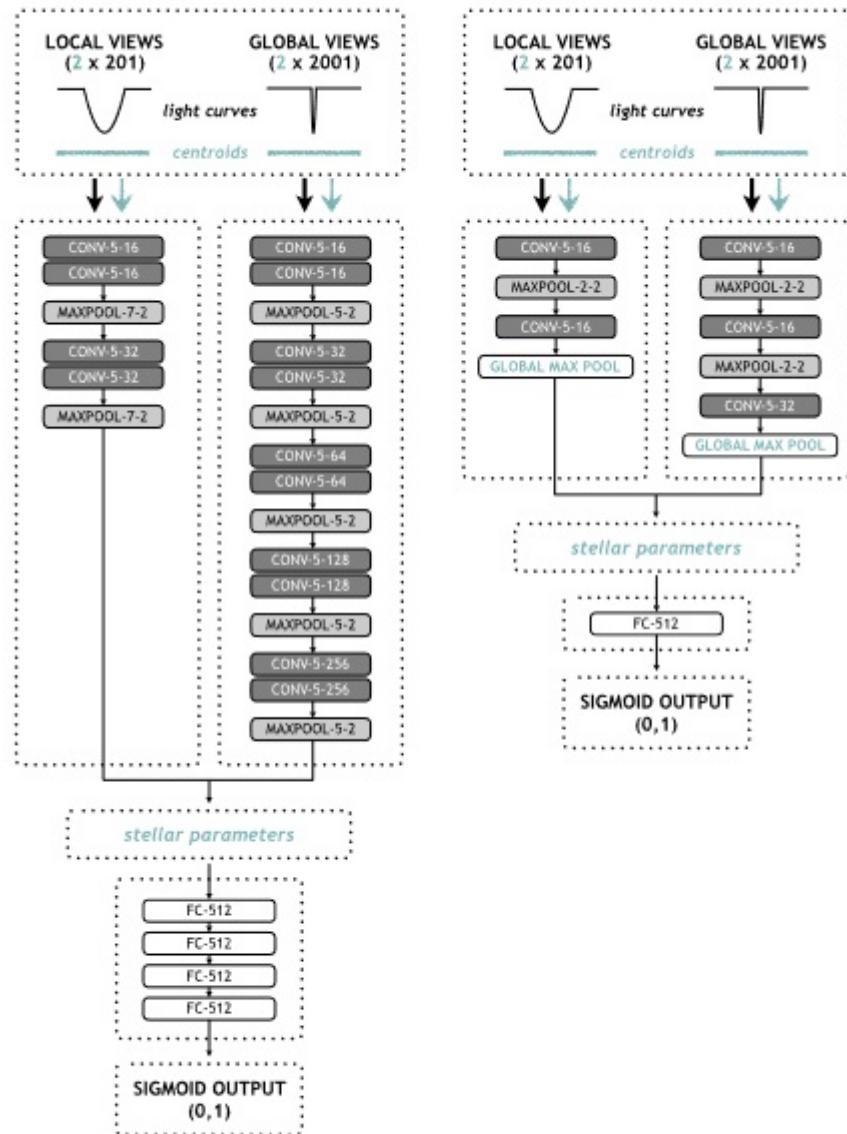
- il passaggio in input delle serie temporali dei centroidi nella forma di viste globali e locali, in modo da aiutare il modello ad apprendere la connessione tra forma delle curve di luce e delle curve dei centroidi, il che può essere utile per identificare falsi positivi. Questa aggiunta è stata indicata come possibile miglioria da Shallue and Vanderburg [12].
- L’aggiunta di parametri stellari come: temperatura effettiva della stella ( $T_{eff}$ ), gravità della superficie ( $logg$ ), metallicità ( $[Fe/H]$ ), raggio ( $R_*$ ), massa ( $M_*$ ) e densità ( $\rho_*$ ). Questi valori sono stati normalizzati: ad ognuno viene sottratta la mediana e divisa per la deviazione standard dove questi valori sono calcolati per ogni parametro nel *training set*, in modo che la distribuzione di ogni parametro abbia mediana 0 e deviazione standard 1.

All’interno della rete questi parametri sono concatenati all’output degli strati convoluzionali prima di passarli agli strati completamente connessi. L’aggiunta di questi parametri è dovuta alla stretta correlazione che esiste con la classificazione, ad esempio: per le stelle giganti con un raggio molto grande è più probabile che il segnale sia dovuto ad una eclissi stellare che al transito di un pianeta [12].

- come Shallue and Vanderburg anche Ansdell et al. hanno effettuato *data augmentation* sui dati di addestramento, invertendo l’asse del tempo di metà delle curve di luce e delle curve dei centroidi durante l’addestramento. Inoltre, è stata applicata un’ulteriore tecnica di *data augmentation* per imitare le incertezze di misurazione durante la misura del flusso: hanno aggiunto rumore Gaussiano randomico alle curve di luce di input dove la deviazione standard è scelta randomicamente in una distribuzione uniforme tra 0 e 1 [12].

Inoltre, Ansdell et al. hanno osservato di poter ridurre le dimensioni della rete mantenendo comunque ottime performance. In questa rete, chiamata *Exonet-XS*, il numero di strati convoluzionali della colonna relativa alla vista locale è stato ridotto da 4 a 2, mentre quella globale da 10 a 3. Inoltre, in output ad ogni colonna convoluzionale è stato aggiunto un *max global pooling* che permette di ridurre il numero di parametri e incrementare la generalizzazione.

In Figura 3.8 è possibile vedere le architetture delle due reti.



**Figura 3.8:** L'architettura di Exonet (sinistra) e Exonet-XS (destra), fonte: "Scientific domain knowledge improves exoplanet transit classification with deep learning".

Gli strati convoluzionali sono indicati come CONV-<kernel size>-<number of feature map>, gli strati di max pooling sono indicati come MAXPOOL-<>window lenght>-<stride lenght> e gli strati completamente connessi sono indicati come FC-<number of units>.

Per valutare i risultati delle performance sono state usate tre metriche: *accuracy*, *average precision* e le *curve di precision vs. recall*. Come valore soglia è stato usato un valore di 0,5.

	Accuracy	Average precison
Astronet	0.958	0.955
Exonet	0.975	0.980
Astronet-XS	0.953	0.963
Exonet-XS	0.966	0.963

**Tabella 3.3:** Confronto dei risultati delle due reti sul *test set*

La Tabella 3.3 mostra i risultati ottenuti dalle due reti sul *test set*: per la rete *Exonet* c'è stato un miglioramento del 1,7% nella *accuracy* e del 2,5% nella *average precision* rispetto ad *Astronet*. Lo stesso vale per *Exonet-XS*: ci sono state migliorie del 0.8% sia nella *accuracy* che nella *average precision* (*Astronet-XS* è il modello ridotto di *Astronet* senza le aggiunte fatte da Ansdell et al.).

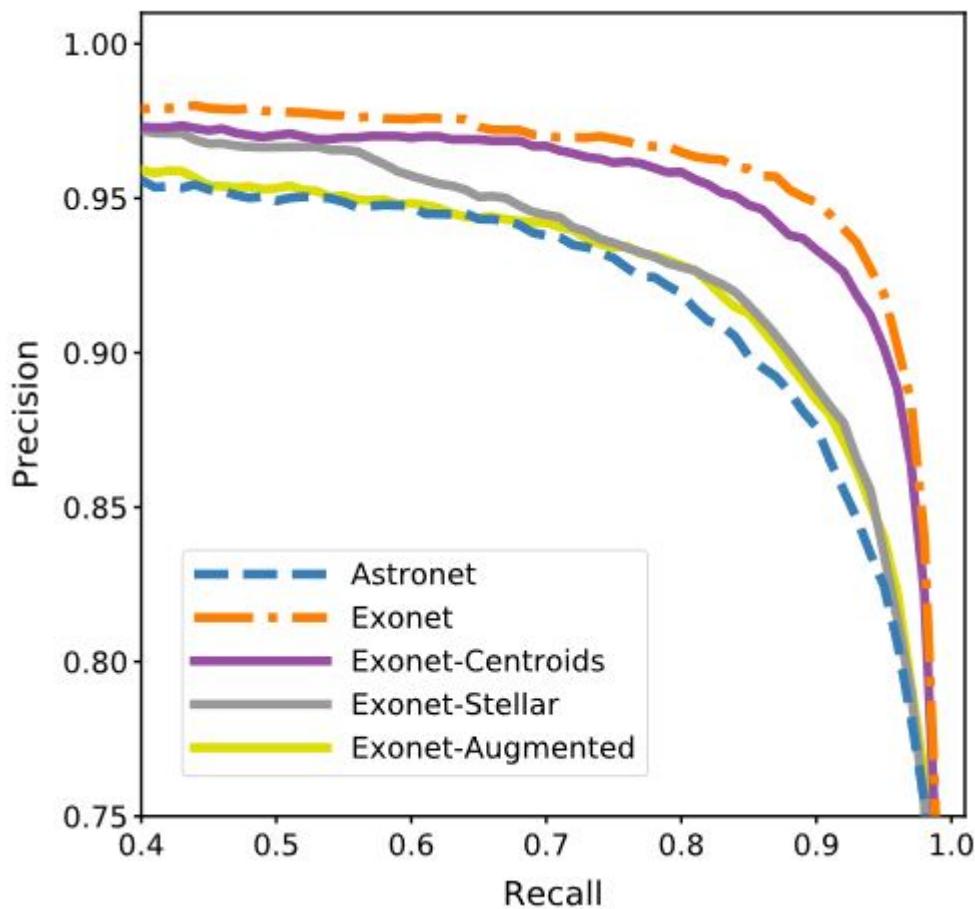
Le curve del *precision vs recall* sono state calcolate aggiungendo individualmente una delle tre caratteristiche descritte prima. Questo è stato fatto per mostrare come ognuna delle *features* contribuisce a migliorare le prestazioni. In Figura 3.9 è possibile osservare il grafico di tali curve.

Per fare ciò Ansdell et al. hanno utilizzato il *k-fold cross validation* sul *training set* unito al *validation set*. Il maggiore contributo è stato dato dall'aggiunta delle serie temporali dei centroidi, seguito dall'aggiunta dei parametri stellari. La *data augmentation* ha invece contribuito a ridurre l'*overfitting* [12].

## 3.6 Astronet-Vetting

Astronet-Vetting [13] è la variante di Astronet sviluppata da Yu et al. per poter identificare pianeti candidati dai dati ottenuti dal telescopio spaziale TESS. Oltre a questa versione, gli autori hanno sviluppato anche un modello capace di distinguere segnali che rappresentano il transito da segnali causati da variabilità stellare o rumore strumentale. Il modello è chiamato Astronet-Triage.

I due modelli utilizzano curve di luce prodotte con il MIT Quick Look Pipeline (QLP) (si



**Figura 3.9:** Le curve del *Precision* vs *Recall* calcolate sui vari modelli: *Exonet-Centroid* è il modello con la sola aggiunta delle curve dei centroidi, *Exonet-Stellar* è il modello con la sola aggiunta dei parametri stellari, *Exonet-Argumented* è il modello con la sola aggiunta della *data-argumentation* e *Exonet* è il modello con l'aggiunta di tutte e tre le caratteristiche. Fonte: "Scientific domain knowledge improves exoplanet transit classification with deep learning".

rimanda all'articolo per una descrizione completa del processo di creazione delle curve di luce [14]).

Dopo aver ottenuto le curve di luce, il QLP cerca all'interno di queste cali periodici usando un algoritmo BLS (*Box Least Square*), ottenendo così i TCE. Il passo successivo è stato quello di etichettare i TCE ottenuti. Una parte di TCE è stata sottoposta a controlli di gruppo, dove un team di scienziati ha esaminato i segnali usando i report prodotti dal QLP e votato sulla loro disposizione, ma questo processo ha portato a inconsistenze: un TCE che appare in più settori può avere diverse disposizioni nei diversi settori. Per assicurare omogeneità nel processo di *labeling*, la dottoressa Liang Yu ha ispezionato visivamente ogni TCE e gli ha assegnato una delle seguenti etichette:

- PC: Pianeta candidato (Planet Candidate)
- EB: Binaria ad eclissi (Eclipsing Binary)
- V: variabilità stellare (Stellar Variability)
- IS: rumore strumentale (Instrumental Noise)

utilizzando le seguenti regole:

- Ogni segnale simile a quello di un pianeta che non ha una forte eclissi secondaria, una differenza nella forma del transito pari/dispari, una profondità di transito che aumenta con l’apertura, viene classificato come PC [13].
- Alcune nane brune e nane rosse (*M dwarfs*) classificate inizialmente come EB confermate anche dall’analisi di gruppo, sono state poi ri-etichettate come PC, poichè, è molto difficile anche per analisti esperti, distinguerle da pianeti giganti in transito [13].
- Al pianeta di tipo giove caldo WASP-18b è stato assegnato etichetta PC, poiché ha una eclissi secondaria visibile [13].
- I segnali in cui la profondità del transito aumenta con l’apertura sono etichettati come EB [13].
- Alcune stelle binarie ad eclissi esibiscono anche variabilità stellare: queste sono etichettate come V se l’ampiezza della variabilità è maggiore della metà della profondità dell’eclissi, altrimenti l’etichetta assegnata è EB [13].
- Tutti i TCE risultanti ambigui, tali che nemmeno gli analisti umani riescono a etichettare sono stati rimossi dal *training-set* [13].
- I segnali etichettati come PC e EB che sono distorti, come ad esempio segnali che non sono più riconoscibili come transiti oppure se la cui profondità cambia del 50%, sono rimossi dal *training-set* [13].
- I segnali di transito profondi che non mostrano segni di essere stelle binarie ad eclissi sono classificati come PC [13].
- I segnali anomali che non cadono in nessuna delle quattro categorie sono classificati come V [13].

Yu et al. considerano queste etichette come *groud truth* anche se imperfette: gli autori hanno osservato come all'interno del dataset possano essere presenti TCE classificati incorrettamente e possibili duplicati, ma questi sono presenti in un numero molto piccolo, quindi si aspettano che non influiscano più di tanto sui risultati del modello.

Dopo che tutti i TCE sono stati etichettati, Yu et al. hanno binarizzato le etichette come "*planet-like*" e "*not planet-like*". Quando è usato il modello per il *triage* i segnali etichettati come EB e PC vengono considerati come *Planet-like*, in modo da poter conservare più candidati possibili, mentre quando è utilizzato il modello per il *vetting*, solo i PC sono considerati come *planet-like*.

In totale hanno utilizzato 16.516 TCE per il *triage*, mentre per il *vetting* sono stati rimossi 65 TCE poiché avevano un numero insufficiente di punti per costruire la vista dell'eclissi secondaria, per un totale di 16.451 TCE. Il dataset è stato mescolato randomicamente e partitionato in tre sottoinsiemi: 80% come *training-set*, 10% come *validation-set*, usato per scegliere i parametri del modello e il restante 10% come *test-set*, usato per valutare le performance del modello finale [13].

Per poter dare in input alla rete le curve di luce, Yu et al. hanno seguito quanto fatto da Shallue and Vanderburg: le curve di luce sono state ripiegate rispetto al periodo del transito individuato dal BLS, in modo tale che i transiti siano allineati e centrati. I dati sono stati poi intervallati per creare due viste:

- *global view*: che mostra la curva di luce su un intero periodo orbitale
- *local view*: che mostra il transito più da vicino

A differenza delle curve di Kepler, che contengono circa 70.000 punti, le curve di TESS ne contengono molte meno e quindi risultano essere molto più rade, per questo motivo Yu et al. hanno usato un numero di intervalli per la vista globale pari a 201, 61 per la vista locale.

Per il modello che si occupa di *vetting* è stata preparata anche una *secondary eclipse view*, secondo quanto indicato da Shallue and Vanderburg come possibile miglioria ad Astronet (Capitolo 3.3). Per prima cosa sono state cercate le probabili (o le più probabili) eclissi secondarie, mascherando il transito nella curva di luce ripiegata e usando un algoritmo simile al BLS "*to fit a box*", la cui larghezza è fissata a quella del primo transito, a varie posizioni tra fasi orbitali 0,1 e 0,9 nella curva di luce mascherata e ripiegata. La posizione che produce il rapporto S/N più alto viene considerato come il punto centrale della più probabile eclissi secondaria. Successivamente, le curve di luce sono state normalizzate e ripiegate entro la durata dei due transiti su entrambi i lati di questo punto in 61 intervalli, usando la stessa

procedura usata per produrre le *local view*.

Il modello usato per il *vetting*, ovvero per la classificazione, deve essere capace di distinguere i TCE etichettati come EB da TCE etichettati come PC: in input vengono passati sia la *global view* che la *local view* con - in aggiunta rispetto al modello usato per il *triage* - un primo piano della eclisse secondaria. Inoltre, sono state aggiunte delle features scalari all'output degli strati convoluzionali, ovvero la differenza nella profondità dei transiti ottenute in due aperture diverse divise per la deviazione standard misurata nell'apertura minore. La motivazione nell'aggiunta di queste caratteristiche è che possano aiutare il modello a identificare meglio potenziali miscugli. Per Yu et al. queste features sono semplici alternative ai centroidi usati da Ansdell et al.. Infine, è stato scelto di non inserire parametri della stella perché la maggior parte dei TCE non ne aveva disponibili.

L'architettura di *Astronet-Vetting* è mostrata in Figura 3.11.

Entrambe le reti usano l'algoritmo di ottimizzazione Adam per minimizzare la *cross-entropy function* sul *training set*. Anche Yu et al. hanno utilizzato una tecnica di *data augmentation* sull'insieme di addestramento: hanno applicato una riflessione orizzontale casuale alle curve di luce con una probabilità del 50%, ottenendo così nuovi dati simili a quelli iniziali con la stessa etichetta e di conseguenza aumentando la dimensione del *training set*.

Yu et al. hanno scelto la dimensione delle batch pari a 64, con un tasso d'apprendimento di  $\alpha = 10^{-5}$ ,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$  ed  $\epsilon = 10^{-8}$ . Anche Yu et al. hanno utilizzato la tecnica del *model averaging*.

Le metriche di valutazione usate da Yu et al. sono le stesse utilizzate da Shallue and Vanderburg. In Tabella 3.4 sono riportati i risultati per le reti senza le due nuove features, le reti con le due features inserite separatamente e le reti dove le features sono inserite entrambe.

In Figura 3.10 sono raffigurate le curve di *Precision vs. Recall* per i modelli *Astronet-Triage* e *Astronet-Vetting*.

Dai risultati ottenuti, Yu et al. osservano che *Astronet-Vetting* non è pronto ad essere adoperato. Utilizzando come valore soglia 0,5, il modello considera la maggior parte dei TCE come non pianeti: solo 28 dei 49 dei PC nel *test set*, con una precisione di 0.651. Sapendo che TESS è una missione progettata per avere successori, Yu et al. vogliono recuperare quanti più PC possibile al costo di più falsi positivi, che potranno facilmente essere rimossi dai programmi successivi. Per questo motivo Yu et al. hanno deciso di valutare le prestazioni del modello con il valore soglia di 0,1. In questo modo sono stati recuperati 44 dei 49 PC, con precisione dello 0,449. L'incapacità del modello di separare EB da PC può essere dovuta al dataset sbilanciato e Yu et al. osservano che una aggiunta di PC può migliorare le *performance*

	Accuracy	AUC	Average Precision
Astronet-Vetting (No feature)	0.977	0.973	0.605
Astronet-Vetting + differenza dei transiti	0.978	0.980	0.669
Astronet-Vetting + eclissi secondaria	0.976	0.978	0.642
Astronet-Vetting + differenza dei transiti + eclissi secondaria	0.978	0.984	0.693

**Tabella 3.4:** I risultati ottenuti sul *test set* dalle diverse tipologie di reti.

del modello.

Anche in questo caso, Yu et al. osservano che il modello può essere utilizzato in assistenza al lavoro umano, rendendo uniforme le regole utilizzate per identificare i pianeti candidati.

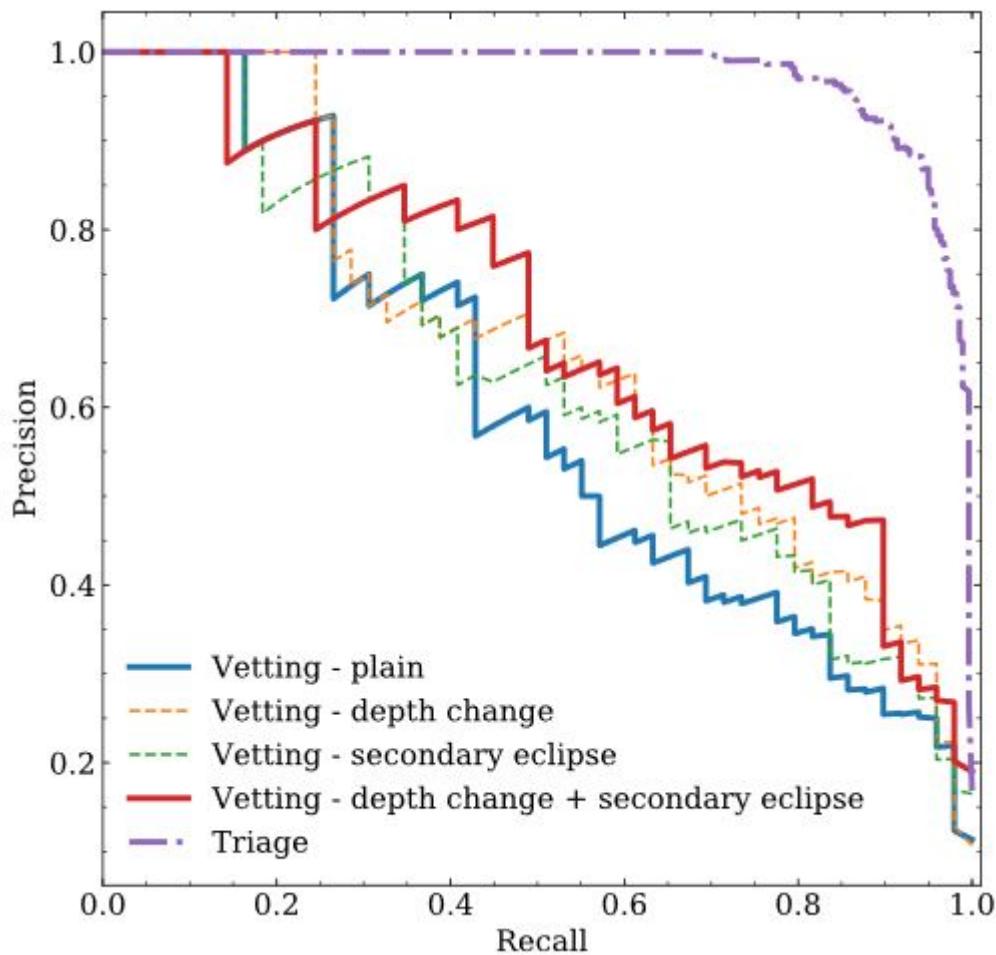
### 3.7 Genesis

Genesis [15] è una CNN utilizzata per l'identificazione degli esopianeti. A differenza di Astronet e le sue versioni successive, Genesis ha un unico input. La motivazione per cui Visser et al. hanno utilizzato un solo input è dovuta da un punto di vista informatico piuttosto che astronomico:

1. Utilizzare le due viste globale e locale introduce ridondanza di dati, poiché la vista locale è contenuta in quella globale.
2. Utilizzare due input fa aumentare il costo computazionale.
3. Utilizzare le due viste aumenta le possibilità di *overfitting* e riduce quelle di generalizzazione.

Visser et al. hanno condotto tre esperimenti: nel primo esperimento hanno paragonato le prestazioni di Genesis con quelle di Astronet; nel secondo hanno valutato le prestazioni se si randomizzano i dati utilizzando la validazione incrociata di Monte Carlo; infine hanno valutato come allargare le curve di luce va a influenzare le performance. [15].

Per condurre questi esperimenti Visser et al. hanno utilizzato due dataset: il dataset utilizzato

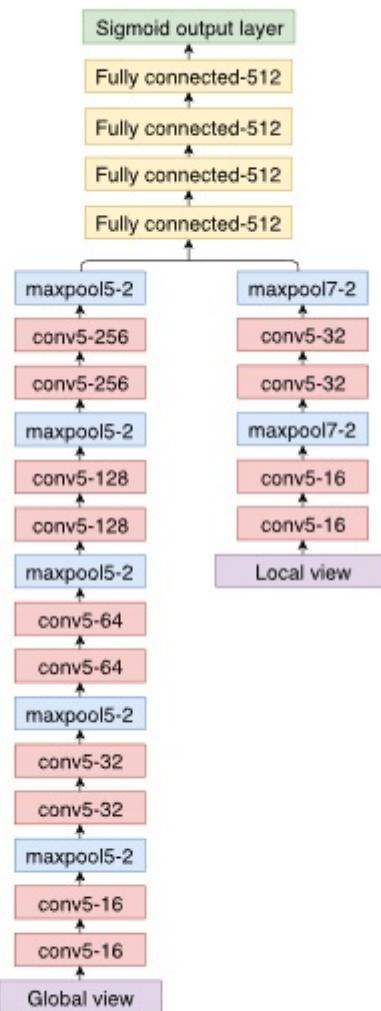


**Figura 3.10:** Le curve del *Precision vs Recall* calcolate sui vari modelli: *Vetting-plain* è il modello Astronet. Fonte: "Identifying exoplanets with deep learning. III. Automated triage and vetting of TESS candidates"

da Shallue and Vanderburg per addestrare Astronet (la creazione di questo dataset è stata descritta nel Capitolo 3.5) e una versione generalizzata di quest'ultimo creato da Visser et al.. Gli autori hanno voluto creare questo dataset per due motivi:

1. Creare curve di luce ripiegate sulla fase che hanno il doppio di intervalli rispetto a quelle nel dataset di Astronet.
2. Poter suddividere il dataset in training-set, validation-set e test-set secondo la procedura della validazione incrociata Monte Carlo.

Per costruire il dataset generalizzato di Astronet, Visser et al. hanno utilizzato la stessa procedura di Shallue and Vanderburg, implementando la validazione incrociata di Monte Carlo andando a partizionare il dataset in 20 partizioni casuali di training, validazione e test e preparando i dati per permettere una dimensione degli intervalli maggiori. Per la



**Figura 3.11:** L'architettura di *Astronet-Vetting*, fonte: "Identifying exoplanets with deep learning. III. Automated triage and vetting of TESS candidates".

Gli strati convoluzionali sono indicati come `conv<kernel size>-<number of feature map>`, gli strati di max pooling sono indicati come `maxpool<window lenght>-<stride lenght>` e gli strati completamente connessi sono indicati come `FC-<number of units>`.

rimozione degli *outliers* e l’appiattimento della curva di luce, gli autori hanno utilizzato i metodi `remove_outliers()` e `flatten()` della libreria del linguaggio di programmazione Python, `Lightkurve` [16]. Utilizzando il metodo `fold()` infine hanno ripiegato le curve di luce nella dimensione degli intervalli desiderati, ottenendo due tipi di viste globali, di intervalli di dimensione 2.001 e 4.002, per un totale di 14.660 e 14.009 curve di luce rispettivamente. [15]. L’architettura della rete è visibile in Figura 3.12.

Anche Visser et al. hanno utilizzato tecniche di *data augmentation* utilizzando la riflessione orizzontale usata da Shallue and Vanderburg e creando quattro versioni delle curve di luce con rumore Gaussiano: il rumore è ottenuto da una distribuzione normale con media e deviazione standard uguale a quella delle curve di luce ripiegate nei training set [15].

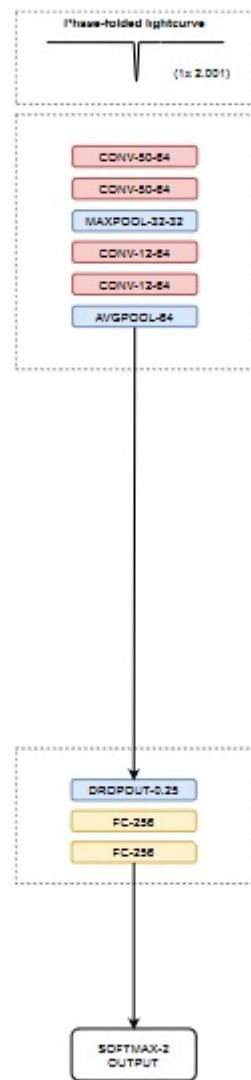
Visser et al., come Shallue and Vanderburg hanno utilizzato la tecnica del *model averaging*, creando 60 insiemi di 10 istanze di Genesis. 20 di questi insiemi sono stati addestrati col dataset di Astronet, 20 con il dataset generalizzato da 2.001 e 4.001.

Hanno utilizzato la uniforme di Xavier per inizializzare i filtri convoluzionali e i pesi degli strati completamente connessi e hanno implementato l’*early stopping*. Come funzione di attivazione è stata usata la funzione ReLU e la funzione da minimizzare utilizzata è stata la *categorical cross entropy*. Per ogni insieme è stata usata come metrica di valutazione l’*ensemble accuracy*, che calcola la media aritmetica su tutte le predizioni fatte dai 10 modelli addestrati sul test-set nell’insieme. Inoltre, è stata calcolata anche la *ensemble AUC*. La Tabella 3.5 mostra i risultati ottenuti. Sulla base di questi Visser et al. hanno fatto tre osservazioni:

1. Genesis ottiene una *accuracy* minore dello 0,5% rispetto ad Astronet.
2. Genesis ottiene prestazioni migliori rispetto ad Astronet<sub>global</sub>.
3. Rispetto a tutte e tre le versioni di Astronet, Genesis ottiene valori di *ensemble AUC* molto inferiori. Questo è dovuto al fatto che per Astronet, Shallue and Vanderburg hanno utilizzato *AUC* come metrica di ottimizzazione, mentre per Genesis si è utilizzata la *accuracy*.

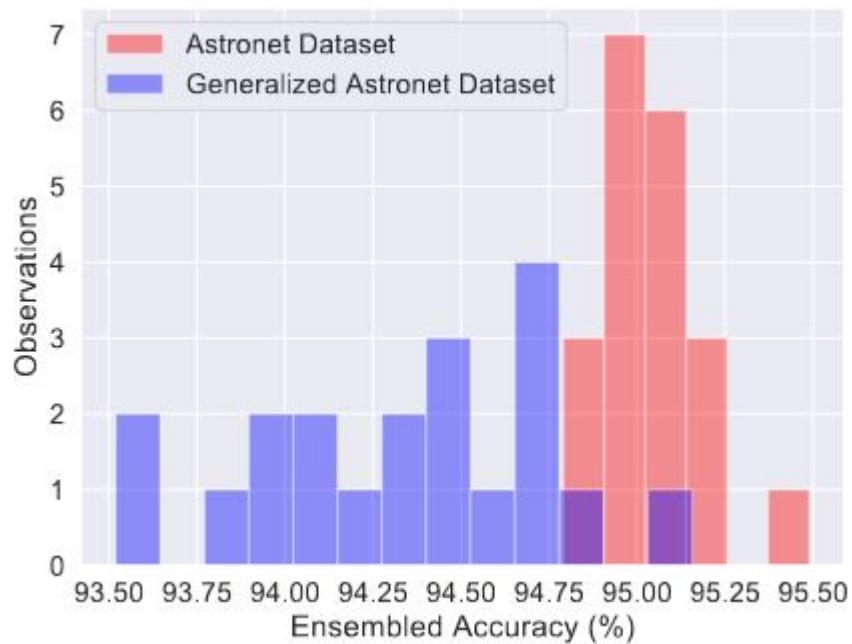
I risultati dell’addestramento che hanno utilizzato i dati randomizzati con la validazione incrociata di Monte Carlo sono illustrati in Figura 3.13a. L’istogramma rappresenta la distribuzione delle *ensambled accuracy* ottenute con e senza randomizzazione (rispettivamente in blu e rosso).

Mentre i risultati relativi all’allargamento delle curve di luce sono illustrati in Figura 3.13b. I risultati ottenuti dimostrano che usare più intervalli non porta a *ensambled accuracies*

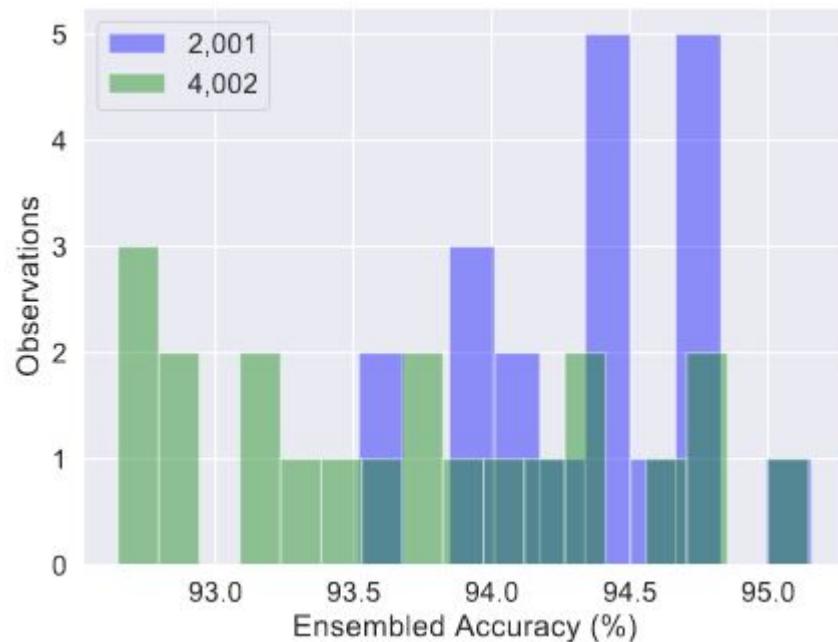


**Figura 3.12:** L'architettura di *Genesis*, fonte: “A one-armed CNN for exoplanet detection from light curves”.

CONV-50-64 indica uno strato convoluzionale di dimensione 50 con un numero di filtri di 64. MAXPOOL-32-32 indica uno strato di max pooling di lunghezza 32 e stride di 32. AVGPOOL-64 indica l'average pooling su 49.216 input. DROPOUT-0.25 indica uno strato di dropout con probabilità del 25%. FC-256 indica uno strato completamente connesso con 256 neuroni.



(a) Istogramma delle *ensambled accuracies* ottenute da Genesis su 100 repliche sul dataset di Astronet (rosso) e il dataset generalizzato di Astronet con 2.001 intervalli (blu).



(b) Istogramma delle *ensambled accuracies* ottenute da Genesis su 100 repliche del dataset generalizzato di Astronet con 2.001 (blu) e 4.002 (verde) intervalli.

**Figura 3.13:** Istogrammi relativi alle ensambled accuracies ottenute da Genesis. Fonte immagini: "A one-armed CNN for exoplanet detection from light curves"

Architettura	Dimensioni di input	Ensamble	Ensamble
		Accuracy	AUC
Genesis	2.001	95.5%	94.5%
Astronet	2.001 e 201	96.0%	98.8%
Astronet global	2.001	95.4%	98.5%
Astronet loca	201	92.4%	97.3%

**Tabella 3.5:** Paragone dei risultati tra Genesis e Astronet

migliori, ma porta ad aumentare la varianza.

Concludendo, Visser et al. osservano come una riduzione del 95% dei parametri fa differire di pochissimo le predizioni ottenute e come la convalida incrociata di Monte Carlo fornisce una stima leggermente migliore del potere predittivo . Utilizzare reti poco profonde può quindi portare a una maggiore generalizzazione e che esplorare questa tipologia di reti può giovare nella generalizzazione nell’identificazione di esopianeti tra le varie *survey*.

### 3.8 Identificazione di esopianeti utilizzando dati reali e artificiali

Lo studio condotto da Cuéllar et al. si concentra sullo sviluppare un modello capace di identificare transiti nelle curve di luce di Kepler. In questo caso, la CNN è stata addestrata usando sia curve di luci reali che sintetiche da loro generate.

I dati delle curve di luce reali sono stati ricavati dalla tabella dei KOI (*Kepler Objects of Interest*) del Nasa Exoplanet Archive. Cuéllar et al. hanno ottenuto le etichette dalla colonna *koi\_disposition*, che contiene 2358 esopianeti confermati, 2366 esopianeti candidati e 4840 falsi positivi<sup>1</sup>, scartando quest’ultimi. I dati etichettati che non contengono transiti sono stati ottenuti dal *Kepler Data Release 25-Q1* (DR25). Gli autori hanno inoltre deciso di utilizzare solo dati di sistemi dove un solo esopianeta o esopianeta candidato è stato individuato [17]. L’idea di base è di utilizzare curve di luce che hanno abbastanza campioni per rappresentare 10 periodi, considerando che maggiore è il periodo, maggiore sarà il numero di campioni. Nel caso di Kepler, le curve hanno 4230 campioni ottenuti ogni 30 minuti, quindi il periodo massimo per ricoprire i 10 segmenti sarà 9 giorni. Di conseguenza, Cuéllar et al. hanno considerato tutti i transiti con un periodo tra 0,85 e 8,5 giorni, per un totale di 583 curve di luce provenienti dal primo quarto. Per mantenere bilanciato il dataset, sono state scelte lo

<sup>1</sup>Rispetto alla data di pubblicazione dell’articolo questi dati sono stati aggiornati in quanto sono stati identificati nuovi esopianeti.

stesso numero di curve di luce senza transito. [17].

Nel dataset sono state incluse curve di luce artificiali con presenza di transito. Queste curve di luce sono state generate usando un modello introdotto da Mandel and Agol.

Dopo aver creato il dataset, Cuéllar et al. hanno analizzato le curve di luce per rimuovere intervalli vuoti, che sono stati sostituiti con il valore medio dei vicini e, per avere consistenza con la realtà, a questi nuovi valori è stato aggiunto rumore con il 10% di potenza [17]. Le curve di luce sono state poi interpolate su 4000 punti ed infine i valori sono stati normalizzati tra 0 e 1.

Le curve di luce sono state poi ripiegate rispetto al periodo in dieci segmenti. Il valore del periodo è stato ricavato dalla colonna `koi_period` della tabella KOI e infine i valori di ogni periodo sono stati incorporati come righe di un'immagine. Con questo procedimento si creano rappresentazioni a due dimensioni, poiché si ripiegano le curve di luce su un periodo che può essere diverso dal periodo del transito, migliorando l'identificazione del transito indipendentemente sia dal periodo del transito, sia dal *folding period* Cuéllar et al..

Cuéllar et al. hanno utilizzato il *transfer learning* per creare il modello della loro rete neurale. Il *transfer learning* è una tecnica che permette di utilizzare una rete neurale già addestrata che viene utilizzata per risolvere un nuovo problema. La rete di partenza è la rete Xception sviluppata da Francois Chollet: questa rete può classificare immagini in 1000 categorie. Per permettere alla rete di poter trattare il problema preso in considerazione, Cuéllar et al. hanno ridimensionato l'input e ridotto il numero di neuroni di output: le immagini di input sono monocromatiche e hanno dimensione 400x10 e, trattandosi di un problema di classificazione binario, lo strato di output contiene un solo neurone con funzione di ottimizzazione sigmoidea. L'output della rete dipende dal valore soglia scelto.

Per vedere l'effetto dell'aumento dei dati sintetici sull'addestramento, Cuéllar et al. hanno costruito i modelli con  $R = 483$  curve reali e un rapporto di curve sintetiche  $S$  con transito definite dal parametro  $\lambda$  utilizzato nella seguente formula:

$$S = \frac{\lambda R}{100 - \lambda} \quad (3.8.1)$$

Inoltre, lo stesso numero di curve senza transito ( $S + R$ ) sono aggiunte al training set per mantenerlo bilanciato. Come test set è stato scelto un numero pari a 100 di curve reali con transito e senza transito.

Il test set è formato da  $R = 100$  curve di luce reali con transito e senza transito. Per valutare le performance del modello, Cuéllar et al. hanno deciso di utilizzare le seguenti metriche:

- Accuracy

- Precision
- Rapporto dei veri positivi (*True Positive Rate*, TPR)
- Rapporto dei falsi positivi (*False Positive Rate*, FPR)
- $F_1$ -Score

Le prime tre metriche dipendono dal valore soglia scelto dal classificatore. Nella loro ricerca Cuéllar et al. vogliono mostrare che le prestazioni del modello possono migliorare utilizzando una proporzione  $\lambda$  di curve di luce sintetiche durante l'addestramento. Per identificare il miglior rapporto di curve di luce sintetiche, il modello è stato addestrato su due diversi scenari:

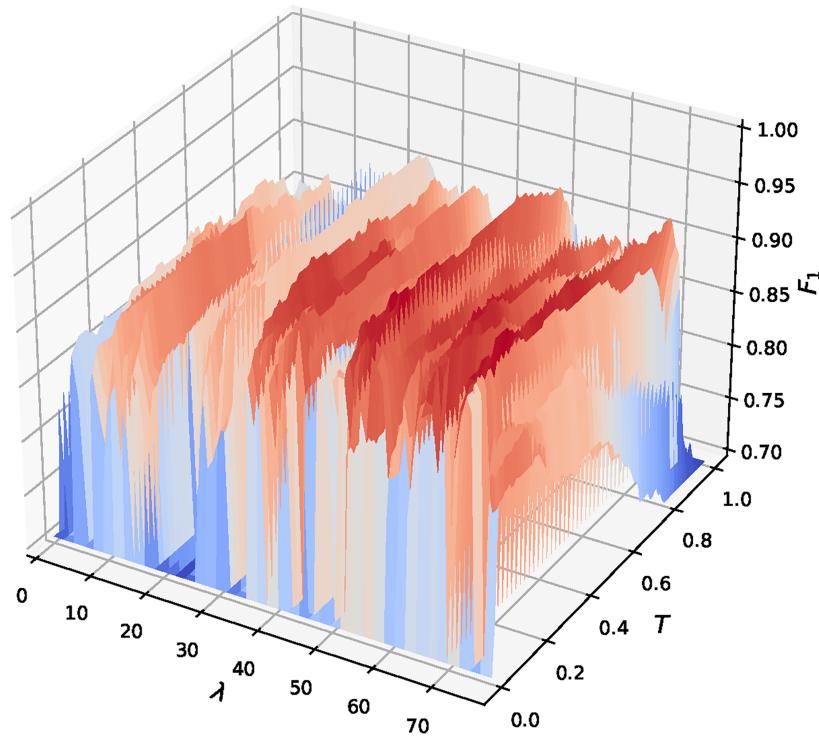
- Scenario 1: il modello è stato addestrato solo con 483 curve di luce con transito reali ( $S = 0, R = 483$ ) e  $S + R$  curve di luce reali senza transito.
- Scenario 2: il modello è stato addestrato solo con 483 curve di luce sintetiche con transito ( $S = 483, R = 0$ ) e  $S + R$  curve di luce reali senza transito.

Per lo scenario 1, il modello ottiene una precisione del 74,7% sul test set. Il modello addestrato sullo scenario 2 ottiene una precisione molto simile (75%), ma non ha tutta la variabilità delle curve con transito. Quindi per Cuéllar et al. addestrare il modello con curve di luce reali fornisce al modello variabilità sul transito e unire le curve di luce sintetiche può migliorare la precisione della predizione [17].

Per identificare i valori migliori di  $\lambda$  e  $T$  (il valore soglia), Cuéllar et al. hanno condotto una ricerca euristica e una analisi della sensitività.

Per quanto riguarda la ricerca euristica, Cuéllar et al. hanno cercato il valore massimo del punteggio  $F_1$  utilizzando  $\lambda$  e  $T$  come variabili del problema. Tale problema è stato risolto utilizzando un algoritmo genetico. Gli algoritmi genetici sono un metodo molto utilizzato per risolvere problemi di ottimizzazione, sfruttando un processo che riflette la selezione naturale: iterativamente, l'algoritmo modificherà una popolazione di soluzioni, conservando solo le migliori dell'iterazione e utilizzandole per generare nuove soluzioni da utilizzare nell'iterazione successiva. I risultati dell'esperimento sono riportati nella Tabella 3.6: la prima colonna rappresenta i valori della dimensione della popolazione; la seconda rappresenta il numero di iterazioni (quindi di generazioni) dell'algoritmo; la terza contiene il numero di curve sintetiche con transito usate durante l'addestramento; la quarta mostra il valore soglia; la quinta colonna contiene i valori della metrica  $F_1$ ; la sesta contiene il numero di modelli

addestrati per ottenere il valore  $F_1$  [17]. Il miglior valore  $F_1$  (0,9801) è stato ottenuto in tre differenti configurazioni (in grassetto in tabella). Tutte e tre hanno lo stesso valore per  $S$  e questo valore corrisponde al  $\lambda = 74,3\%$  di curve di luce sintetiche e al 27,7% di curve di luce reali con transito nel training set. Il valore di  $T$  è compreso tra 0,21 e 0,23. La configurazione scelta è stata quella composta da una popolazione di 10, 50 generazioni, perché addestra il numero minore di modelli per ottenere gli stessi risultati con un valore di  $T$  più alto [17]. L'analisi della sensitività è stata svolta andando ad analizzare la dipendenza del punteggio  $F_1$  dai valori di  $\lambda$  e  $T$ , con il primo che può variare tra lo 0 e l'80%, facendo incrementare  $S$  da 0 a 1932 a passi di 23.  $\lambda$  diventa importante per valori tra il 60 e l'80%. I valori di  $T$  invece sono compresi tra 0 e 1, considerando numeri a due decimali. In Figura 3.14 si può osservare il grafico tridimensionale che confronta i valori di  $F_1$  rispetto a  $\lambda$  e  $T$ : si può osservare che per valori maggiori del 50% per  $\lambda$  il valore di  $F_1$  è maggiore, il che prova l'ipotesi che incrementare il numero di curve sintetiche migliora le performance. Il valore maggiore viene ottenuto per valori di  $\lambda$  vicini al 74% e  $T$  vicini allo 0,20, risultati simili a quelli ottenuti tramite l'utilizzo dell'algoritmo genetico.



**Figura 3.14:** Grafico 3D dei valori di  $F_1$  rispetto a  $\lambda$  e  $T$ . Fonte: "Deep learning exoplanets detection by combining real and synthetic data".

Per ottenere una visione maggiore dell'effetto del valore soglia  $T$ , Cuéllar et al. hanno addestrato i modelli facendo variare il valore di  $\lambda$  tra lo 0 e l'80%, incrementandolo del

5% di volta in volta. Per ognuno di questi valori, è stato fatto variare il valore di  $T$  e sono stati calcolati il TPR e il FPR per costruire le curve ROC. In questo modo Cuéllar et al. hanno osservato che incrementare  $T$  porta a un incremento dei veri positivi andando però a classificare erroneamente le istanze negative. Quindi bisogna trovare il migliore valore di  $T$  che si avvicina al caso ideale ( $FPR = 0$ ,  $TPR = 1$ ). La curva con  $\lambda = 70\%$  ha il punto  $(0,050; 0,970)$ , che è il più vicino a quello ideale [17]. Il grafico con tutte le curve ROC è visibile in Figura 3.15.

Dimensione della Popolazione	Numero di Generazioni				
		S	T	F1	F1 calc
10	5	1633	0,09	0,9607	80
	10	1173	0,07	0,9371	160
	15	1334	0,41	0,9591	240
	20	759	0,28	0,9607	320
	<b>50</b>	<b>1403</b>	<b>0,23</b>	<b>0,9801</b>	<b>800</b>
20	5	1794	0,10	0,9560	180
	10	1403	0,28	0,9751	380
	15	1403	0,19	0,9753	540
	<b>20</b>	<b>1403</b>	<b>0,22</b>	<b>0,9801</b>	<b>820</b>
	50	1794	0,1	0,9560	1800
50	5	1334	0,23	0,9651	440
	<b>10</b>	<b>1403</b>	<b>0,21</b>	<b>0,9801</b>	<b>880</b>
	15	1403	0,19	0,9753	1320
	20	1403	0,18	0,9705	1760
	50	1403	0,2	0,9753	4400

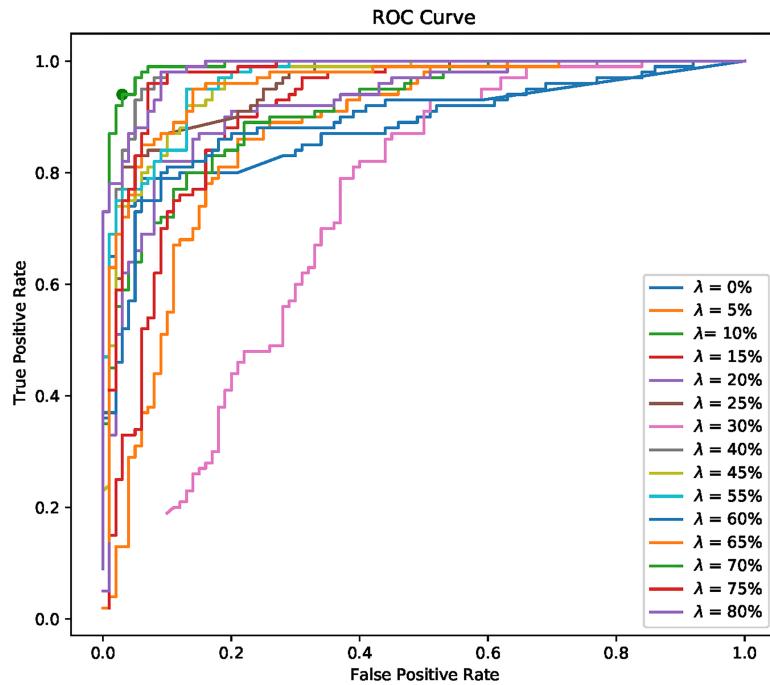
**Tabella 3.6:** I risultati dell’algoritmo genetico con diversi parametri.

Cuéllar et al. osservano che rispetto allo scenario 1, dove  $\lambda = 0$ , l’aumento del solo il valore soglia  $T$  da 0,5 a 0,608 fa aumentare la precisione da 0,747 a 0,923 e di conseguenza anche il valore F<sub>1</sub> aumenta da 0,743 a 0,808.

Il valore del TPR incomincia a salire da  $\lambda = 25\%$ , quindi per Cuéllar et al. aggiungere curve di luce sintetiche aggiunge conoscenza e permette al modello di identificare curve reali con transiti. I valori della precisione incominciano ad aumentare da  $\lambda = 60\%$ , quindi [17] osservano che aggiungere curve di luce al modello incrementa la precisione, ma allo stesso

tempo genera *bias* nel modello poiché diventa difficile identificare vere curve di luce con transito, per cui è necessario decrementare il valore di  $T$  che separa le due classi.

Il miglior modello è stato ottenuto con  $\lambda = 74,3\%$  e  $T = 0,23$ . Per valutare il modello sono state utilizzate  $R = 200$  curve di luce reali (100 con transito e le restanti senza transito), ottenendo una precisione dello 0,9705; un TPR dello 0,99; un punteggio  $F_1$  dello 0,9801; un FPR dello 0,03 e un *accuracy* dello 0,98 [17].



**Figura 3.15:** Curve ROC per i diversi valori di  $\lambda$ . Fonte = "Deep learning exoplanets detection by combining real and synthetic data".

# CAPITOLO 4

---

ExoHunter

---

NIENTE PANICO!

## 4.1 Formulazione del problema

Le ricerche presentate nel capitolo precedente hanno fornito una base molto solida nell'affrontare il problema dell'identificazione di transiti dai dati ottenuti dai telescopi spaziali. Per questo motivo, con questa tesi, si vuole presentare una variante del problema: la classificazione delle curve di luce di esopianeti, ovvero creare un modello basato sulle tecniche di Intelligenza Artificiale più usate, capace di distinguere tra curve di luce di esopianeti da curve ottenute da falsi positivi dovute a errori causati da fenomeni astrofisici e simili. Il motivo per cui si è scelta questa variazione del problema presentato è dovuta al fatto che, come visto nel Capitolo 3, dopo aver identificato un possibile transito in un TCE, questo viene comunque sottoposto ad ulteriori analisi per confermare la correttezza dell'identificazione. Quindi fornire un metodo automatizzato capace di distinguere se una curva di luce è o non è un esopianeta potrà essere d'aiuto.

Possiamo quindi formulare formalmente il problema: data una curva di luce, vogliamo che il modello sia capace di dirci se tale curva è di un esopianeta oppure si tratta di un falso positivo. Trattiamo il problema come un problema di classificazione binario: dato un input - la curva di luce in questo caso - il modello dovrà indicarci a quale delle due classi appartiene. In questo caso il modello restituirà il risultato sotto forma di probabilità: valori vicino ad 1

indicano che l'input è stato classificato come esopianeta, valori vicino a 0 indicano che l'input è stato classificato come falso positivo. Per questo motivo utilizzeremo un valore soglia per la classificazione: valori maggiori o uguali di 0,5 saranno classificati come esopianeti mentre valori inferiori a 0,5 saranno classificati come falsi positivi.

Il metodo di risoluzione scelto è quello delle reti neurali convoluzionali: come visto nel Capitolo 3, sono uno strumento molto potente la cui efficacia è stata già provata.

## 4.2 Creazione del dataset

Una volta formulato il problema da risolvere, il passo successivo è quello di ottenere i dati da utilizzare per poter addestrare la CNN. Si è scelto quindi di utilizzare i dati ottenuti dalle osservazioni del telescopio Kepler. I dati sono disponibili nel *Nasa Exoplanet Archive*<sup>1</sup>: un catalogo astronomico *online* che racchiude dati relativi agli esopianeti.

Poiché vogliamo creare una CNN capace di classificare e curve di luce di esopianeti, recupereremo i dati degli esopianeti identificati dalla missione Kepler. I dati sono stati recuperati dalla pagina dell'archivio relativa ai KOI (*Kepler Objects of Interest*, stelle osservate da Kepler nel cui sistema si sospetta siano presenti uno o più pianeti). La pagina si presenta sotto forma di tabella interattiva, come si può osservare in Figura 4.1), che contiene informazioni relative agli esopianeti confermati, falsi positivi e esopianeti candidati. Di questa tabella sono state selezionate le seguenti colonne:

- `kepid`: numero identificativo del target;
- `kepler_name`: nome del target nel formato "Kepler-N" più una lettera minuscola che rappresenta l'esopianeta;
- `koi_disposition`: la classificazione effettuata dall'archivio. I possibili valori sono:
  1. CANDIDATE (Candidato)
  2. FALSE POSITIVE (Falso positivo)
  3. NOT DISPOSITIONED (Non disposto)
  4. CONFIRMED (Confermato)
- `koi_period`: rappresenta l'intervallo tra due transiti planetari consecutivi, misurati in giorni;

---

<sup>1</sup><https://exoplanetarchive.ipac.caltech.edu/>

- `koi_time0bk`: rappresenta il tempo corrispondente al centro del primo transito identificato, misurato in *Barycentric Julian Day* (BJD), meno una costante di 2.454.833,0 giorni (questo *offset* corrisponde alle ore 12:00 del 1° Gennaio 2009 UTC).

The screenshot shows a table titled "Cumulative KOI Data" from the NASA Exoplanet Archive. The table has 15 columns and 9564 rows. The columns are: KeplerID, KOI Name, Kepler Name, Exoplanet Archive Disposition, Disposition Using Kepler Data, Disposition Score, Not Trans- Like False Positive Flag, Stellar Eclipse False Positive Flag, Centroid Offset False Positive Flag, Ephemeris Match Indicates Contamination False Positive Flag, Orbital Period [days], Transit Epoch [BJD], and Impact P. The data includes various Kepler objects like Kepler-227 b, Kepler-227 c, Kepler-64 b, etc., with their respective dispositions (CONFIRMED, CANDIDATE, FALSE POSITIVE) and orbital parameters.

**Figura 4.1:** La tabella relativa ai KOI del *Nasa Exoplanet Archive*

La tabella è stata scaricata sotto forma di file .csv (*comma separated value*). Utilizzando la libreria `pandas`, una libreria per il trattamento dei dati disponibile per il linguaggio Python, il file è stato diviso in due *dataframe*, uno contenente tutti i dati relativi agli esopianeti confermati (quindi tutti i dati che nella colonna `koi_disposition` hanno valore CONFIRMED) e l'altro relativo ai falsi positivi (tutti i dati che nella colonna `koi_disposition` hanno valore FALSE POSITIVE).

Il passo successivo è stato quello di recuperare le curve di luce dal MAST. Questa operazione è stata resa possibile grazie all'utilizzo di un'altra libreria chiamata chiamata `lightkurve` [16]: sviluppata dal *Kepler/K2 Guest Observer (GO) Office* per permettere di lavorare con i dati di Kepler e K2 e, recentemente, anche sui dati di TESS. Oltre a ciò, `lightkurve` fornisce anche metodi per poter operare sulle curve di luce.

Il procedimento seguito per ottenere le curve di luce è il seguente:

1. Tramite il metodo `search_lightcurve()` [18] sono stati cercati nel MAST i dati relativi target specificato: nel caso degli esopianeti confermati la ricerca è stata effettuata specificando il nome dell'esopianeta, mentre nel caso dei falsi positivi si indica l'identificativo. Inoltre, è stato specificato che si tratta della missione Kepler e che siamo

interessati ai dati con una risoluzione temporale di 30 minuti. Il risultato della ricerca è una tabella contenente i dati relativi alla curva di luce del target indicato per ogni quarto separato, che, tramite il metodo `download_all()` vengono tutti scaricati.

2. Tramite il metodo `stitch()` le diverse curve di luce sono state unite assieme per formare un'unica curva di luce.
3. Il passo successivo è la rimozione degli *outliers*, ovvero di quei valori che si distaccano troppo dagli altri. Questa operazione è stata effettuata tramite il metodo `remove_outliers()`: vengono rimossi i punti i cui valori sono inferiori di venti volte o di quattro volte superiori la deviazione standard. Questa nuova curva di luce ottenuta, tramite il metodo `flatten()`, viene appiattita.
4. Recuperiamo dal *dataframe* i valori del periodo e del tempo di transito.
5. Tramite il metodo `fold()`, ripieghiamo i punti rispetto al periodo e al tempo di transito ricavati nel passo precedente. Tramite il metodo `bin()` dividiamo la curva ripiegata in 2001 intervalli equispaziati e con il metodo `normalize()` normalizziamo i valori del flusso della curva.
6. L'ultimo passaggio consiste nel salvare l'array contenente i valori del flusso.

Questi passaggi sono ovviamente effettuati per tutti gli elementi presenti nei due differenti *dataframe* tramite due diverse funzioni. Da questo momento in poi per curve di luce si intendono gli array contenenti i valori del flusso delle curve di luce.

Nelle figure 4.2 e 4.3 possiamo osservare la rappresentazione grafica delle curve di luce: la prima rappresenta la curva di luce relativa all'esopianeta *Kepler-53 d*, la seconda rappresenta la curva di luce di un falso positivo.

Il totale abbiamo 2663 curve di luce relative a esopianeti confermati, mentre 4762 curve di luce relative ai falsi positivi. Ovviamente, in questo momento, il dataset è sbilanciato: il numero di campioni nella classe dei falsi positivi è maggiore rispetto a quella dei pianeti confermati. Prima di effettuare il bilanciamento del dataset, bisogna controllare se tra i valori del flusso sono presenti valori *NaN* (*Not-a-Number*): la presenza di *NaN* va a intaccare le prestazioni dei modelli di IA, per cui vanno rimossi dal dataset. Nel nostro caso i valori *NaN* sono stati sostituiti con il valore medio del flusso.

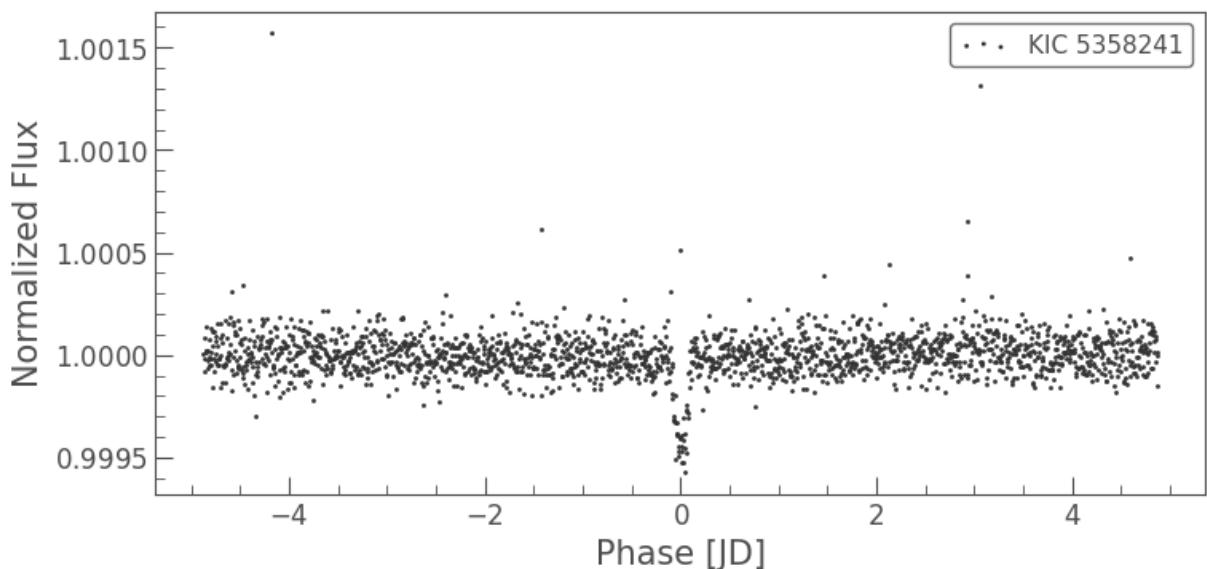
Possiamo ora bilanciare il dataset. Bisogna quindi aumentare il numero di valori del flusso di esopianeti confermati. Per fare ciò, sfruttiamo la seguente idea: possiamo invertire l'array

contenente i valori del flusso, ottenendo così nuove curve di luce speculari di quelle originali. Ovviamente non effettuiamo questa operazione per tutte le curve di luce di esopianeti confermati, altrimenti il dataset sarebbe ancora sbilanciato. Invertiamo quindi solo i primi 2200 array. Ora il dataset è bilanciato: ci sono 4863 curve di luce di esopianeti confermati e 4762 curve di luce di falsi positivi, per un totale di 9625 curve di luce totali.

Tipicamente, però, i dataset per problemi di Deep Learning sono molto più grandi: in questo modo il modello ha molti più campioni per poter apprendere. Per questo motivo, andiamo ad ampliare il dataset creando delle curve di luce artificiali a partire da quelle reali. Il procedimento di creazione delle curve di luce sintetiche è molto semplice: creiamo un vettore di numeri uniformemente distribuiti nell'intervallo (-0,0001; 0,0001) e sommiamo questi numeri ai valori del flusso della curva di luce. In questo modo, avremo una nuova curva di luce, simile nella forma a quella originale ma con valori diversi.

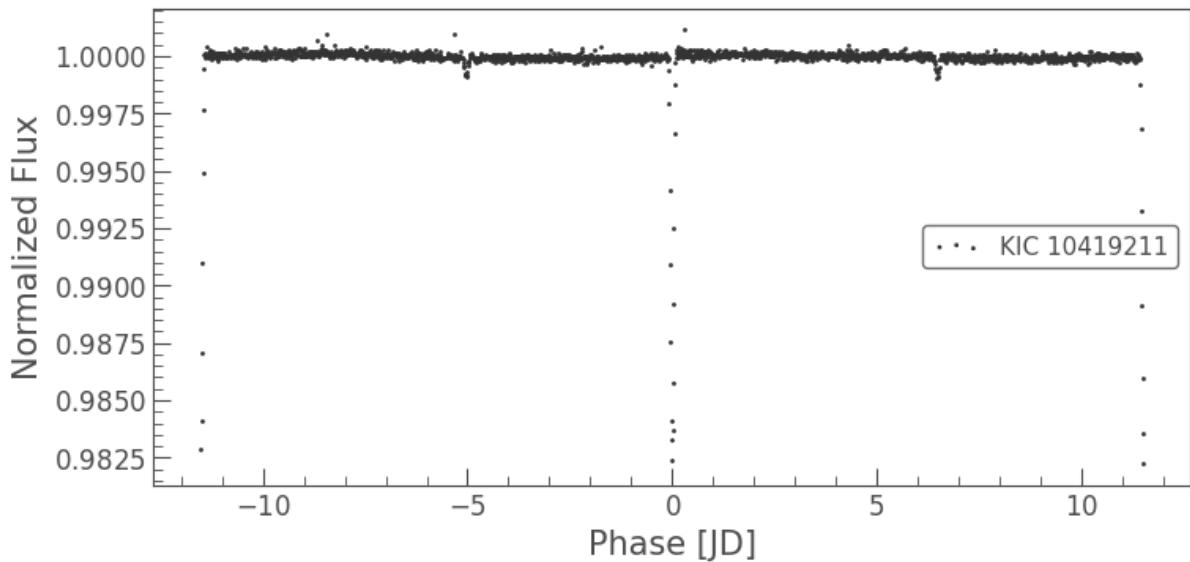
In Figura 4.4 si può osservare il confronto tra la curva di luce originale e quella artificiale. In questo modo raddoppiamo il numero di campioni nel dataset, arrivando a un totale di 19.250.

Passiamo ora a etichettare il dataset. Poiché stiamo trattando un problema di classificazione binario, le etichette saranno 0 e 1: 0 per i falsi positivi e 1 per i pianeti confermati.



**Figura 4.2:** Curva di luce di *Kepler-53 d*

Il passo successivo è la normalizzazione dei valori del flusso effettuata usando il metodo `minmax_scale()` della libreria `sklearn`.



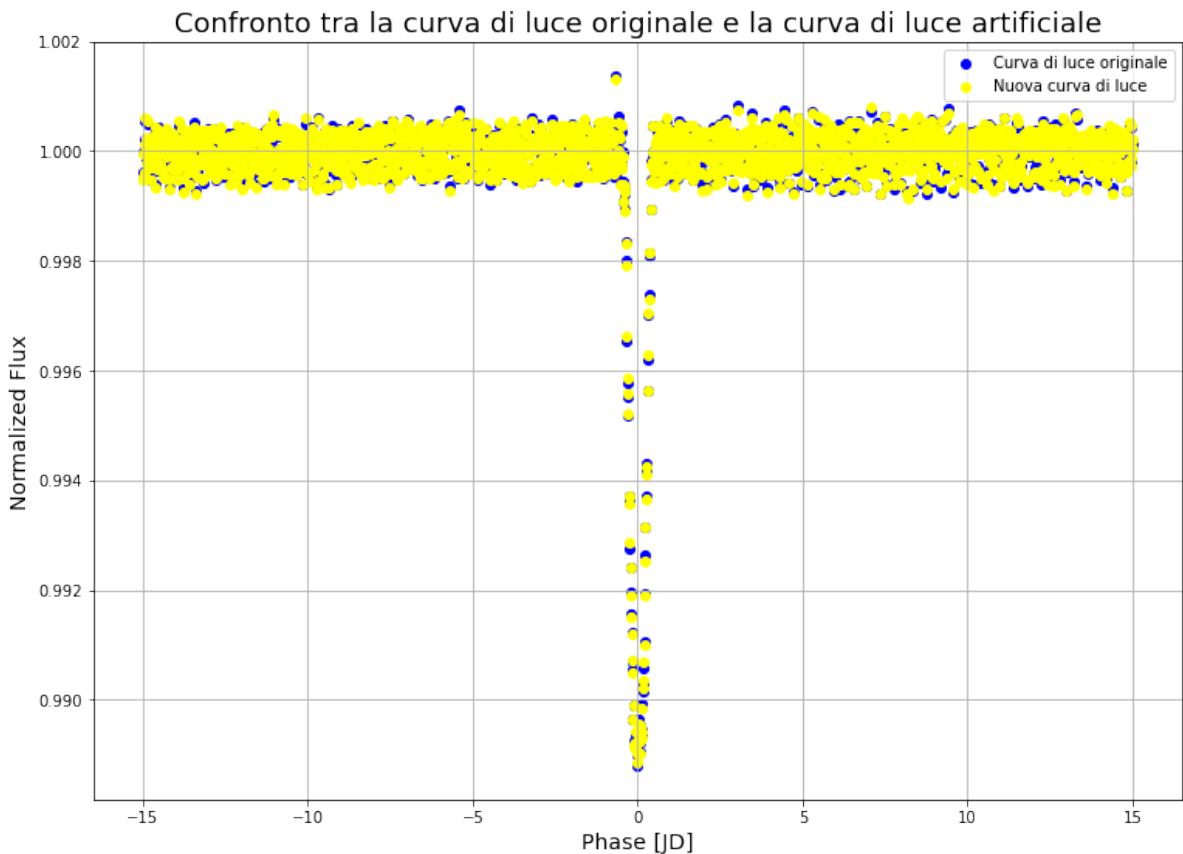
**Figura 4.3:** Esempio di curva di luce di falso positivo

L’ultima operazione del processo di creazione del dataset è la creazione degli insiemi di training, validazione e test. Per realizzarli, utilizziamo un’altra funzione di `sklearn` chiamata `train_test_split()` che crea questi tre insiemi in modo automatico secondo un rapporto specificato. Nel nostro caso, l’80% del dataset sarà utilizzato per l’addestramento, il 10% per la validazione e il restante 10% come test set. Il training set si compone di 15592 campioni, il validation set conta 1733 campioni, il test set 1925 campioni.

### 4.3 ExoHunter

L’approccio scelto per classificare le curve di luce prodotte, seguendo anche quanto esposto nel capitolo relativo allo stato dell’arte, è stato quello di utilizzare le reti neurali convoluzionali (CNN).

La è stata implementata usando Tensorflow [9], una libreria di Python per il Deep Learning. Inizialmente, si sono valutate le prestazioni dell’architettura Astronet - nello specifico la parte che riceve in input la vista globale - sul dataset creato. Le prestazioni delle reti si sono rivelate molto scarse, con un *accuracy* media intorno al 60%. Successivamente il modello è stato modificato - quindi sono stati aggiunti e/o rimossi strati convoluzionali e/o completamente connessi, cambiati gli iperparametri della rete (numero di filtri, dimensione dei kernel, ampiezza del passo, numero di neuroni, tasso d’apprendimento, funzioni di attivazione) - arrivando a testare una grande varietà di diverse reti. Addestrare tutte queste reti ha richiesto molto tempo e nessuna di esse raggiungeva risultati soddisfacenti: la migliore ha raggiunto



**Figura 4.4:** Confronto tra la curva di luce originale (blu) e la curva di luce artificiale (giallo)

un *accuracy* del 73,87%.

Questo approccio *trial and error* si è quindi rivelato molto deludente ed è stato scartato in favore di un approccio alternativo: per trovare la migliore architettura e i valori di tutti gli iperparametri si è ricorsi all'utilizzo di una libreria, sempre per il linguaggio Python chiamata Keras-Tunes [19], una libreria utilizzata per trovare i valori dei migliori iperparametri in una rete neurale.

Keras-Tunes mette a disposizione tre diversi algoritmi di ottimizzazione: l'algoritmo Random Search, l'algoritmo HyperBand e l'algoritmo Bayesian Optimization. Quest'ultimo è l'algoritmo che è stato utilizzato ed il motivo risiede nel modo di operare dell'algoritmo:

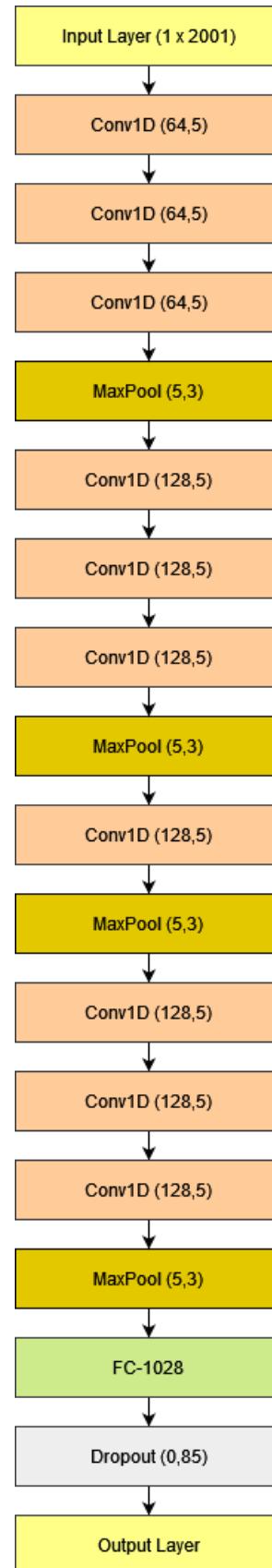
Questa libreria permette di operare una ricerca dei migliori iperparametri da utilizzare utilizzando tre tipologie di algoritmi: l'algoritmo Random Search, HyperBand e Bayesian Optimization. Si è scelto di utilizzare quest'ultimo per identificare gli iperparametri. Il Bayesian Optimization fa uso del teorema di Bayes per identificare il minimo (o massimo) globale di una funzione, puntando a costruire un modello probabilistico della

funzione obiettivo  $f(x)$  (ovvero la rappresentazione di probabilità della funzione obiettivo) che approssima  $f(x)$ . L'algoritmo sfrutta questo modello per determinare dove nel dominio della funzione estrarre un nuovo punto su cui valutare il modello probabilistico e aggiornarlo di conseguenza. Dopo un certo numero di iterazioni, troveremo il punto di minimo (o massimo) globale. L'idea fondamentale è quindi quella di utilizzare tutte le informazioni precedentemente ottenute dalle valutazioni del modello probabilistico per ottenere un nuovo modello più preciso [20].

L'architettura di Astronet è stata utilizzata anche in questo caso per definire l'algoritmo di creazione della nostra rete neurale: come si può osservare dal modello in Figura 3.5, possiamo suddividere Astronet in quattro blocchi, ognuno dei quali contiene due strati convoluzionali e uno strato di *max pooling*. Questi blocchi sono poi seguiti da strati completamente connessi. L'algoritmo di creazione della CNN è stato definito per poter ricreare questo schema: saranno decisi un determinato numero di blocchi, ognuno contenente strati convoluzionali seguiti da uno di *max pooling* e dopo i blocchi saranno aggiunti gli strati completamente connessi. Di seguito vengono riportati i valori degli iperparametri che l'algoritmo ha a disposizione :

- Numero di blocchi (valore massimo 5). Per ogni blocco:
  - Numero di strati convoluzionali (valore minimo 0, valore massimo 3). Per ogni strato convoluzionale:
    - \* Numero di filtri (valori tra cui scegliere: [8,16] per il primo blocco, [32,64] per il secondo, [64,128] per il terzo, quarto e quinto blocco)
    - \* Dimensione del kernel (valori tra cui scegliere: [3,5] per tutti gli strati)
  - Dimensione della *pool* (valori tra cui scegliere [2, 3, 5])
  - Ampiezza del passo (*pool stride*) (valori tra cui scegliere [1,2,3])
- Numero di strati completamente connessi (valore massimo 5). Per ogni strato:
  - Numero di neuroni (valori tra cui scegliere [32, 64, 128, 512, 1028])
  - Se inserire o meno lo strato di *dropout* dopo lo strato connesso (trattato come variabile booleana: *True* indica l'inserimento dello strato di *dropout*, viceversa *False*)
- Tasso di apprendimento (valori tra cui scegliere [0.001, 0.0001, 0.00001, 0.000001])
- *Rate* di *dropout* (valori compresi tra 0,30 e 0,85)

L'ampiezza del passo degli strati convoluzionali è stato fissato a 1 e quindi escluso dagli iperparametri da cercare: questo perchè poteva capitare che con determinate combinazioni



**Figura 4.5:** L'architettura di ExoHunter. Gli strati convoluzionali sono indicati come Conv1D (numero di filtri, dimensione del kernel), gli strati di *Maxpooling* sono indicati come MaxPool (dimensioni della pool, ampiezza del passo), gli strati completamente connessi come FC-numero di neuroni, lo strato di *dropout* come Dropout (tasso di *dropout*).

l'algoritmo non riuscisse a creare il modello. Come funzione di perdita è stata utilizzata la *binary crossentropy* mentre come algoritmo di ottimizzazione è stato utilizzato Adam.

L'algoritmo è stato istanziato perché provasse un massimo di 50 tentativi. Il modello migliore è raffigurato in Figura 4.5. Questo modello soffre però di *overfitting*, quindi il modello ‘impara a memoria’ invece di apprendere. Per contrastare questo problema, si è deciso di inserire uno strato di *dropout* dopo lo strato completamente connesso con un tasso di 0,85 ed il tasso d’apprendimento è stato abbassato da  $\alpha = 10^{-3}$  a  $\alpha = 10^{-5}$ . Il *dropout* è una tecnica di regolarizzazione che consiste nell’ignorare un certo numero di nodi durante la fase di addestramento della rete neurale. Ad ogni iterazione, i nodi vengono ignorati con una probabilità pari a  $1 - p$  dove  $p$  è il tasso di *dropout*.

Il nuovo modello, a cui è stato dato il nome di **ExoHunter**, è stato addestrato per 100 epoch con una dimensione delle batch di 64. Lo strato di output utilizza come funzione di attivazione la funzione sigmoidea, mentre lo strato completamente connesso utilizza la funzione ReLU. Una volta identificati i valori ottimali degli iperparametri, è stata utilizzata la tecnica del *model averaging*: sono state addestrate dieci copie indipendenti e calcolata la media dei risultati ottenuti. In Figura 4.6a si può osservare l’andamento dell’errore sul training set e sul validation set mentre in Figura 4.6b è possibile osservare i valori di *accuracy* nelle varie epoch durante l’addestramento sull’insieme di addestramento e di validazione. Entrambe le immagini fanno riferimento al modello che ha ottenuto i risultati migliori dei dieci addestrati, chiamato ExoHunter\_10. Tutti i file, *notebook*, modelli sono disponibili pubblicamente su GitHub<sup>2</sup>.

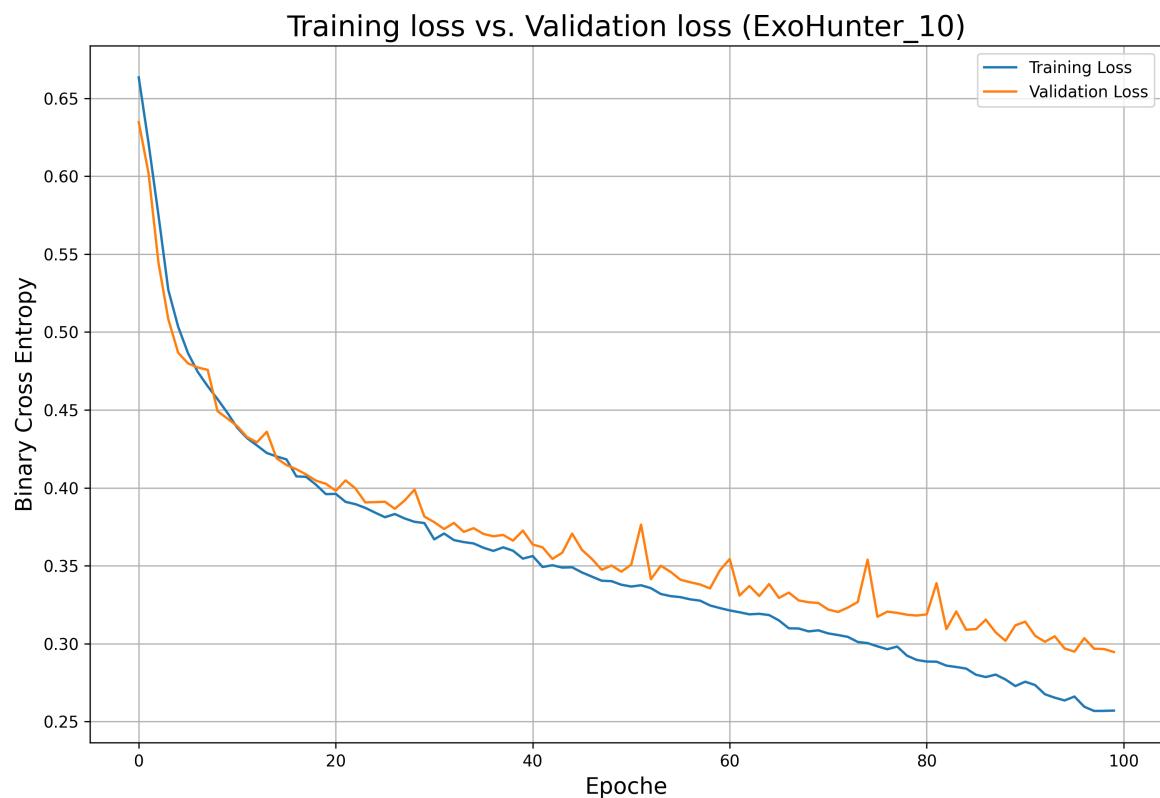
## 4.4 Risultati ottenuti

ExoHunter ottiene un *accuracy* media dell’89% sul test set. Un altro modo in cui si sono valutate le performance di ExoHunter è quello di utilizzare la matrice di confusione e la curva ROC-AUC.

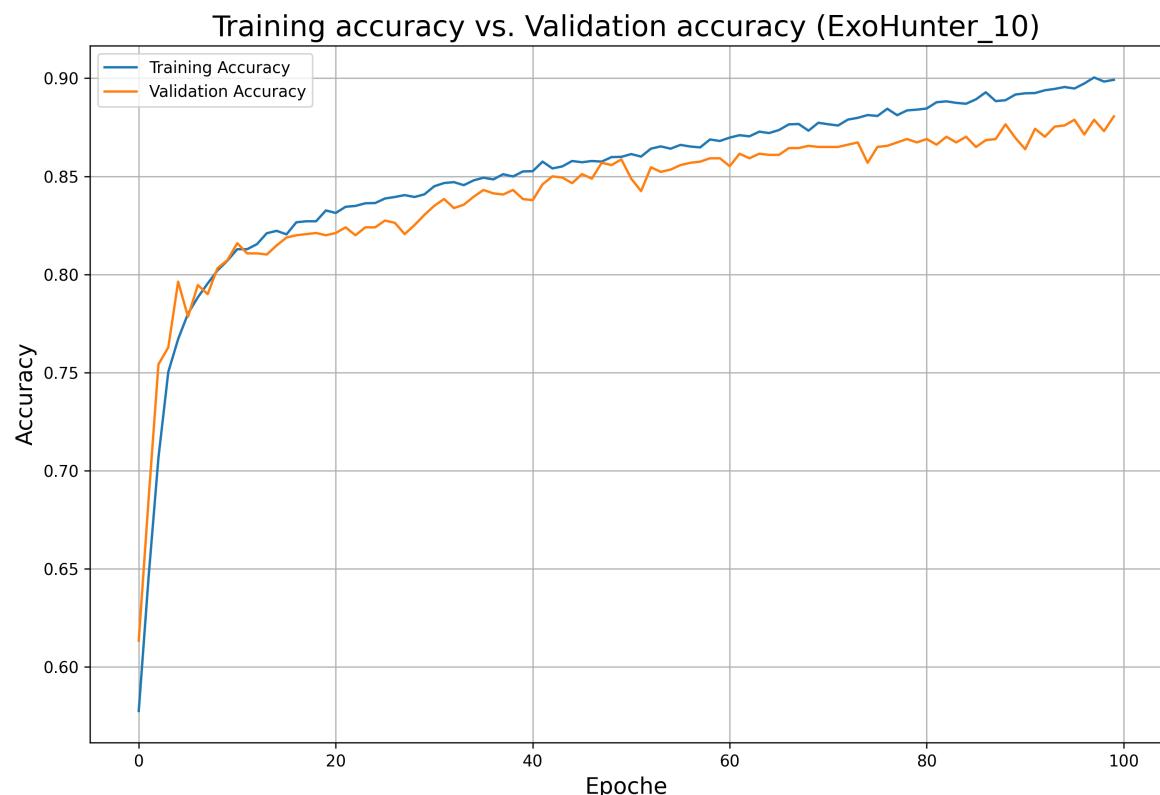
La matrice di confusione permette di visualizzare le prestazioni di un classificatore. Come possiamo osservare dalla Figura 4.7, in media sono state classificate correttamente 1706 curve di luce (834 falsi positivi e 872 confermati) mentre, sempre in media, 219 curve di luce sono state classificate erroneamente (91 curve di luce sono state classificate come falsi positivi e 128 falsi positivi sono stati classificati come curve di luce di esopianeti).

---

<sup>2</sup><https://github.com/giorgio-angelo-esposito/ExoHunter>

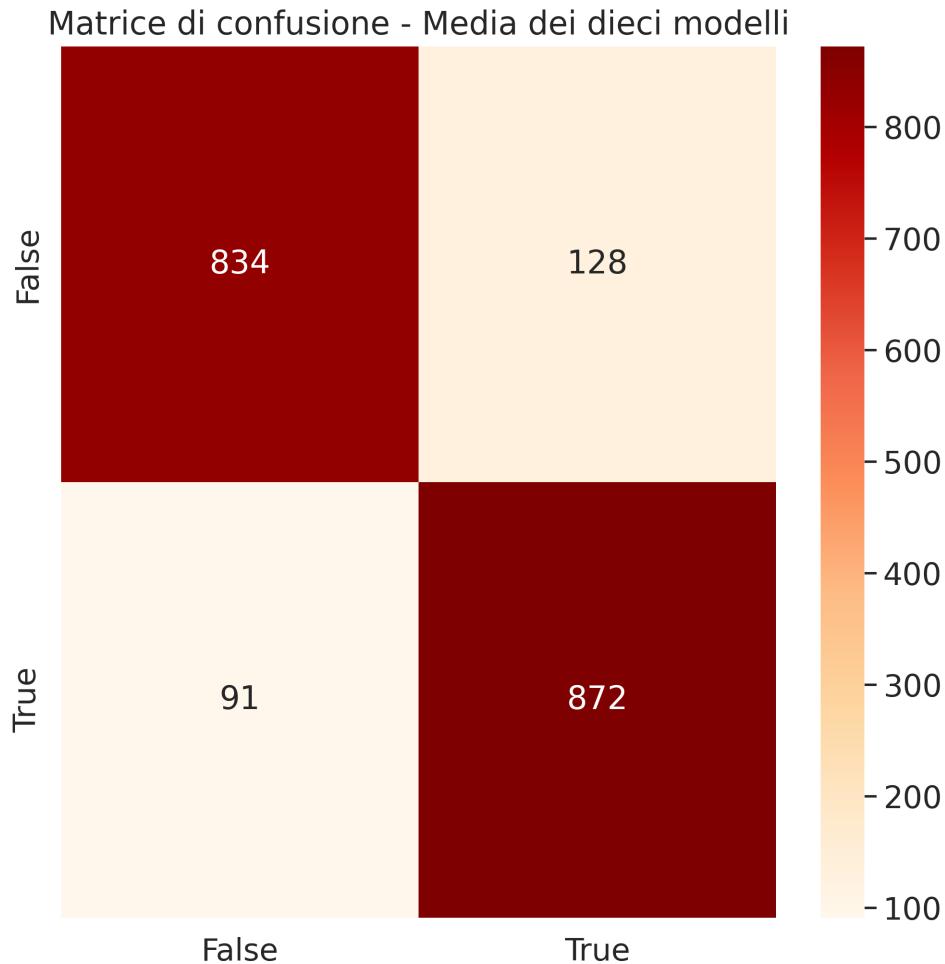


**(a)** L'andamento della funzione di perdita *binary crossentropy* sull'insieme di training (blu) e sull'insieme di validazione (giallo).



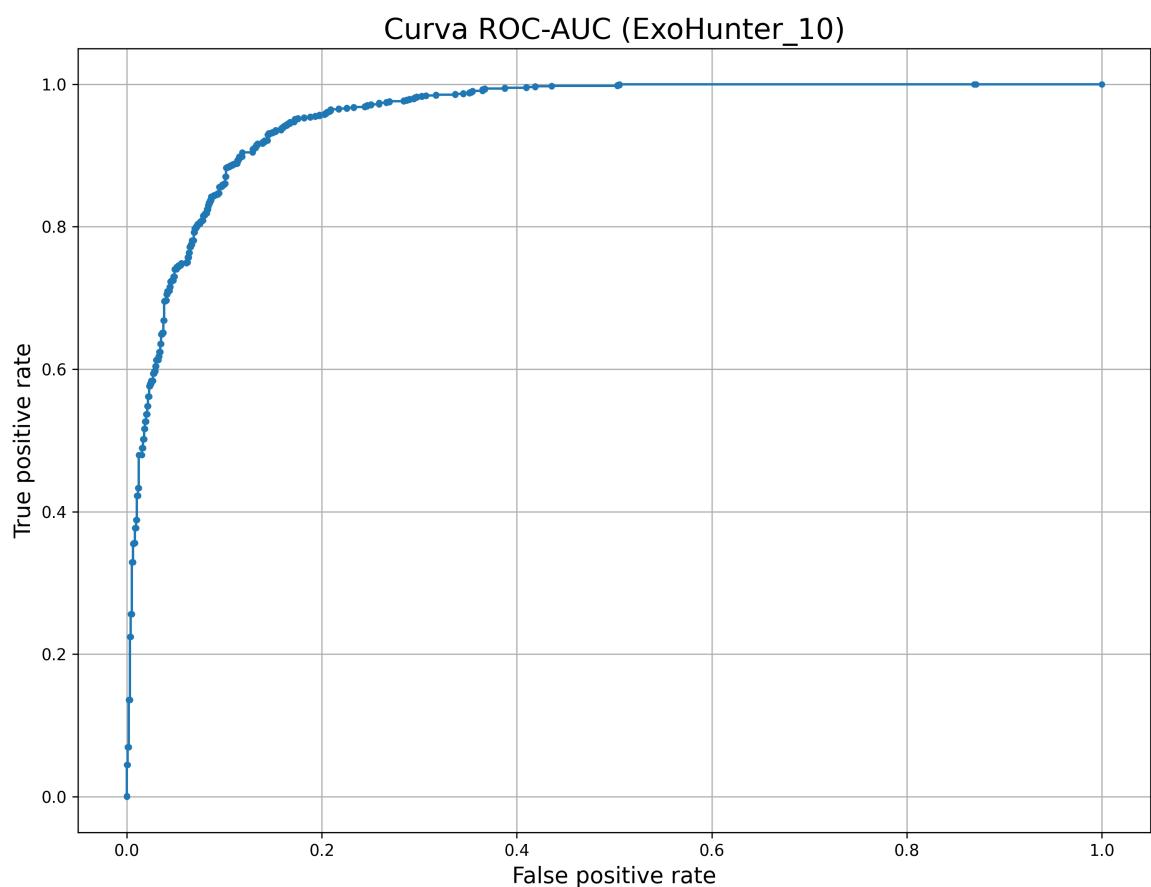
**(b)** L'andamento dell'accuracy sul training (blu) e sul validation set (giallo) col passare delle epoche.

**Figura 4.6:** Grafici dell'andamento della funzione di perdita e dell'*accuracy* rispetto alle epoche per ExoHunter\_10.



**Figura 4.7:** Matrice di confusione che riporta la media dei risultati di tutte le matrici di confusione calcolate dai dieci modelli sul test set.

La curva ROC mette in relazione il rapporto dei veri positivi (quindi la proporzione delle osservazioni predette vere rispetto a tutte le osservazioni positive) e dei falsi negativi (la proporzione delle osservazioni predette false rispetto a tutte le osservazioni false). Poichè il modello restituisce valori probabilistici, possiamo utilizzare diverse soglie di classificazione per indicare che un campione appartiene a una classe piuttosto che ad un'altra. La curva ROC mostra la relazione rispetto a tutti i valori soglia. L'area sottostante alla curva ROC (AUC) riassume in un singolo valore le prestazioni di ogni classificatore. In Figura 4.8 si può osservare la curva ROC-AUC calcolata sul test set con ExoHunter\_10, mentre il valore medio dell'AUC è di 0,95.



**Figura 4.8:** La curva ROC-AUC calcolata sul test set.

# CAPITOLO 5

---

## Conclusioni

---

Concludiamo discutendo i risultati ottenuti.

Come indicato nella formulazione del problema, si ribadisce che non è possibile confrontare in maniera speculare i risultati ottenuti con quelli presentati nel Capitolo 3. Questo perché il problema preso in esame si differenzia per ipotesi formulate di partenza.

I dieci modelli addestrati da cui sono stati ricavati i valori medi di *accuracy* e AUC si comportano sostanzialmente allo stesso modo: non sono stati riscontrati durante l'addestramento modelli meno performanti di altri e non ci sono stati blocchi in punti di minimo locale, cosa che porta la funzione di perdita a non diminuire ma a rimanere bloccata e di conseguenza il modello non apprende.

Come possiamo evincere dalla matrice di confusione in Figura 4.7 le curve di luce classificate correttamente sono in media circa 1 89% del totale, mentre il risultato medio dell'AUC, molto vicino ad 1, ci permette di capire che il modello è molto capace di distinguere tra le due classi. L'introduzione delle curve di luce artificiali hanno portato a un aumento delle prestazioni: addestrare la rete utilizzando il dataset senza le curve di luci artificiali risulta in *underfitting*. Il modello quindi non aveva abbastanza campioni per apprendere le sostanziali differenze tra curve di luce di esopianeti e falsi positivi.

Dati i buoni risultati ottenuti possiamo pensare ai futuri utilizzi e sviluppi di ExoHunter: si esclude l'utilizzo autonomo, ma in sintonia con la supervisione umana potrebbe ottenere ottimi risultati. Infatti, si potrebbe utilizzare la rete come ulteriore step di conferma durante

---

la convalida di un esopianeta.

Uno dei limiti osservati nello studio è la rappresentazione dell'input: utilizzare un procedimento simile a quello utilizzato da Shallue and Vanderburg potrebbe portare a risultati migliori. Questa è una delle possibili migliorie che si potrebbero implementare. Altra limitazione è dovuta al metodo di creazione del modello: utilizzando l'algoritmo di ottimizzazione bayesiano i migliori parametri trovati vengono utilizzati per la costruzione del modello successivo. Questo è uno dei motivi per cui il miglior modello identificato soffre di *overfitting*. Utilizzare un altro tipo di algoritmo potrebbe portare alla costruzione di un modello che non soffre di *overfitting* e ottiene anche prestazioni migliori. Inoltre, altra miglioria possibile è relativa al modo in cui si è scritto l'algoritmo di costruzione del modello che l'ottimizzatore ha utilizzato: tra i vari modelli creati, una decina di questi erano identici nella struttura. La differenza in questi modelli sta negli iperparametri non utilizzati: ad esempio, l'ottimizzatore costruisce il modello senza il secondo blocco, scegliendo comunque valori per gli strati del secondo blocco. Nell'iterazione successiva, farà la stessa cosa, ma scegliendo iperparametri diversi per gli strati del secondo blocco. Così facendo, spreca iterazioni. Migliorare l'algoritmo di costruzione può portare a maggiore diversità nei modelli, facendo contemporaneamente non sprecare iterazioni all'ottimizzatore. Sempre seguendo questa strada, si potrebbe tentare un approccio di costruzione del modello utilizzando gli algoritmi genetici.

Concludendo, l'approccio utilizzato costruendo una rete neurale convoluzionale si rivela comunque ottimo per affrontare il problema e ulteriori studi su questa strada potranno portare a risultati molto positivi.

---

## Ringraziamenti

---

Chi mi conosce sa quanto sia difficile per me riuscire ad esprimere le mie emozioni. Questa, infatti, è per me la parte più complicata di tutta questa tesi.

Inizio ringraziando il professore Fabio Palomba per aver supervisionato questa tesi, per la grande disponibilità mostrata durante le lezioni e per essere riuscito a farmi appassionare a quello che spero sia il mio futuro campo di studi. Un ringraziamento va anche al dottor Giammaria Giordano, per i numerosi consigli fornitemi nel corso di questi mesi durante la stesura di questa tesi.

Voglio ringraziare la mia famiglia per il supporto fornитomi in questi strani anni universitari, in particolare mia nonna Maddalena, capace con le sue parole di incoraggiamento di spronarmi a fare sempre il mio meglio.

Un sentito grazie va anche a Giacomo e Gerardo, conosciuto anche come *Picone*, veri amici dal tempo del liceo, con cui trovo sempre piacevole chiacchierare e scambiare idee.

Infine, visto che mi dicono che sono troppo duro con me stesso, ringrazio me stesso: "*Congratulazioni*".

---

## Bibliografia

---

- [1] Christopher J Fluke and Colin Jacobs. Surveying the reach and maturity of machine learning and artificial intelligence in astronomy. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10(2):e1349, 2020. (Citato a pagina 3)
- [2] Robert J Brunner, S George Djorgovski, Thomas A Prince, and Alex S Szalay. Massive datasets in astronomy. In *Handbook of massive data sets*, pages 931–979. Springer, 2002. (Citato a pagina 3)
- [3] EM Howard. Machine learning algorithms in astronomy. In *Astronomical Data Analysis Software and Systems XXV*, volume 512, page 245, 2017. (Citato a pagina 3)
- [4] Jason T Wright and B Scott Gaudi. Exoplanet detection methods. *arXiv preprint arXiv:1210.2471*, 2012. (Citato a pagina 7)
- [5] Christopher J Shallue and Andrew Vanderburg. Identifying exoplanets with deep learning: A five-planet resonant chain around kepler-80 and an eighth planet around kepler-90. *The Astronomical Journal*, 155(2):94, 2018. (Citato alle pagine 9, 13, 14, 15, 16, 17, 19, 20, 21, 25, 26, 31, 32, 34, 36 e 59)
- [6] Coryn AL Bailer-Jones, Ranjan Gupta, and Harinder P Singh. An introduction to artificial neural networks. *arXiv preprint astro-ph/0102224*, 2001. (Citato a pagina 10)
- [7] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985. (Citato a pagina 11)

- [8] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015. (Citato alle pagine 11, 12 e 13)
- [9] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. {TensorFlow}: a system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016. (Citato alle pagine 15 e 50)
- [10] Anne Dattilo, Andrew Vanderburg, Christopher J Shallue, Andrew W Mayo, Perry Berlind, Allyson Bieryla, Michael L Calkins, Gilbert A Esquerdo, Mark E Everett, Steve B Howell, et al. Identifying exoplanets with deep learning. ii. two new super-earths uncovered by a neural network in k2 data. *The Astronomical Journal*, 157(5):169, 2019. (Citato alle pagine 19, 20, 21, 23, 24 e 25)
- [11] Kaisey Mandel and Eric Agol. Analytic light curves for planetary transit searches. *The Astrophysical Journal*, 580(2):L171, 2002. (Citato alle pagine 21 e 40)
- [12] Megan Ansdell, Yani Ioannou, Hugh P Osborn, Michele Sasdelli, Jeffrey C Smith, Douglas Caldwell, Jon M Jenkins, Chedy Räissi, Daniel Angerhausen, et al. Scientific domain knowledge improves exoplanet transit classification with deep learning. *The Astrophysical journal letters*, 869(1):L7, 2018. (Citato alle pagine 25, 26, 28 e 32)
- [13] Liang Yu, Andrew Vanderburg, Chelsea Huang, Christopher J Shallue, Ian JM Crossfield, B Scott Gaudi, Tansu Daylan, Anne Dattilo, David J Armstrong, George R Ricker, et al. Identifying exoplanets with deep learning. iii. automated triage and vetting of tess candidates. *The Astronomical Journal*, 158(1):25, 2019. (Citato alle pagine 28, 30, 31, 32 e 33)
- [14] Chelsea X Huang, Andrew Vanderburg, Andras Pál, Lizhou Sha, Liang Yu, Willie Fong, Michael Fausnaugh, Avi Shporer, Natalia Guerrero, Roland Vanderspek, et al. Photometry of 10 million stars from the first two years of tess full frame images: Part i. *Research Notes of the AAS*, 4(11):204, 2020. (Citato a pagina 29)
- [15] Koko Visser, Bas Bosma, and Eric Postma. A one-armed cnn for exoplanet detection from light curves. *arXiv preprint arXiv:2105.06292*, 2021. (Citato alle pagine 33, 34, 36 e 39)

- [16] Lightkurve Collaboration, J. V. d. M. Cardoso, C. Hedges, M. Gully-Santiago, N. Saunders, A. M. Cody, T. Barclay, O. Hall, S. Sagear, E. Turtelboom, J. Zhang, A. Tzanidakis, K. Michell, J. Coughlin, K. Bell, Z. Berta-Thompson, P. Williams, J. Dotson, and G. Barentsen. Lightkurve: Kepler and TESS time series analysis in Python. *Astrophysics Source Code Library*, December 2018. (Citato alle pagine 36 e 47)
- [17] Sara Cuéllar, Paulo Granados, Ernesto Fabregas, Michel Curé, Héctor Vargas, Sebastián Dormido-Canto, and Gonzalo Farias. Deep learning exoplanets detection by combining real and synthetic data. *Plos one*, 17(5):e0268199, 2022. (Citato alle pagine 39, 40, 41, 42, 43 e 44)
- [18] A. Ginsburg, B. M. Sipőcz, C. E. Brasseur, P. S. Cowperthwaite, M. W. Craig, C. Deil, J. Guillochon, G. Guzman, S. Liedtke, P. Lian Lim, K. E. Lockhart, M. Mommert, B. M. Morris, H. Norman, M. Parikh, M. V. Persson, T. P. Robitaille, J.-C. Segovia, L. P. Singer, E. J. Tollerud, M. de Val-Borro, I. Valtchanov, J. Woillez, The Astroquery collaboration, and a subset of the astropy collaboration. astroquery: An Astronomical Web-querying Package in Python. , 157:98, March 2019. doi: 10.3847/1538-3881/aafc33. (Citato a pagina 47)
- [19] Tom O'Malley, Elie Bursztein, James Long, François Chollet, Haifeng Jin, Luca Invernizzi, et al. Kerastuner. <https://github.com/keras-team/keras-tuner>, 2019. (Citato a pagina 51)
- [20] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012. (Citato a pagina 52)