Relazione attività progettuale
computer vision and image processing

Giovanni Baratta
Matricola 0000853633

Analisi e pre-processing del dataset

La prima fase del processo di sviluppo del classificatore è stata dedicata ad una breve analisi del dataset. Il dataset contiene delle immagini appartenenti a 43 categorie differenti non equamente distribuite. Ad ogni categoria è associato un cartello stradale reale. Le 43 categorie possono essere divise in 5 gruppi semantici.

Tabella 1 - A sinistra il gruppo, a destra il numero di immagini disponibili nel training set

0	210
1	2220
2	2250
3	1410
4	1980
5	1860
6	420
7	1440
8	1410
9	1470
10	2010
11	1320
12	2100
13	2160
14	780
15	630
16	420
17	1110
18	1200
19	210
20	360
21	330
22	390
23	510
24	270
25	1500
26	600
27	240
28	540
29	270
30	450
31	780
32	240
33	689
34	420
35	1200
36	390
37	210
38	2070
39	300
40	360
41	240
42	240

All'interno di ogni categoria, per ogni cartello fisico reale sono presenti 30 immagini dello stesso cartello con dimensione, posizione, illuminazione e prospettiva leggermente differenti. L'unica eccezione è rappresentata dal gruppo 19 della classe 33 che ha un'immagine in meno delle altre.

Dimensioni

La dimensione delle immagini è variabile, la dimensione minima è 15x15 mentre la massima 250x250. La maggior parte delle immagini è di dimensioni ridotte, inferiori a 50x50. Per semplificare la struttura della rete è stata scelta una dimensione fissa e tutte le immagini sono state ridimensionate. Il ridimensionamento è stato effettuato preservando l'*aspect ratio* delle immagini aggiungendo del padding ai bordi. Il padding aggiuntivo è una copia dei pixel dell'immagine lungo i bordi. In alternativa al ridimensionamento si potevano utilizzare tecniche più complesse come lo *spatial pyramid pooling* [1].

Inizialmente la dimensione scelta è stata 75x75 ma per limiti legati alla RAM disponibile e per la necessità di generare immagini aggiuntive a partire dalle originali la dimensione è stata diminuita a 48x48.

Pre-processing

Avendo a disposizione un test set già pronto, il training set è stato diviso unicamente in due gruppi, training e validation set, rispettivamente contenenti il 70% e il 30% delle immagini. Come detto precedentemente per ogni cartello reale sono presenti 30 immagini. Per cercare di creare un validation set il più rappresentativo possibile del test set, per ogni cartello, sono state scelte casualmente 3 immagini ogni 10 invece si sceglierne casualmente 9 dalle 30 presenti. La fase successiva è stata la normalizzazione dei dati. Inizialmente era stata calcolata un'immagine che

rappresentasse la media di tutto il training set, quest'ultima veniva sottratta dalle immagini utilizzate per il training, la fase di validazione e di testing. Successivamente provando semplicemente a riscalare i valori dei pixel nel range 0-1 è stato possibile ottenere performance leggermente superiori.

Sono state generate tre coppie di training e validation set da utilizzare durante l'allenamento per provare ad ottenere risultati migliori combinando il risultato di più classificatori.

Classificatore

Per realizzare il classificatore sono stati utilizzati due approcci differenti. Il primo prevede l'uso di un classificatore per l'inferenza della classe utilizzando tutte le categorie presenti. Il secondo invece usa sei classificatori, il primo per determinare il gruppo semantico dell'immagine, i restanti cinque per calcolare la label definitiva. In base al gruppo semantico viene applicato un classificatore specifico.

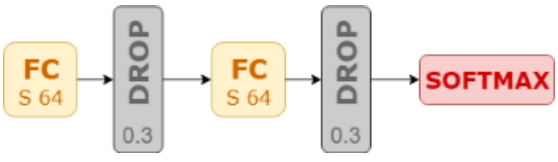
Classificatore tipo 1

Il primo passo per costruire il classificatore è stato quello di provare diversi modelli di reti in modo da individuare quello più promettente per migliorarlo in una fase successiva. I primi modelli non presentano una profondità elevata. Ogni rete è stata allenata fino a 50 epoche utilizzando 6 valori di learning rate differenti, interrompendo l'allenamento se la funzione di loss sul validation set non diminuiva in modo significativo. Per tutte le reti è stata usata come funzione di attivazione la *ReLu* ad eccezione dell'ultimo livello in cui è necessario utilizzare una funzione *softmax* per poter ottenere la label calcolata dalla rete. La tabella seguente mostra i modelli utilizzati per questa fase.

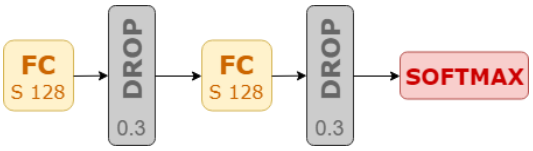
MODELLO 1 Parametri 445.655

MODELLO 2 Parametri 891.095

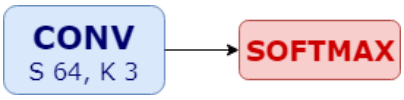
MODELLO 3 Parametri 450.071



MODELLO 4 Parametri 908.119

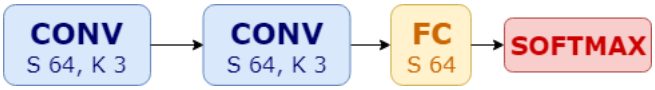


MODELLO 5 Parametri 6.342.871



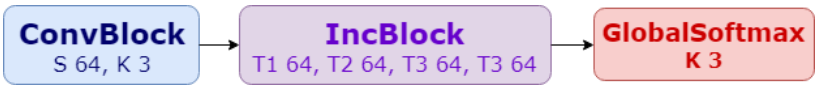
MODELLO 6 Parametri 6.380.055

MODELLO 7 Parametri 9.479.703



MODELLO 8 Parametri 2.401.815

MODELLO 9 Parametri 333.271



MODELLO 10 Parametri 535.127

Descrizione dei modelli

I primi quattro modelli sono delle reti composte unicamente da layer fully connected, nel terzo e quarto modello sono stati aggiunti i layer di dropout per ridurre l'overfitting e cercare di aumentare la precisione del classificatore. I modelli 5 e 6 fanno uso unicamente di layer di convoluzione, per entrambi si usa un kernel di dimensioni 3x3, generando 64 immagini per l'estrazione delle features. Il modello 7 unisce i layer di convoluzione con i layer fully connected mentre il modello 8 utilizza il classico schema delle CNN, dei layer di convoluzione seguito da un layer di max pooling per effettuare il downsampling. I modelli 9 e 10 fanno uso dei blocchi sviluppati ed utilizzati all'interno delle reti Inception [2] e ResNet [3], rispettivamente di Google e Microsoft. Sono gli unici due modelli utilizzati che non fanno uso di uno schema sequenziale ma utilizzano dei blocchi in parallelo. Inoltre, invece dei classici layer fully connected finali, utilizzano il global pooling, ovvero in questo caso specifico vengono generate 43 immagini finali e da ognuna di esse viene estratto il valore medio. Quest'ultimi valori vengono poi utilizzati per la classificazione. In questo modo si riducono drasticamente il numero di parametri della rete, riducendo di conseguenza l'overfitting [4].

Performance

Di seguito vengono elencati i risultati sul training, validation e test set per ogni modello elencato precedentemente, viene riportato il miglior valore ottenuto durante il training utilizzando come metrica per la bontà del modello la loss calcolata sul validation set. Nell'appendice 2 sono presenti i training completi effettuati sui modelli.

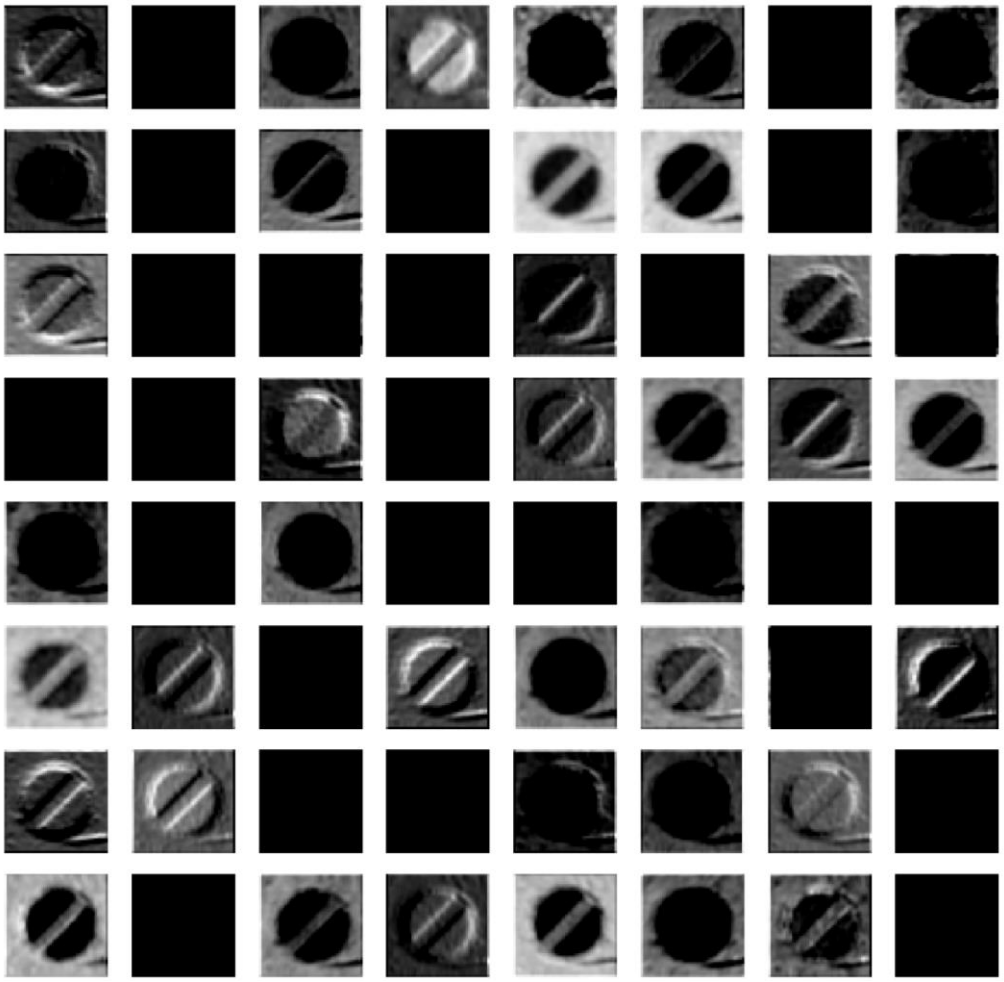
Modello (LR)	Tr	Val	Test
1 (10^{-6})	95.28 %	92.49 %	80.29 %
2 (10^{-6})	95.74 %	92.99 %	81.50 %
3 (10^{-7})	88.11 %	95.40 %	87.22 %
4 (10^{-7})	93.10 %	97.20 %	88.46 %
5 (10^{-7})	100 %	97.77 %	87.23 %

Modello (LR)	Tr	Val	Test
6 (10^{-6})	100 %	99.03 %	94.67 %
7 (10^{-7})	100 %	99.04 %	95.11 %
8 (10^{-7})	100 %	99.17 %	95.32 %
9 (10^{-3})	100 %	99.99 %	98.06 %
10 (10^{-4})	100 %	99.99 %	98.67 %

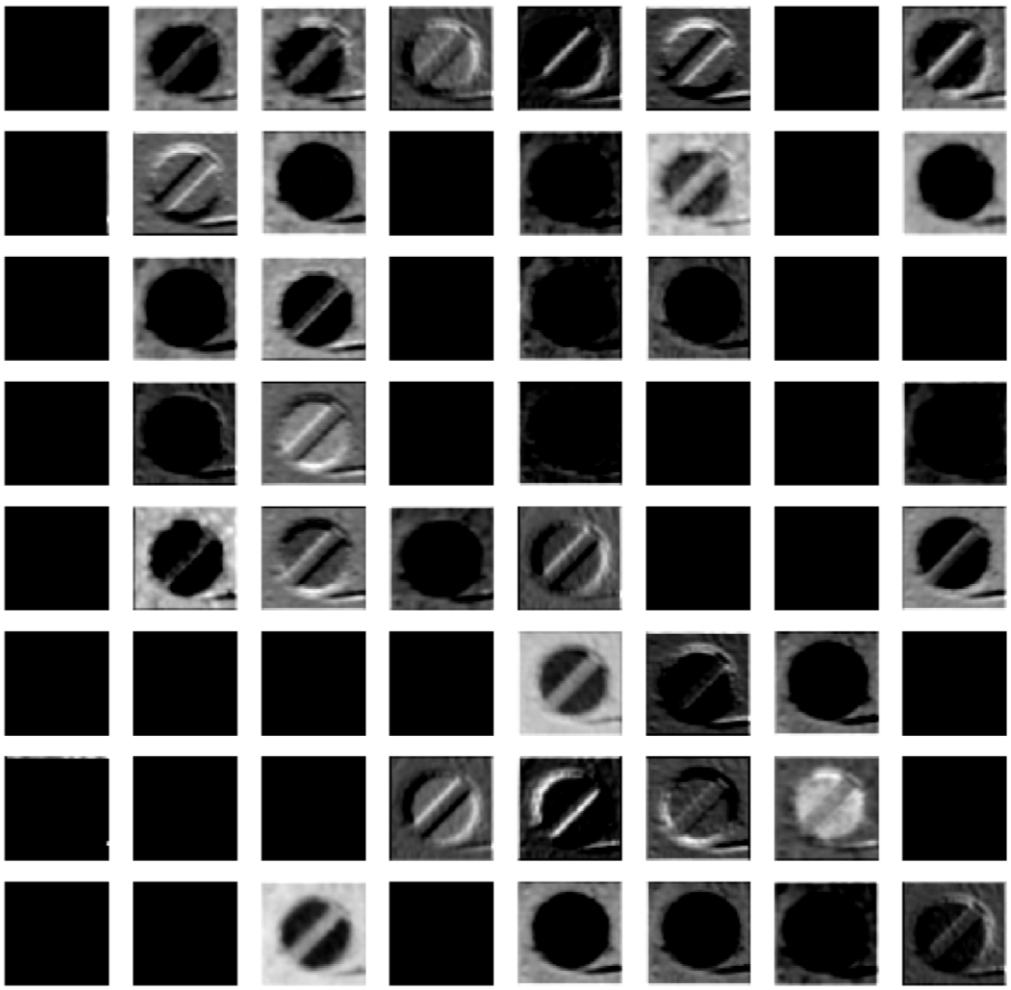
Come si può notare dalla tabella precedente, le reti che fanno uso di layer di convoluzione hanno una precisione nettamente superiore e riescono a generalizzare meglio, mentre utilizzano unicamente dei layer fully connected la precisione non supera il 90%. Reti più profonde come le ultime due riescono ad ottenere risultati migliori. In tutti i casi è presente dell'overfitting, in alcuni casi più limitato, in altri molto evidente. Nonostante il numero di parametri decisamente inferiore, i modelli 9 e 10 riescono ad avere performance migliori del modello numero 8.

Activation map

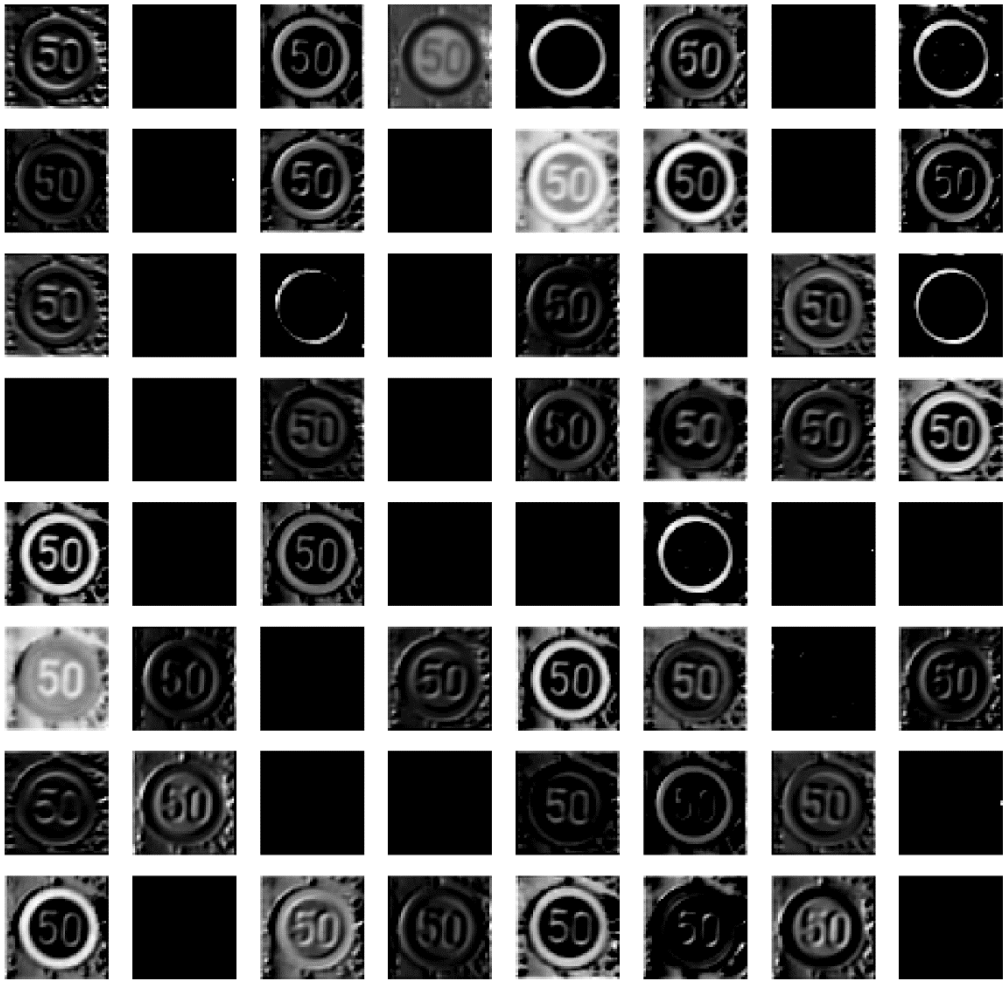
Di seguito vengono riportati due esempi di funzioni di attivazione generate dal primo layer di convoluzione dei modelli numero 9 e numero 10 con due immagini prese dal test set.



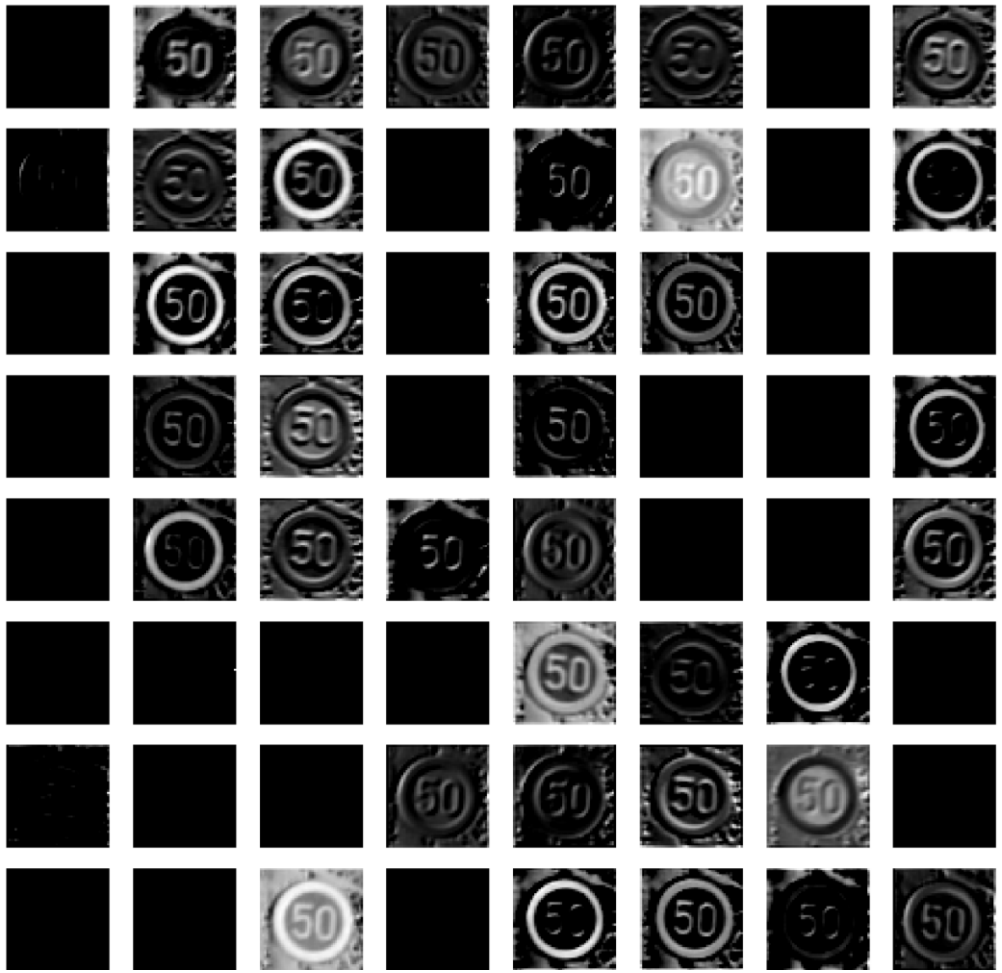
Modello 9



Modello 10



Modello 9



Modello 10

Miglioramenti

Gli ultimi tre modelli (CNN, Inception e ResNet) sono stati scelti come base di partenza per provare a migliore ulteriormente le performance. Le modifiche effettuate ai modelli consistono nel :

- aumentare il numero di layer dei modelli,
- aumentare la dimensione dei singoli layer,
- utilizzo di layer di *dropout* per l'overfitting
- aumentare il numero di immagini di training e utilizzare sampling differenti,
- utilizzare più modelli per l'inferenza,
- modifica degli iperparametri.

Per il quarto punto sono state utilizzate le API di DataAugmentation fornite da Keras che permettono di manipolare le immagini in modo semplice. Tramite le API è possibile agire sulla rotazione, zoom, spostamento, shear, alterazione della luminosità e alterazione dei canali dell'immagine. Ogni trasformazione viene applicata generando casualmente i parametri all'interno dei range specificati. È importante scegliere accuratamente i range dei parametri per evitare di alterare la semantica dell'immagine.

Rotation (degrees)	± 25	**Zoom**	(0.4, 1.15)
Shift	± 8	**Channel shift**	0.13
Shear (degrees)	± 15	**Brightness shift**	[0.25, 0.6]

Ad esclusione di qualche caso particolare, le immagini ottenute non alterano la semantica associata alla classe di partenza.

Generare delle immagini aggiuntive per il dataset di validazione può essere utile per ottenere una metrica più affidabile delle reali capacità del modello. Come si può vedere dai risultati ottenuti sui primi modelli, gli ultimi tre, riescono a memorizzare completamente tutto il training set ottenendo ottimi risultati sul set di validazione ma performance inferiori sul set di test.

Ensemble

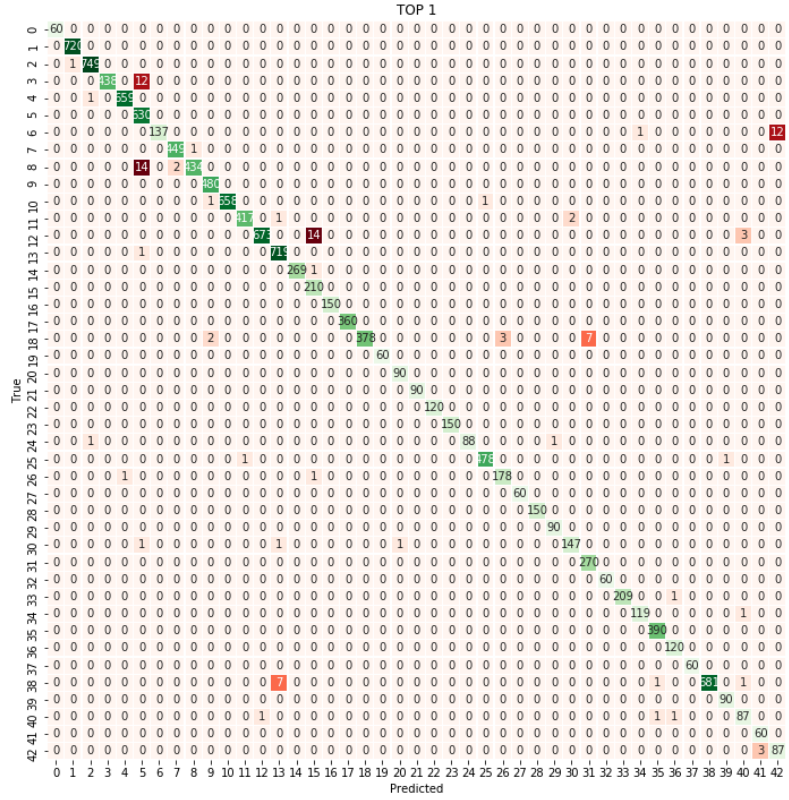
In alcuni casi, dopo aver raggiunto un limite all'accuratezza raggiungibile con un singolo modello, sono stati utilizzati due o più modelli insieme. Per fare questo sono stati sommati i vettori in uscita dai modelli, e poi è stato utilizzato l'indice del valore massimo come label finale.

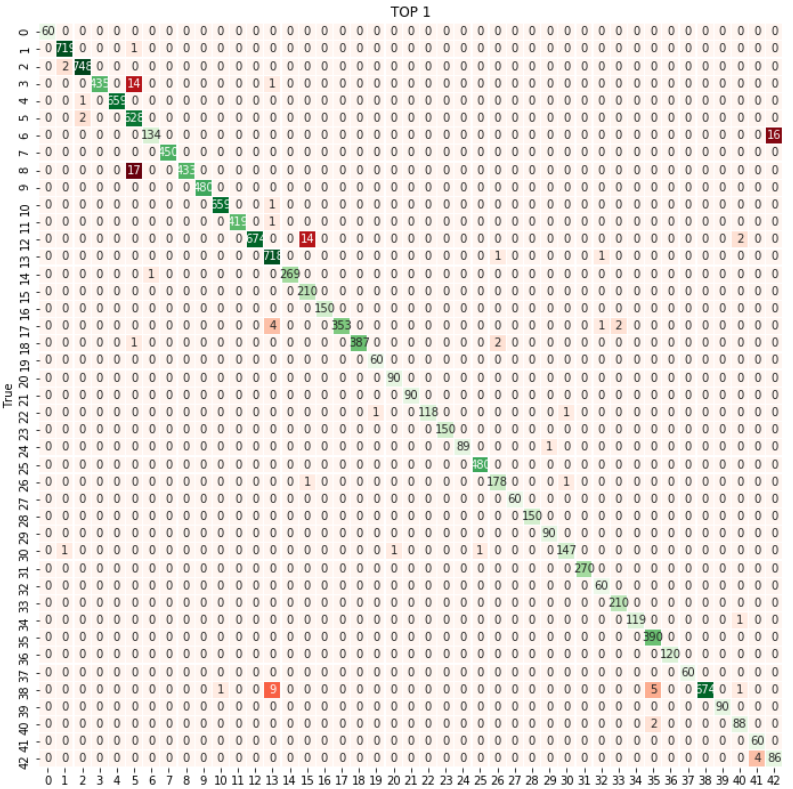
CNN

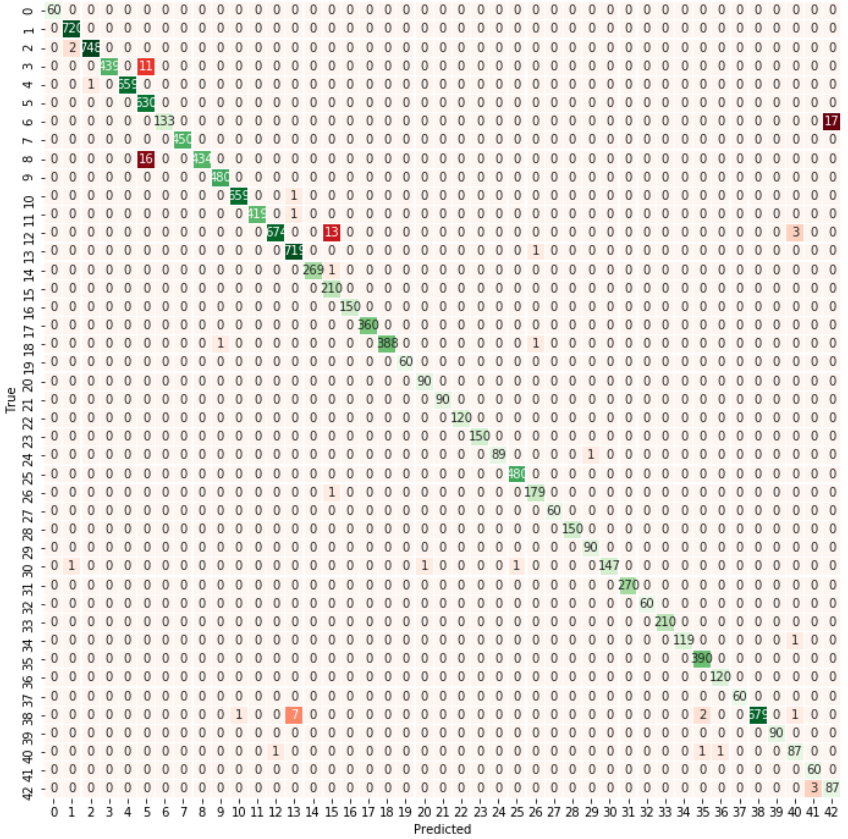
Partendo dal modello 8 è stato possibile raggiungere il 99,16 % di accuratezza sul test set con un singolo modello con le seguenti modifiche :

- Utilizzo di un kernel di dimensioni 5x5 per i layer di convoluzione
- Utilizzo di 6 layer di convoluzione di dimensioni 64, 96, 128, 164, 192 e 256 con un max pooling dopo il terzo

Con le seguenti modifiche, il numero di parametri del modello risulta essere 5.373.483 . Se si combina il seguente modello con un modello con gli ultimi tre layer di convoluzione di dimensioni pari a 128 si riesce a raggiungere un'accuratezza del 99.27 %.







In alto a sinistra CNN con 99,16% di accuracy, in alto a destra un modello alternativo con accuracy del 99.08%, a destra l'unione dei due modelli con un'accuracy del 99,27%

Inception

Nel caso del modello con blocchi Inception è stato possibile aumentare l'accuratezza del modello utilizzando più blocchi ed avendo di conseguenza una rete più profonda. L'accuratezza massima raggiunta da un singolo modello è di 98,99 %, mentre combinando i due modelli è stato possibile ottenere un'accuratezza del 99.24 %. Il modello migliore ha un numero di parametri pari a 953.431, un quinto del modello CNN.

ResNet

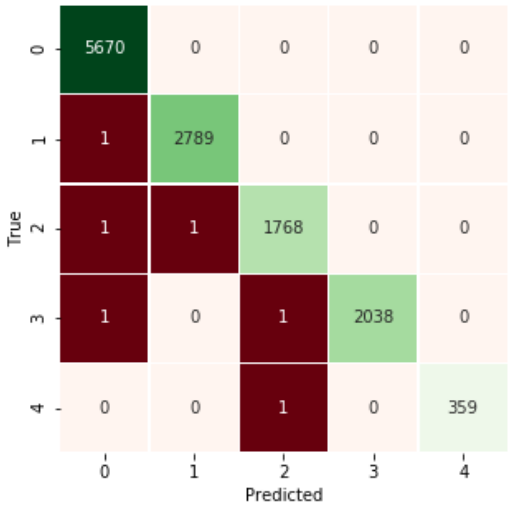
Infine, per l'ultimo modello, non ci sono stati miglioramenti significativi anche utilizzando modelli più profondi e con layer più grandi.

Classificatore tipo 2

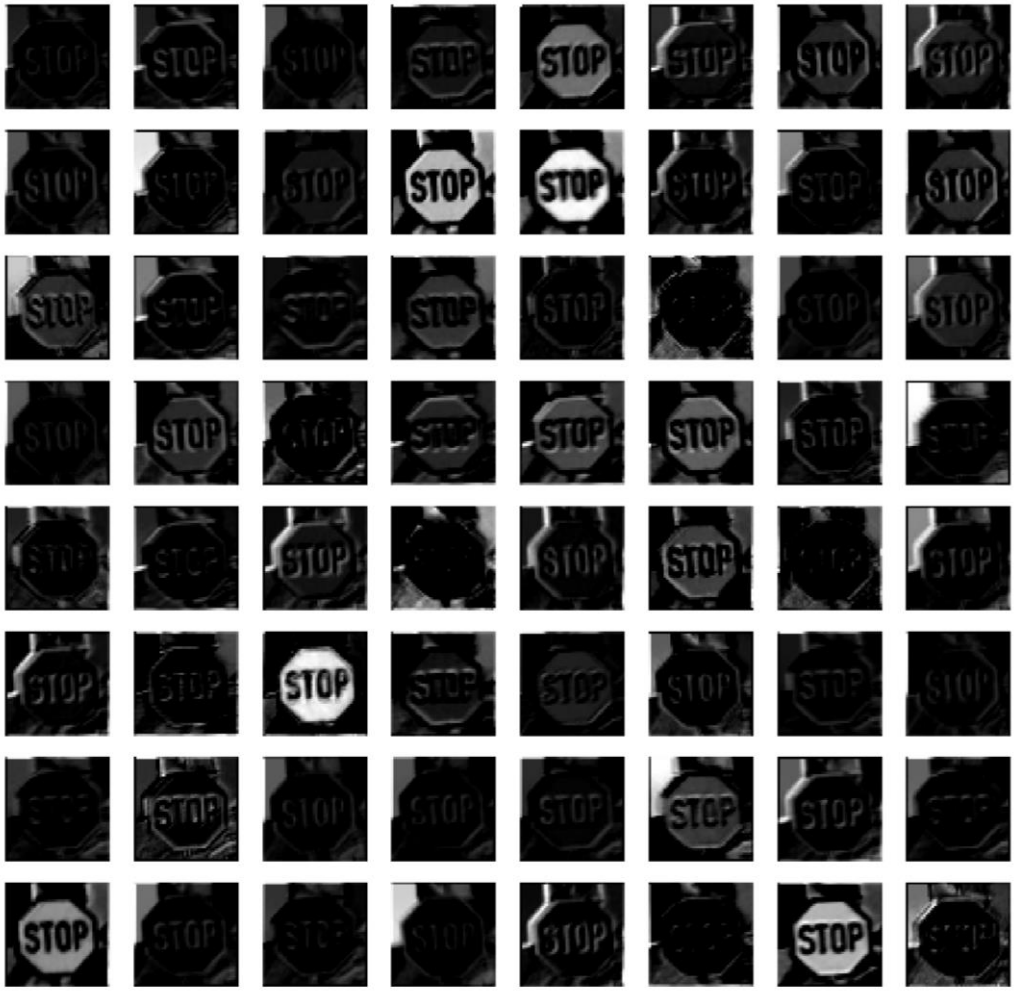
Come accennato precedentemente, il secondo approccio è stato quello di sviluppare un classificatore composto da due parti. La prima parte è responsabile di determinare il gruppo semantico del cartello (nella prima tabella vengono riportare le label suddivise per gruppo), mentre la seconda di individuare la classe reale dell'immagine. In base al gruppo calcolato dal primo classificatore si fa uso di un classificatore specifico. I gruppi semantici sono :

- in giallo i segnali di divieto,
- in verde fine divieto,
- in rosso i segnali di pericolo,
- in grigio i segnali di d'obbligo,
- in blu i segnali misti.

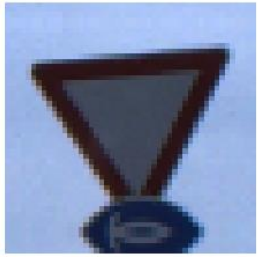
Il classificatore della prima parte viene allenato utilizzando tutte le immagini presenti nel training set mentre i cinque classificatori specifici per ogni gruppo sono stati allenati con un sottoinsieme del training set con le classi appartenenti al gruppo semantico al quale fanno riferimento. Il primo classificatore riesce ad ottenere un'accuratezza del 99.95 % sbagliando unicamente 6 immagini su 12630.

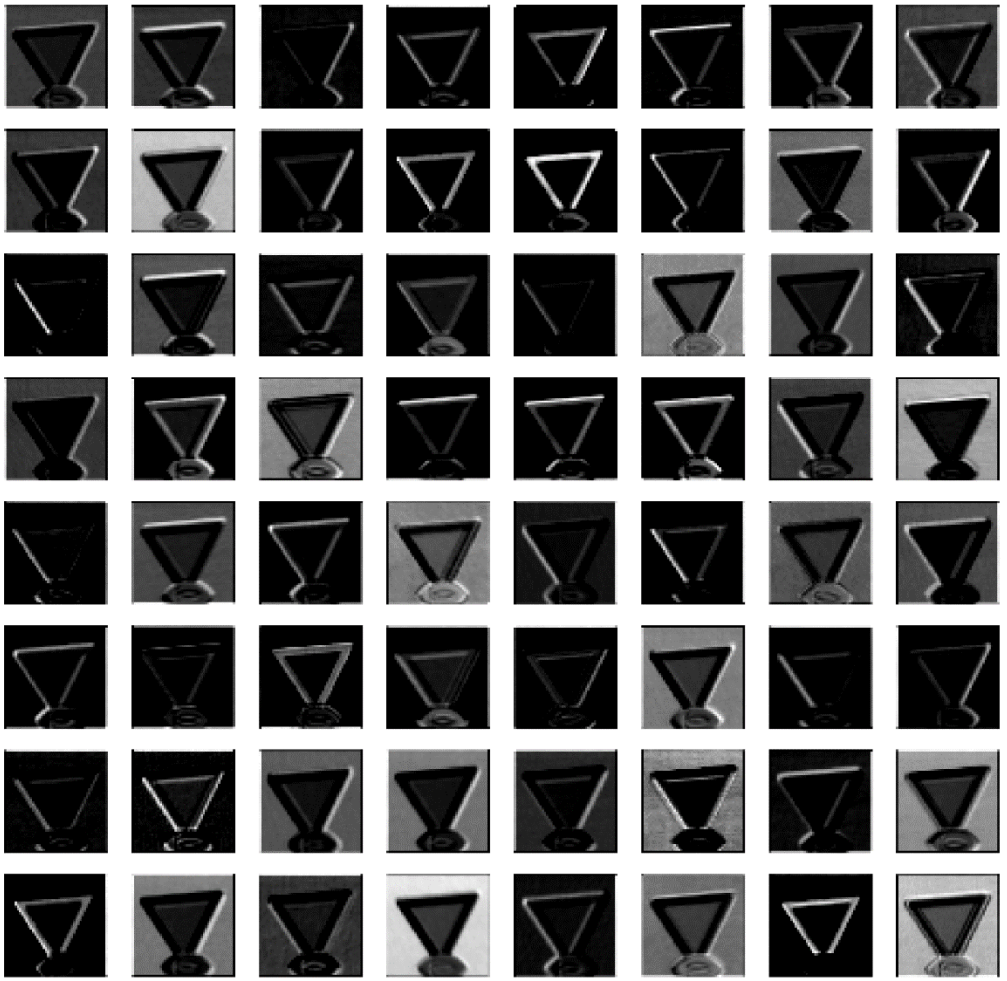


Grazie all'uso di singoli modelli specializzati è stato possibile ottenere un'accuratezza complessiva del 99.47%. I sei modelli utilizzati sono riportati all'interno del notebook utilizzato per l'allenamento. Di seguito sono riportate le funzioni di attivazioni con due immagini di esempio.



Step 2 - Modello gruppo segnali misti

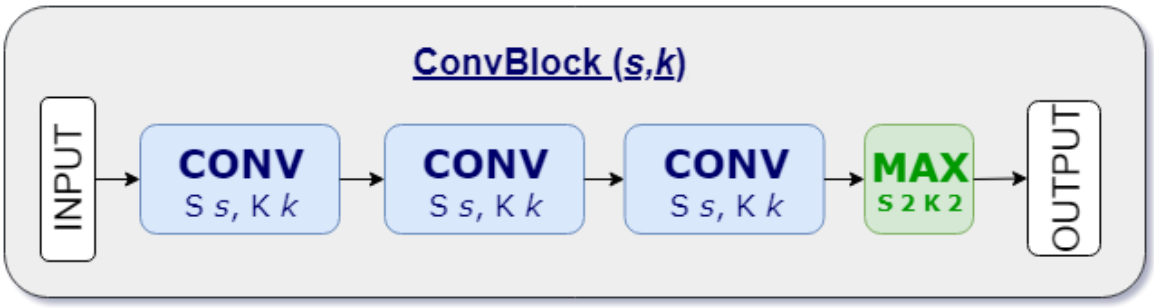




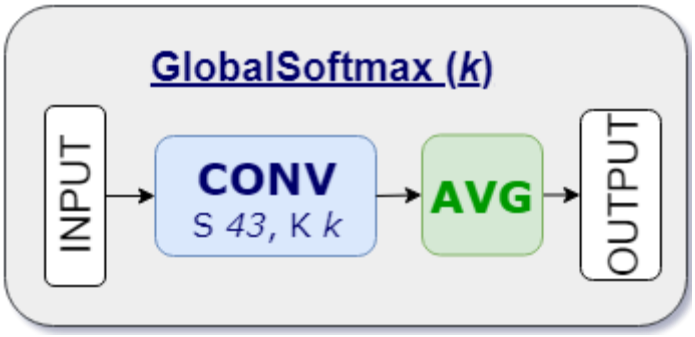
Step 2 – Modello gruppo segnali misti

Appendice 1 – Blocchi utilizzati

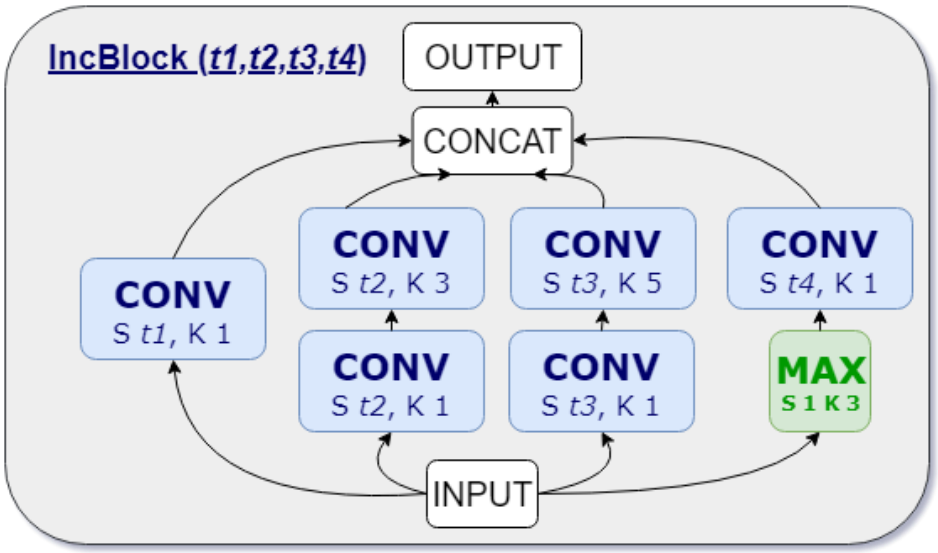
Blocco di layer per la convoluzione, la concatenazione di più layer di convoluzione con kernel di piccole dimensioni permette di ottenere risultati simili ad un unico blocco di convoluzione ma con un kernel di dimensioni maggiori riducendo il numero di parametri [5].

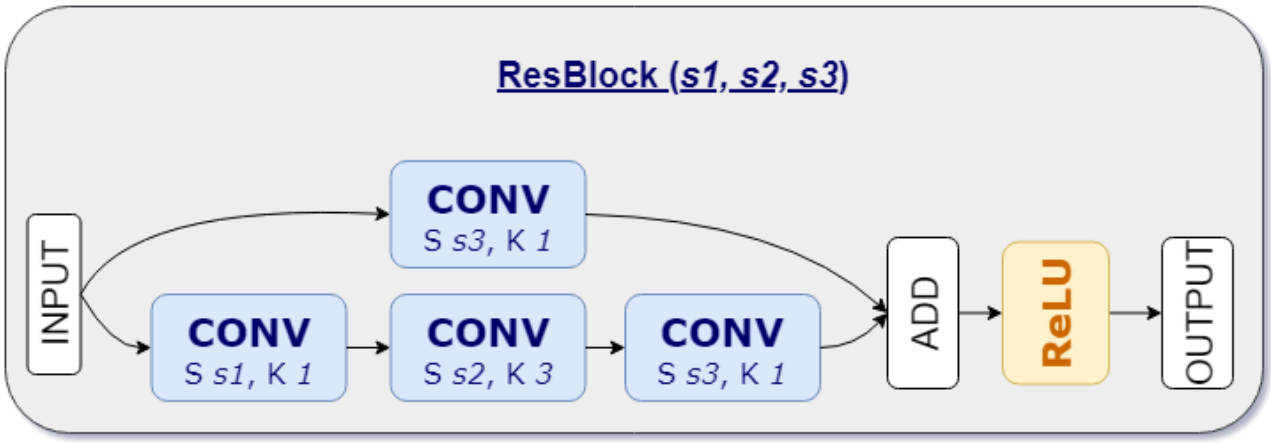


Tramite i blocchi di global average pooling è possibile ridurre drasticamente il numero di parametri generati dai layer fully connected. Invece di applicare i layer Dense all'output di un layer di convoluzione (ad esempio 6x6x256), si utilizza un layer di convoluzione aggiuntivo che generi N funzioni di attivazione e a queste si applica un layer di pooling globale. In questo modo per ogni layer di attivazione si ottiene un valore unico e non una matrice. I valori ottenuti possono essere utilizzati direttamente per la classificazione oppure è possibile reintrodurre dei Dense layer che essendo applicati a solo N valori generano meno parametri.



I due blocchi successivi schematizzano i blocchi utilizzati all'interno delle reti Inception e ResNet.

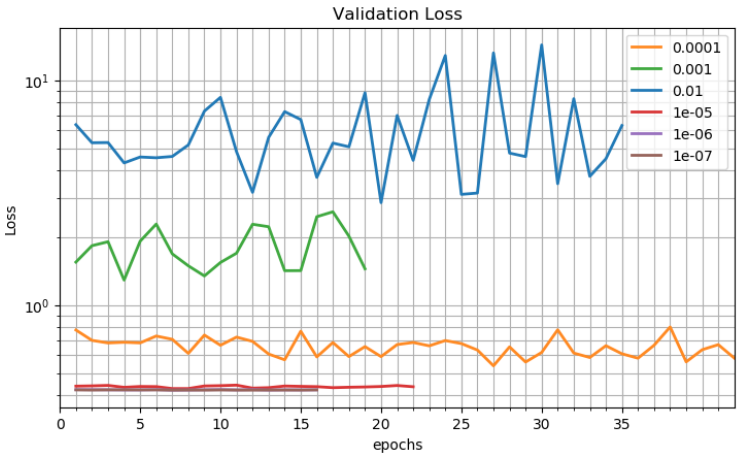


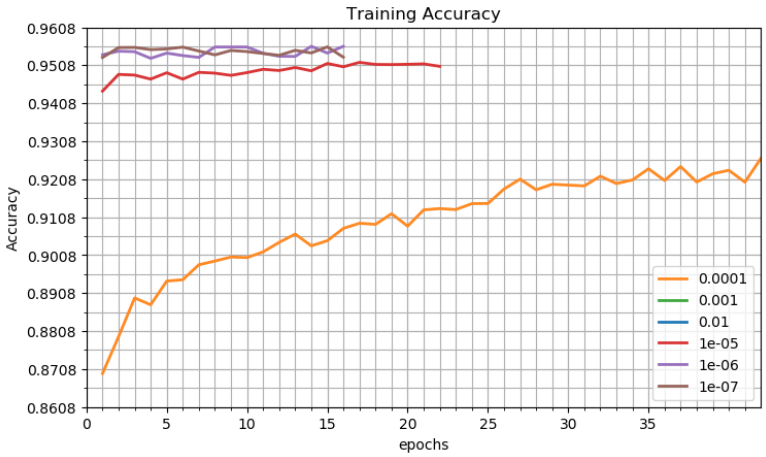


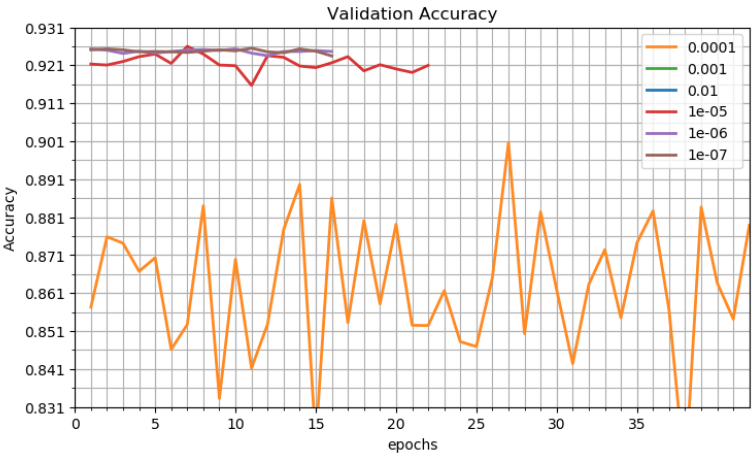
Appendice 2 – Risultati fase

Per ogni modello vengono riportati i valori di *training loss*, *training accuracy*, *validation loss* e *validation accuracy* al variare del learning rate.

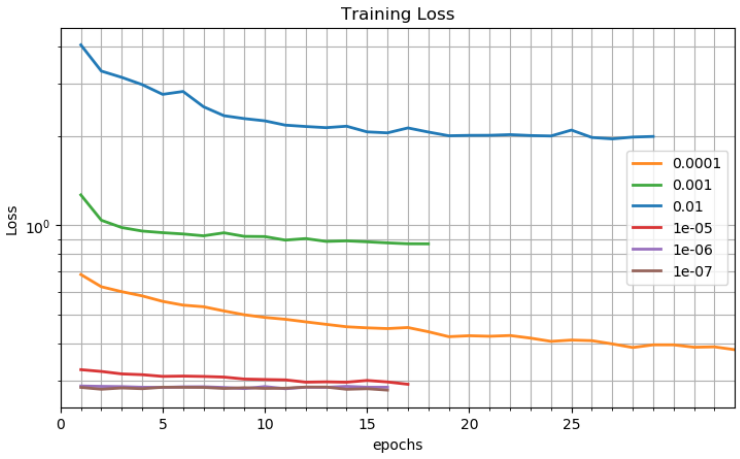
Modello 1 – Dense64

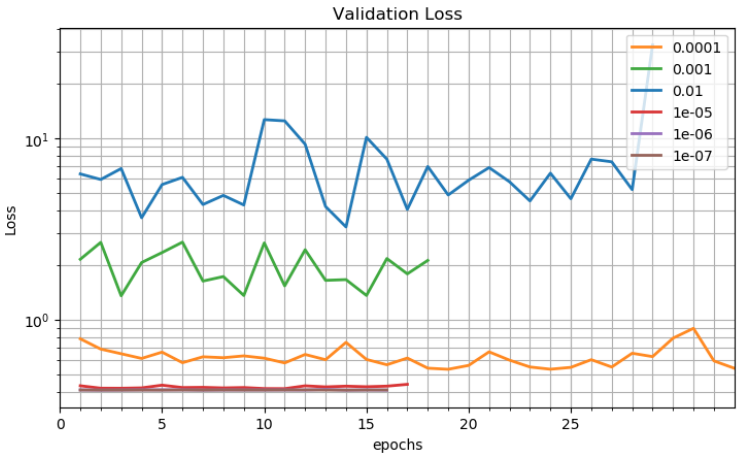


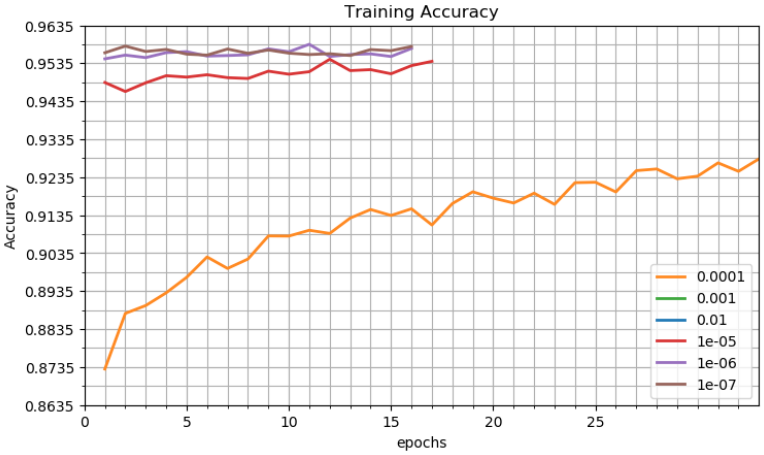


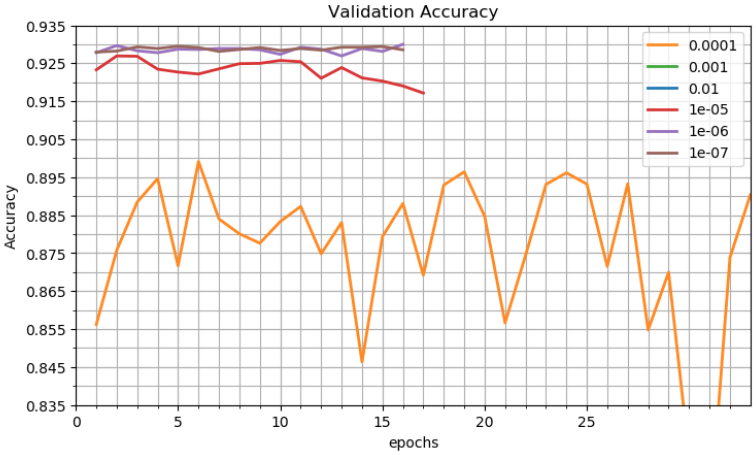


Modello 2 - Dense128

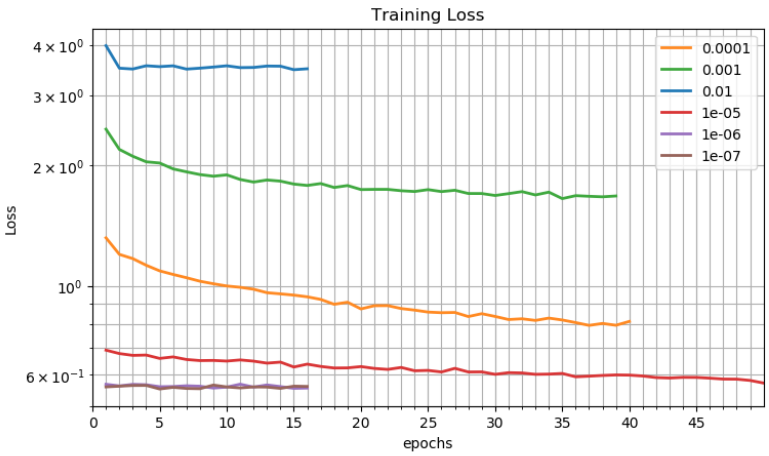


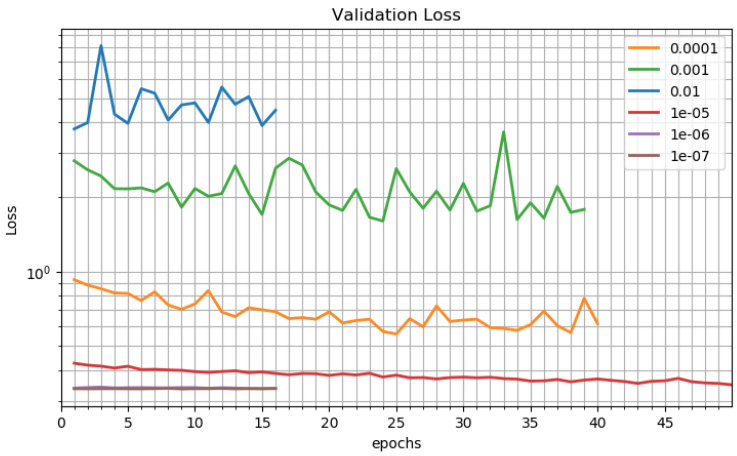


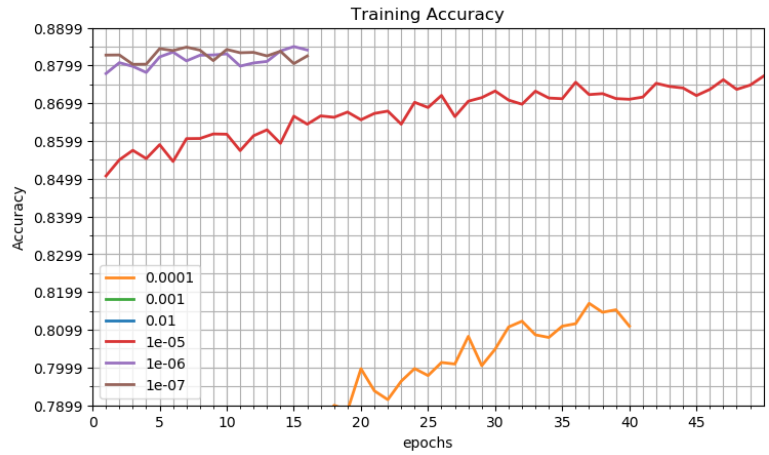


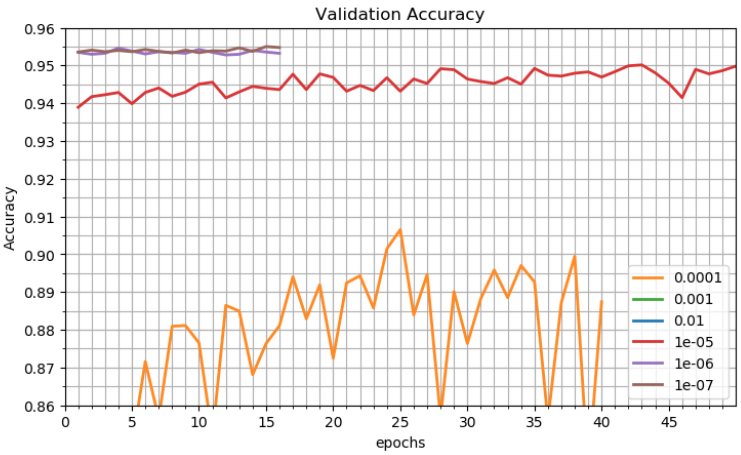


Modello 3 - Dense64 -> Dense64

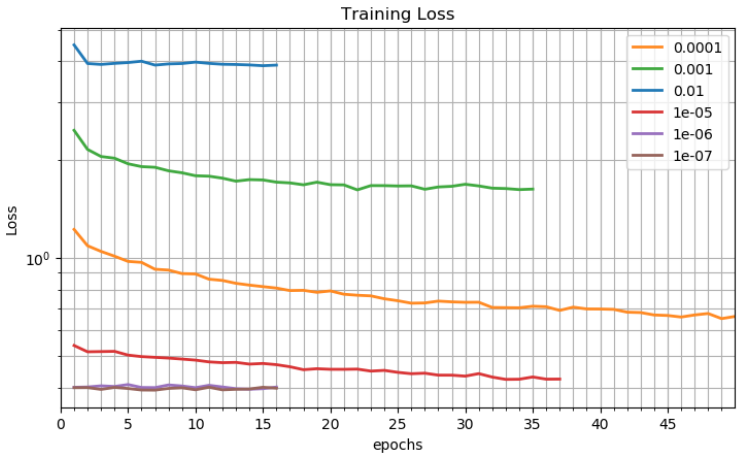


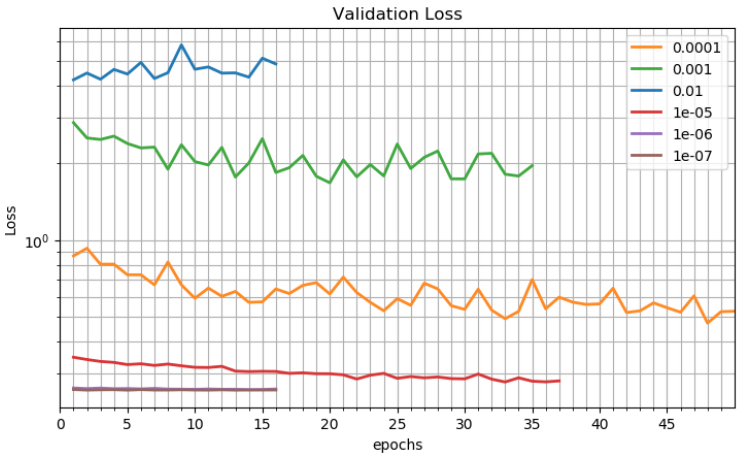


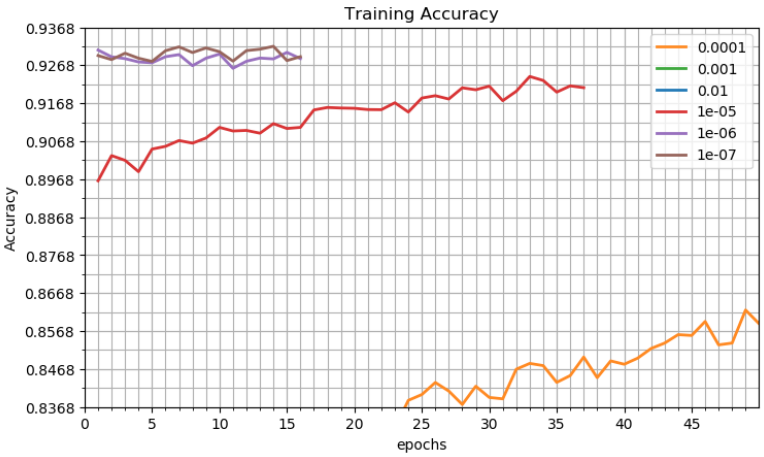


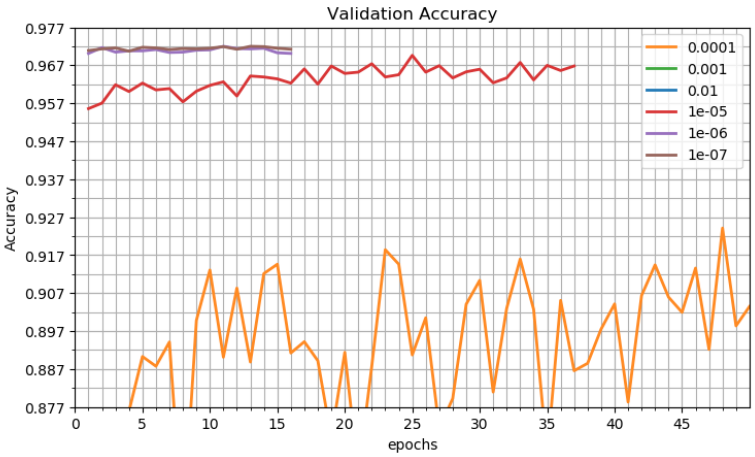


Modello 4 - Dense128 -> Dense128

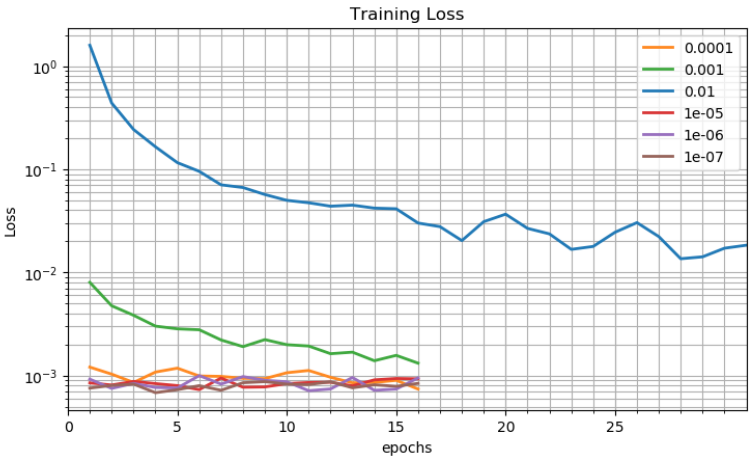


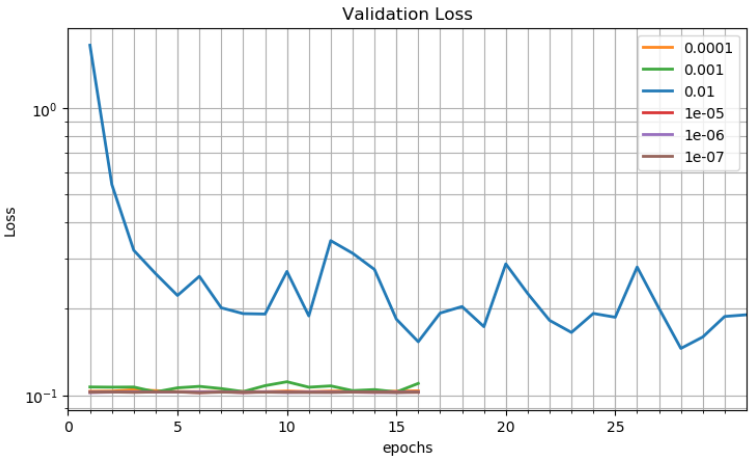


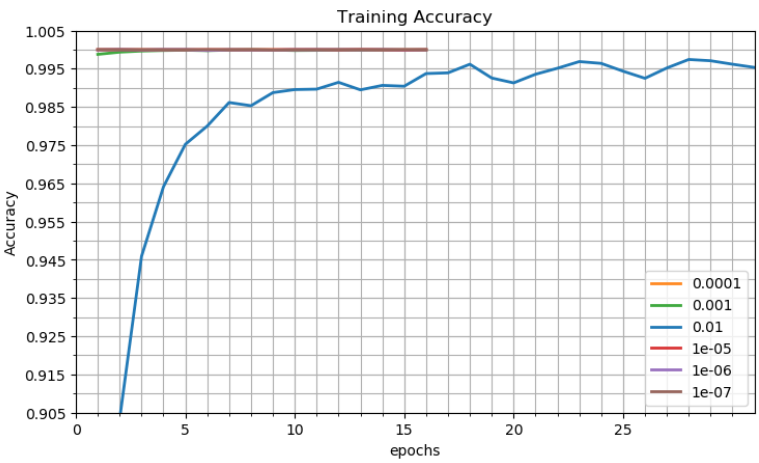


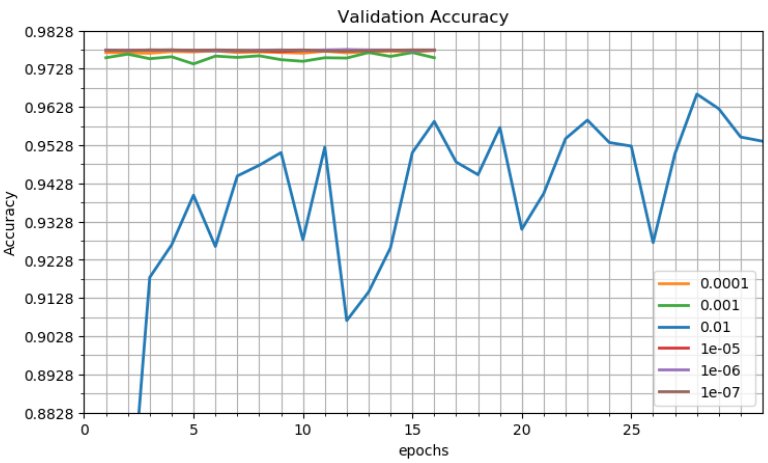


Modello 5 – Conv2D 64f 3x3

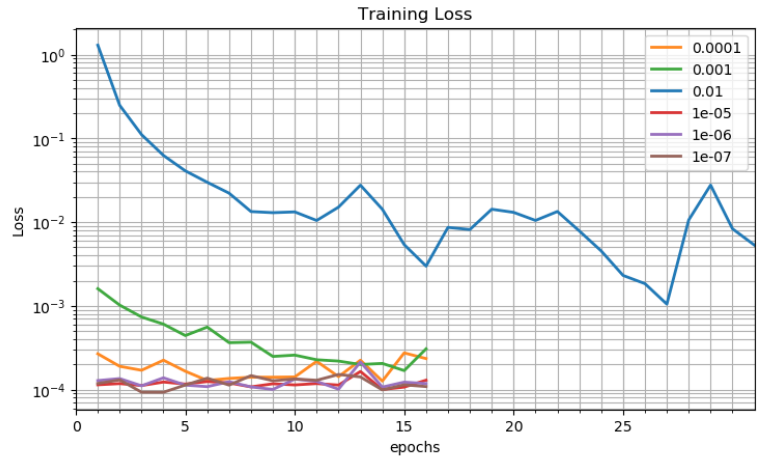


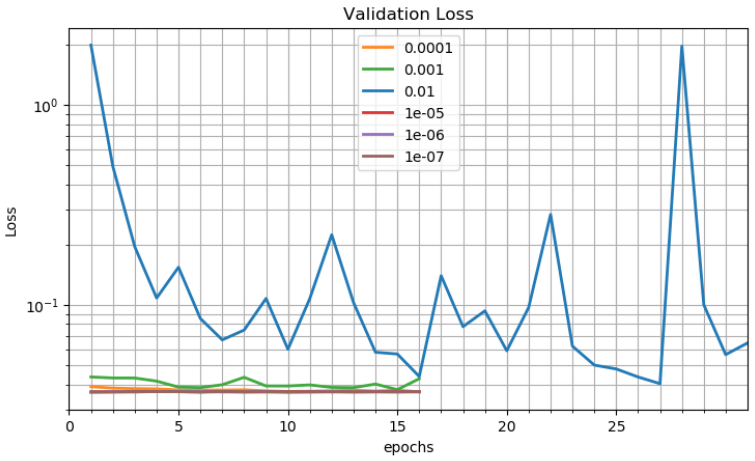


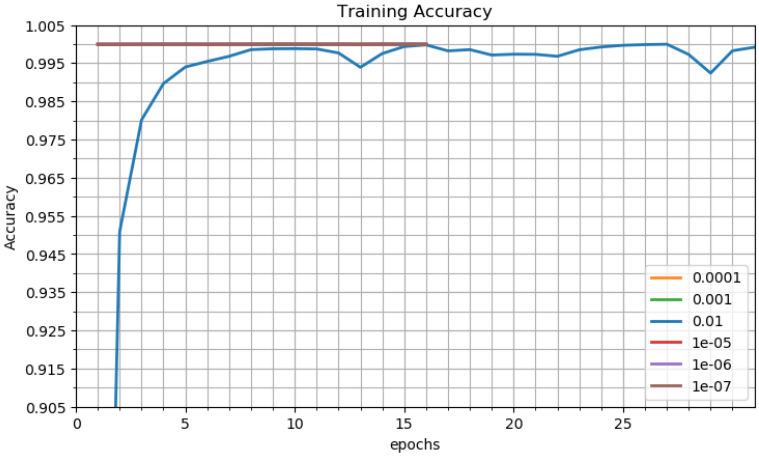


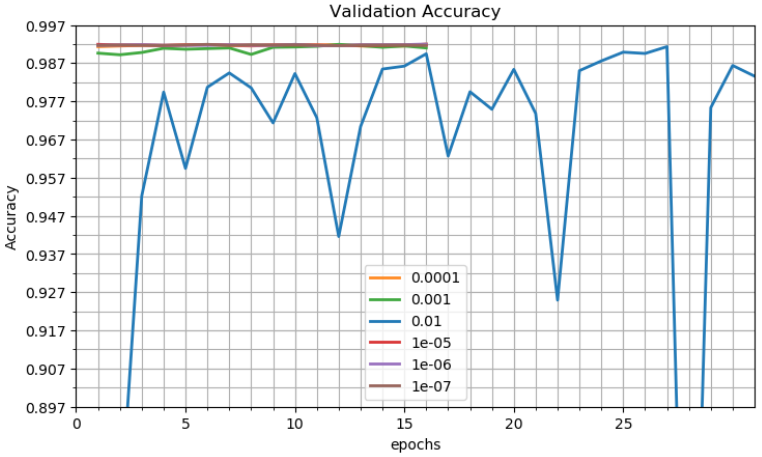


Modello 6 – Conv2D 64f 3x3 -> Conv2D 64f 3x3

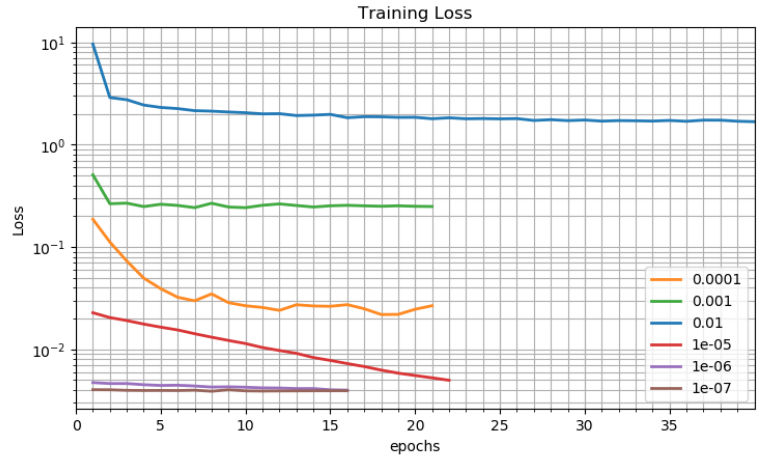


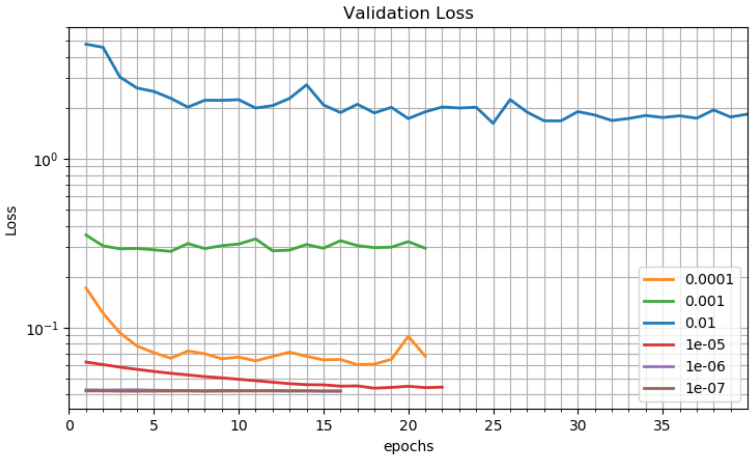


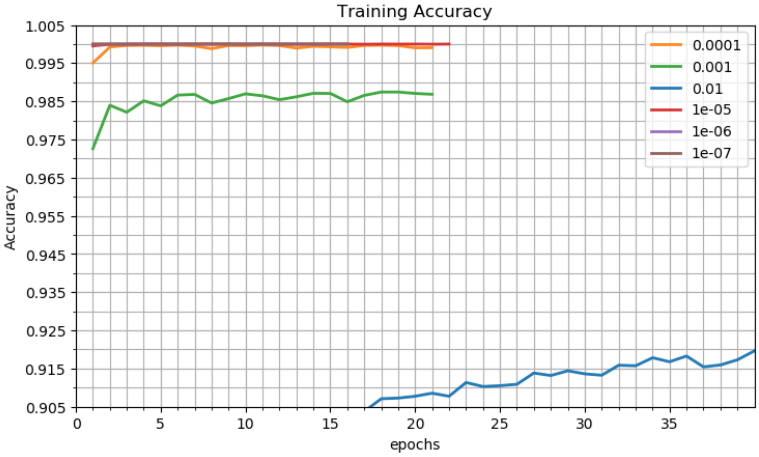


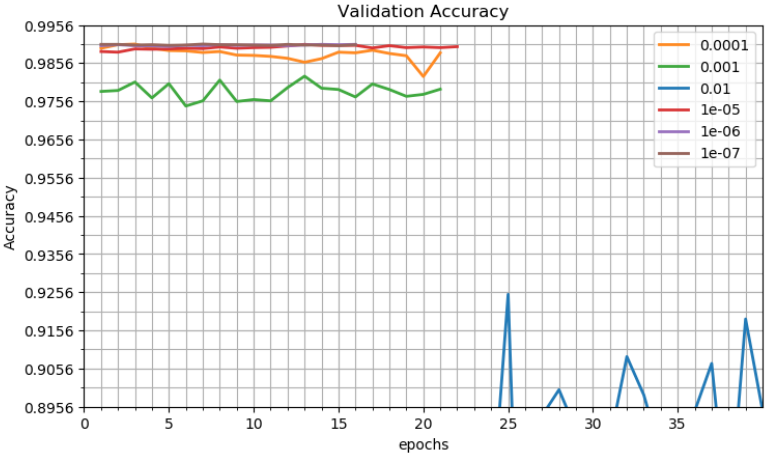


Modello 7 - Conv2D 64f 3x3 -> Conv2D 64f 3x3 -> Dense64

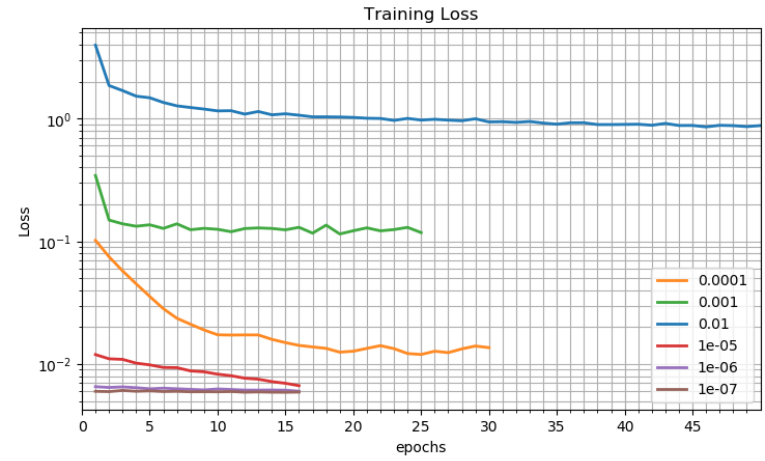


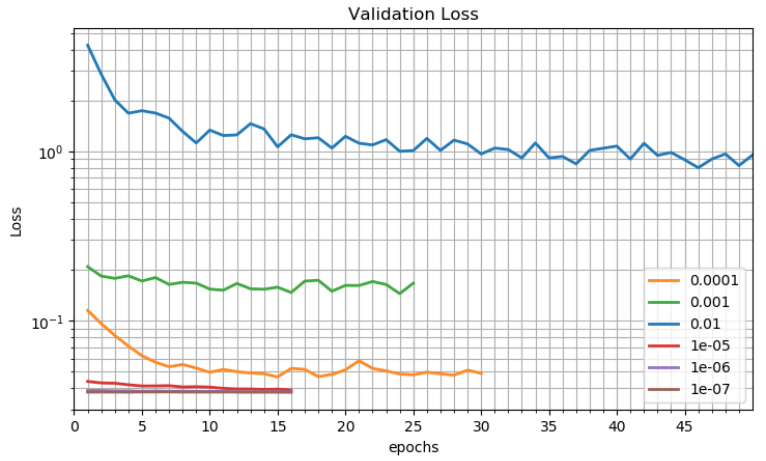


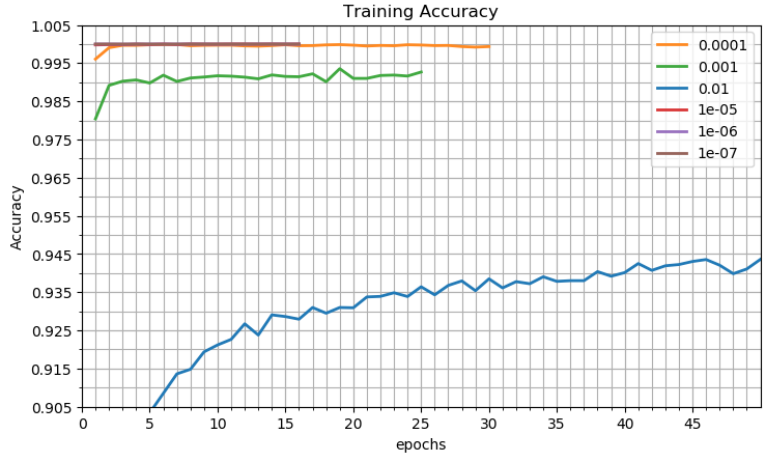


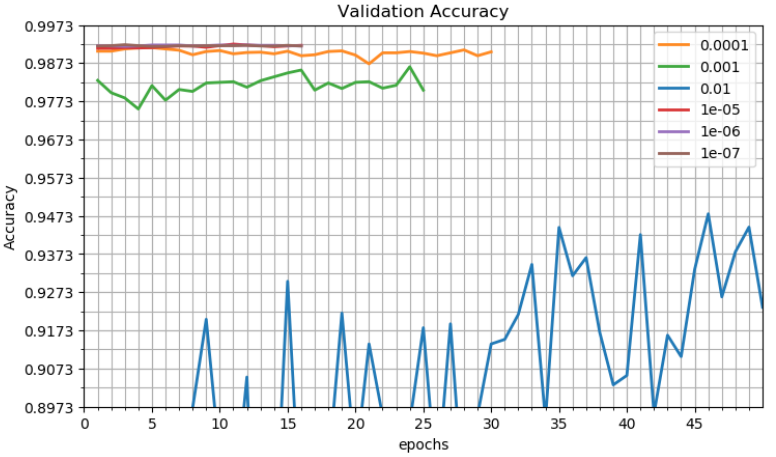


Modello 8 - Conv2D 64f 3x3 -> Conv2D 64f 3x3 -> MaxPooling -> Dense64

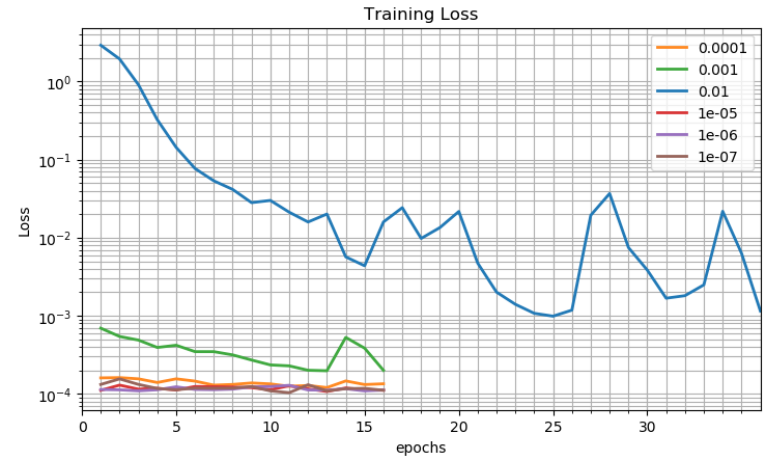


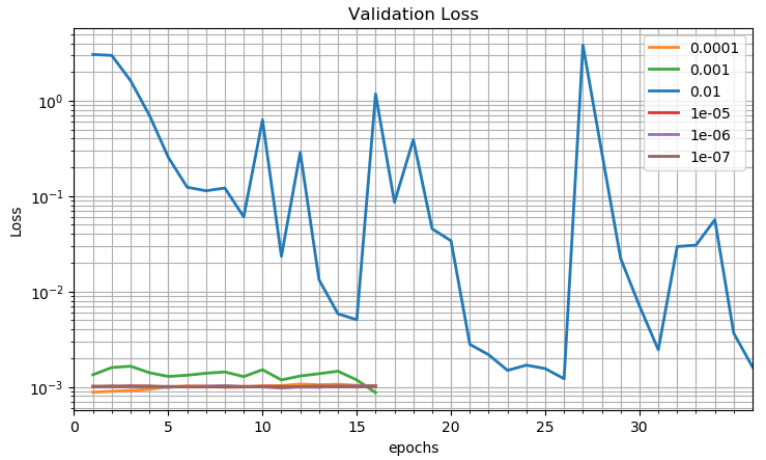


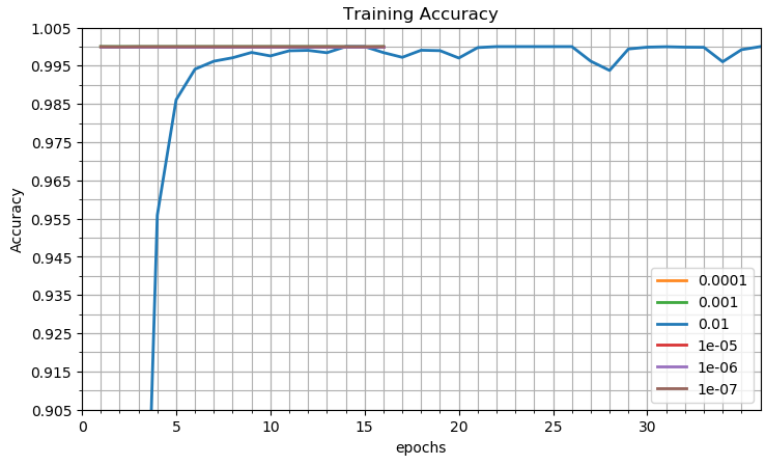


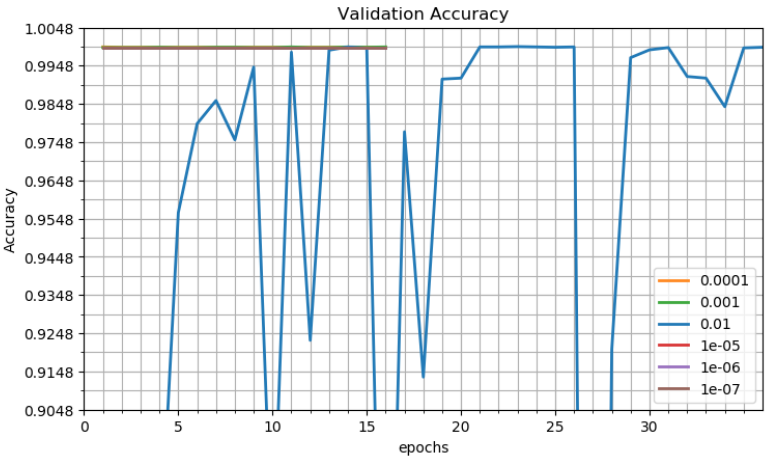


Modello 9 – 3 X Conv2D 64f 3x3 -> MaxPooling - > InceptionBlock -> Conv2D 43f 3x3 -> GlobalPooling

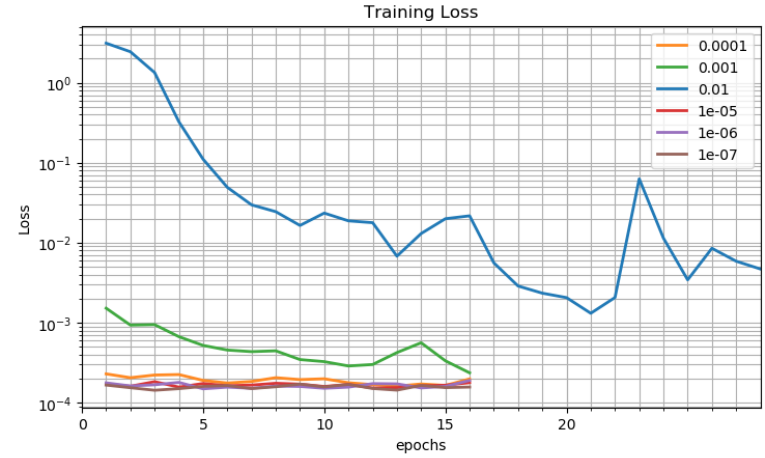


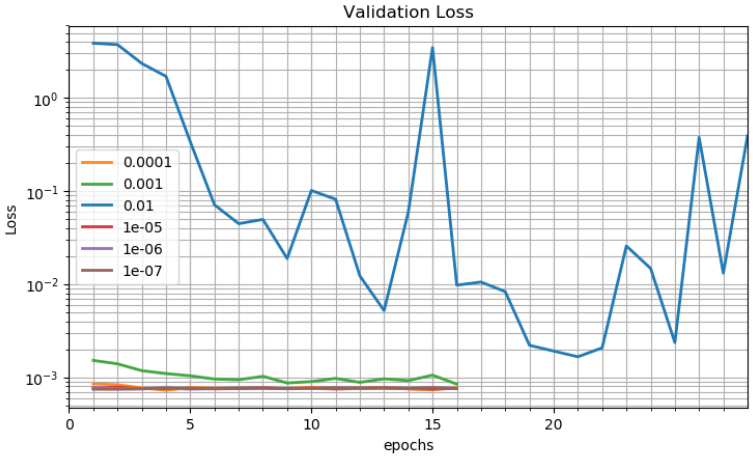


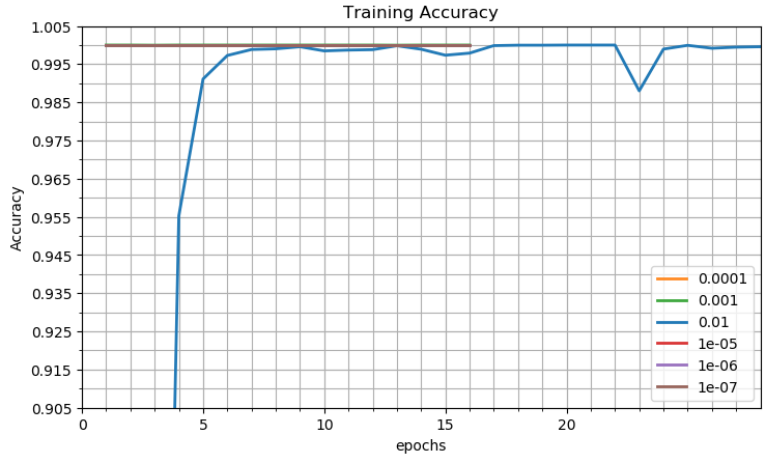


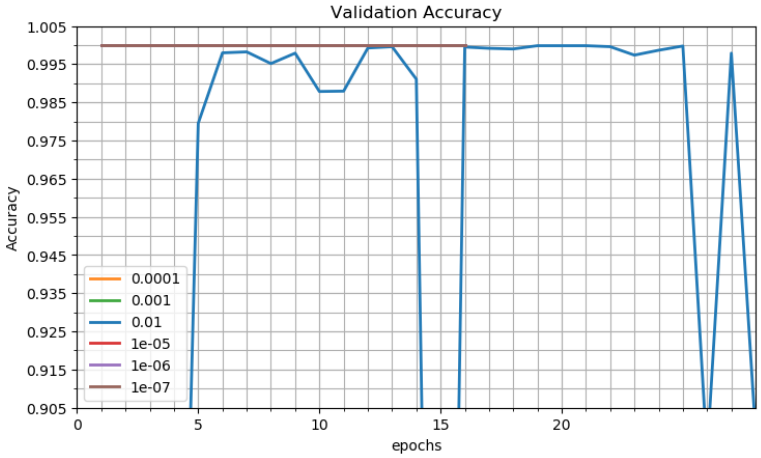


Modello 10 – 3 X Conv2D 64f 3x3 -> MaxPooling - > ResNetBlock -> Conv2D 43f 3x3 -> GlobalPooling









Riferimenti

[1] X. Z. S. R. J. S. Kaiming He, «Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition,» 2014.

[2] W. L. Y. J. P. S. S. R. D. A. D. E. V. V. A. R. Christian Szegedy, «Going deeper with convolutions,» 2014.

[3] X. Z. S. R. J. S. Kaiming He, «Deep Residual Learning for Image Recognition,» 2015.

[4] Q. C. S. Y. Min Lin, «Network In Network,» arXiv, 2013.

[5] V. V. S. I. J. S. Z. W. Christian Szegedy, «Rethinking the Inception Architecture for Computer Vision,» 2015.