RELAZIONE PROGETTO DI METODI QUANTITATIVI PER L'INFORMATICA

"Un approccio di Transfer Learning con Classification: analisi e confronto delle prestazioni di una semplice rete neurale convoluzionale ed una rete pre-trained fornita da TensorFlow"

Lorenzo Nicoletti – 1797464

Luca Polenta – 1794787

Data 12/07/2020

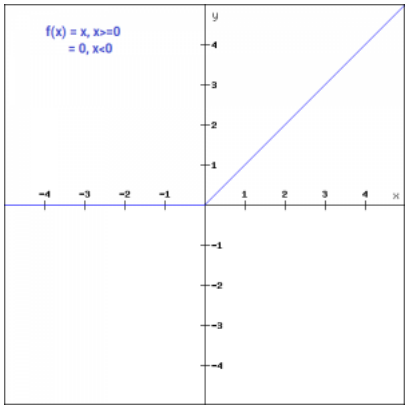
ABSTRACT

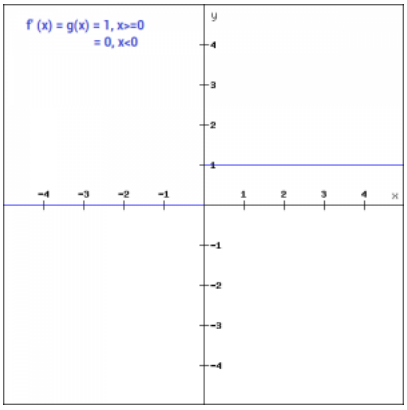
Il progetto ha come obiettivo la classificazione di un dataset di cartelli stradali, ponendo particolare attenzione all'analisi e al confronto delle prestazioni temporali in relazione ai risultati ottenuti da una rete pre-trained (MobileNetV2) importata da TensorFlow e da una semplice rete neurale convoluzionale sviluppata in completa autonomia da noi alunni del corso di Metodi Quantitativi Per L'Informatica.

PRELIMINARIES

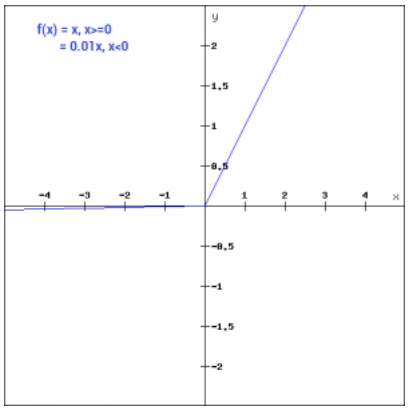
Le funzioni di attivazione usate nella semplice rete neurale convoluzionale sono la ReLU (Rectified Linear Unit), la LeakyReLU e la softMax. Quest'ultima in particolare è stata usata come funzione di attivazione nel layer finale fully-connected della rete neurale e verrà poi anche usata dalla funzione loss, la quale è illustrata successivamente. Le rispettive formule matematiche per le funzioni di attivazione sono:

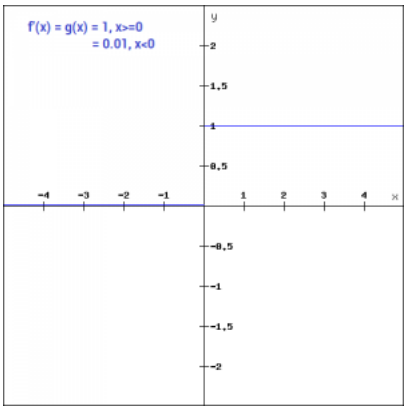
$$Relu: \qquad f(x) = \begin{cases} 0 & se\ x < 0 \\ x & se\ x > 0 \end{cases} \qquad f'(x) = \begin{cases} 0 & se\ x < 0 \\ 1 & se\ x > 0 \end{cases} \tag{1}$$



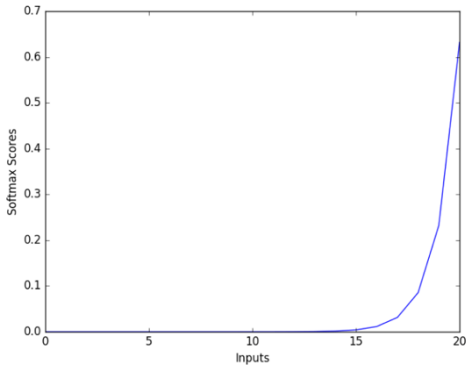


$$Leaky\ Relu: \qquad f(x) = \begin{cases} 0.01x & se\ x < 0 \\ x & se\ x > 0 \end{cases} \qquad f'(x) = \begin{cases} 0.01 & se\ x < 0 \\ 1 & se\ x > 0 \end{cases} \tag{1}$$





$SoftMax$: $$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}}$$ Dove s_i sono i punteggi inferiti dalla rete per la classe i-esima appartenente all'insieme delle classi C; (1)



Si è scelto di utilizzare le funzioni di attivazione ReLU e LeakyReLU sia per la loro semplicità di computazione, in quanto permettono di non appesantire eccessivamente la rete, e sia per la loro eccellente efficienza nei risultati. In particolare, si è deciso di utilizzarle entrambe per confrontare il caso della ReLU, che può portare alla completa disattivazione di alcuni nodi, con la LeakyReLU, che non disattiva mai completamente i nodi, lasciando che anche quelli meno utili al raggiungimento dell'obiettivo si addestrino in modo da poter dare un contributo positivo, se possibile, in fasi successive dell'addestramento. Invece, la softMax utilizzata nel layer finale è stata scelta in quanto rielabora egregiamente i logits che si ottengono come output dell'ultimo livello e li trasforma in distribuzioni di probabilità, normalizzandoli tra 0 e 1.

Inoltre, sia la rete pre-trained MobileNetV2 che la semplice rete convoluzionale viene usata come funzione per il calcolo della loss la "Categorical Cross-Entropy". Essa è spesso definita "Softmax Loss" in quanto è composta da una funzione di attivazione SoftMax sommata alla loss Cross-Entropy. La rispettiva formula matematica è:

$$Cross - Entropy: \quad CE = -\sum_i^C t_i * \log\left(f(s)_i\right) \tag{2}$$

Dove: $f(s)_i$ è la funzione di attivazione softMax descritta dalla formula (2): s_i sono i punteggi inferiti dalla rete per la classe i-esima appartenente all'insieme delle classi C; t_i è il vettore Target (vettore che contiene i veri output).

Invece, come ottimizzatore (optimizer) della rete pre-trained MobileNetV2 viene utilizzato "RMSProp". Quest'ultimo viene spesso utilizzato in quanto tenta di smorzare le oscillazioni verso l'ottimo rispetto al Momentum. Inoltre, l'RMSProp elimina la necessità di regolare il tasso di apprendimento, facendolo autonomamente, e sceglie una diversa velocità di apprendimento per ciascun parametro. La rispettiva formula matematica è:

$$Per\ ogni\ valore\ in\ \omega_j: \quad \omega_{t+1} = \omega_t + \Delta\omega_t \tag{3}$$

$$\Delta\omega_t = -\frac{\eta}{\sqrt{s_t+\epsilon}} * g_t \tag{3}$$

$$s_t = \beta_2 * s_{t-1} - (1 - \beta_2) * g_t^2 \tag{3}$$

Dove: η è il learning rate iniziale; s_t è la media esponenziale quadratica del gradiente lungo ω_j; g_t è il gradiente al tempo t di ω_j; β_2 è l'iperparametro che permette di regolare la dipendenza dell'RMSProp dallo stato precedente e dal calcolo del gradiente al quadrato; ϵ è una costante molto piccola per evitare di avere divisioni e radici per zero.

Infine, come ottimizzatore (optimizer) della semplice rete neurale convoluzionale è stato utilizzato "Adam", in quanto combina esaurientemente il Momentum, caratterizzato da una veloce e ottima ricerca del minimo, e l'RMSProp, che impedisce eccessive oscillazioni durante tale ricerca. La rispettiva formula matematica è:

$$Per\ ogni\ valore\ in\ \omega_j: \quad \omega_{t+1} = \omega_t + \Delta\omega_t \tag{3}$$

$$\Delta\omega_t = -\eta * \frac{v_t}{\sqrt{s_t+\epsilon}} * g_t \tag{3}$$

$$v_t = \beta_1 * v_{t-1} - (1 - \beta_1) * g_t \tag{3}$$

$$s_t = \beta_2 * s_{t-1} - (1 - \beta_2) * g_t^2 \tag{3}$$

Dove: η è il learning rate iniziale; v_t è la media esponenziale del gradiente lungo ω_j; s_t è la media esponenziale quadratica del gradiente lungo ω_j; g_t è il gradiente al tempo t di ω_j; $\beta_1 e\ \beta_2$ sono degli iperparametri che permettono di regolare la dipendenza di Adam rispettivamente dall'approccio del Momentum e dall'approccio dell'RMSProp; ϵ è una costante molto piccola per evitare di avere divisioni e radici per zero.

SELECTED DATASET

Il dataset utilizzato è costituito da un totale di 43 classi, ciascuna contenente un numero variabile di samples di dimensione fissa 32x32, che rappresentano vari tipi di cartelli stradali. Una sua caratteristica fondamentale consiste nella diversificazione all'interno di una stessa classe: segnali stradali simili sono stati acquisiti in momenti diversi della giornata, in modo da avere alcuni samples ritratti al buio, in condizioni di sovra-illuminazione o in condizioni atmosferiche avverse ad un loro immediato riconoscimento. In aggiunta, sono presenti dei samples per ogni classe in cui è stata particolarmente accentuata la forma e le caratteristiche peculiari di ogni cartello tramite l'applicazione di una

colorazione a contrasto blu/bianco e l'equalizzazione di ogni bordo dei soggetti in esame. Il dataset è sembrato dunque appropriato per il task prefissato, coniugando una non sempre facile identificazione dei segnali con un alto numero di classi a disposizione.

Durante il confronto prestazionale tra le due reti utilizzate, il dataset evolve a seconda del caso studiato. In particolare, sono stati analizzati tre differenti contesti d'azione:

1. "**10RandomClasses**", composto da dieci classi scelte casualmente e pertanto di differente densità. Quest'ultimo presenta 25939 samples totali per il train set, 4573 samples totali per la validation set e 3209 samples totali per il test set;
2. "**10PoorestClasses**", a partire dalle 43 classi totali presenti nel dataset di origine, sono state estratte quelle che presentano il minor numero di samples al loro interno, in quanto un minor numero di samples ne rende più difficoltoso l'addestramento della rete. Quest'ultimo presenta 3640 samples totali per il train set, 635 samples totali per la validation set e 706 samples totali per il test set;
3. "**43Classes**", dataset acquisito nella sua totalità. Quest'ultimo presenta 54658 samples totali per il train set, 9623 samples totali per la validation set e 8858 samples totali per il test set.

Per riportare con precisioni alcune informazioni peculiari del dataset completo "43Classes", è stata stilata la seguente tabella, la quale racchiude il relativo numero di ogni classe, il nome associato alla classe in questione e il numero di samples per ogni classe nei set di training, validation e test. Il numero dei samples nel validation set è calcolato come il 15% del numero totale di elementi nel training set (arrotondato per difetto), e quindi nella colonna del numero di samples del training set è riportato il restante 85% (arrotondato per eccesso). Inoltre, sono state evidenziate delle classi per segnalare anche quali sono presenti nei dataset "10RandomClasses" e "10PoorestClasses":

Legenda:

- Giallo: classe presente in "10RandomClasses"
- Verde: classe presente in "10PoorestClasses"
- Arancione: classe presente sia in "10RandomClasses" che in "10PoorestClasses"

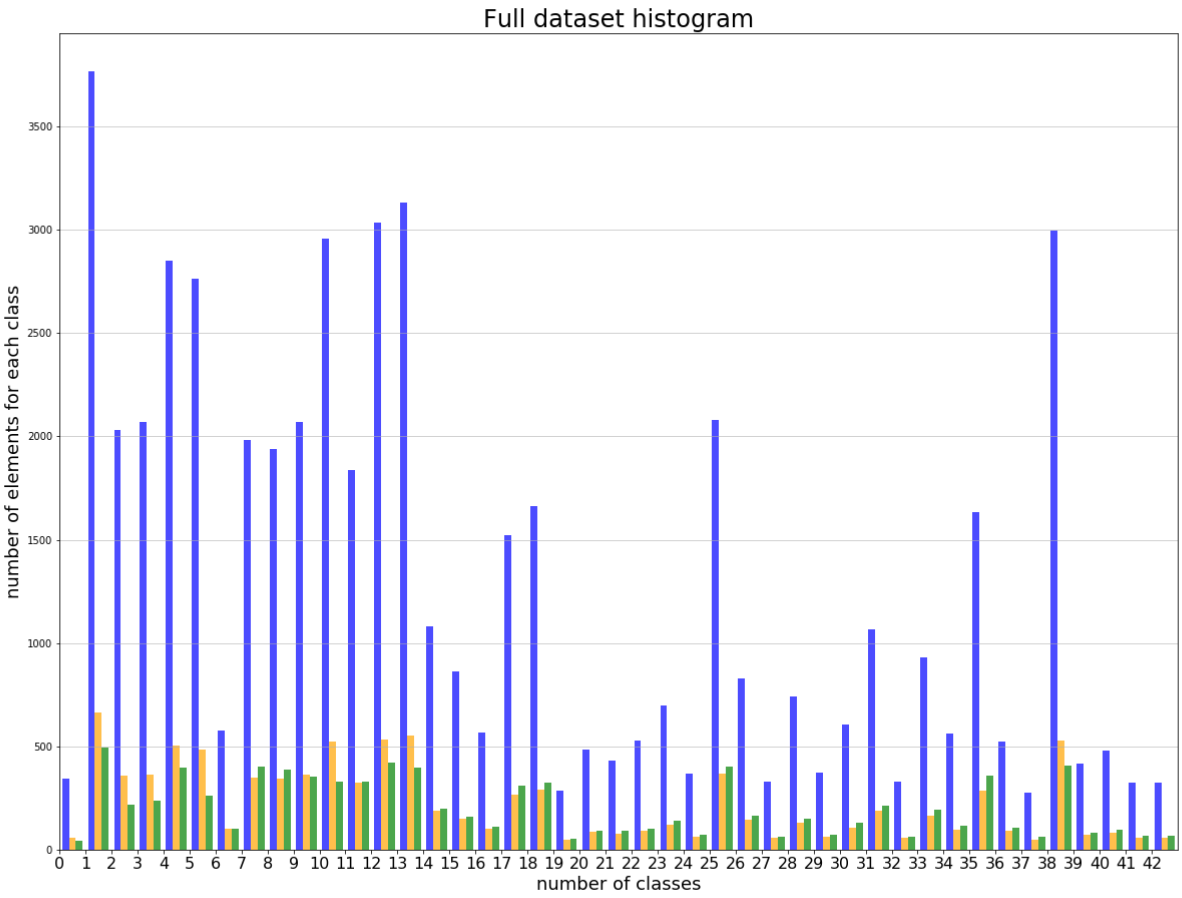
NUM CLASSE	NOME CLASSE	#TRAINING SET	#VALIDATION SET	#TEST SET
0	Speed limit (20km/h)	346	60	46
1	Speed limit (30km/h)	3765	664	492
2	Speed limit (50km/h)	2034	358	219
3	Speed limit (60km/h)	2070	365	237
4	Speed limit (70km/h)	2850	502	400
5	Speed limit (80km/h)	2993	487	260
6	End of speed limit (80km/h)	578	101	103

7	Speed limit (90km/h)	1982	349	407
8	Speed limit (100km/h)	1939	342	391
9	No passing	2071	365	355
10	No passing for vechiles over 3.5 metric tons	2109	372	330
11	Right-of-way at the next intersection	1837	324	330
12	Priority road	3036	535	420
13	Yield	2994	528	399
14	Stop	1080	190	201
15	No vehicles	862	151	158
16	Vehicles over 3.5 metric tons prohibited	568	100	114
17	No Entry	1523	268	310
18	General caution	1665	293	324
19	Dangerous curve to the left	288	50	53
20	Dangerous curve to the right	486	85	91
21	Double curve	432	76	93
22	Bumpy road	526	92	102
23	Slippery road	699	123	140
24	Road narrows on the right	370	65	76
25	Road work	2080	367	405
26	Traffic signals	830	146	166
27	Pedestrians	386	57	66
28	Children crossing	714	130	150
29	Bicycles crossing	373	65	72
30	Beware of ice/snow	605	106	129
31	Wild animals crossing	1069	188	214
32	End of all speed and passing limits	329	57	66
33	Turn right ahead	932	164	194
34	Turn left ahead	564	99	119
35	Ahead only	1636	288	357
36	Go straight or right	522	92	107
37	Go straight or left	278	49	65
38	Keep right	2994	528	410
39	Keep left	416	73	83
40	Roundabout mandatory	480	84	98
41	End of no passing	327	57	67
42	End of no passing by vehicles over 3.5 metric tons	328	57	67

Oss.: le classi appartenenti anche ai dataset “10RandomClasses” e “10PoorestClasses” presentano un numero di classe diverso rispetto alla tabella qui sopra riportata, in quanto sono state rinumerate in ordine crescente e continuo, partendo dallo 0, in base al loro numero nel dataset generale.

In aggiunta è possibile visionare anche l’istogramma delle densità delle classi. In blu è riportata la

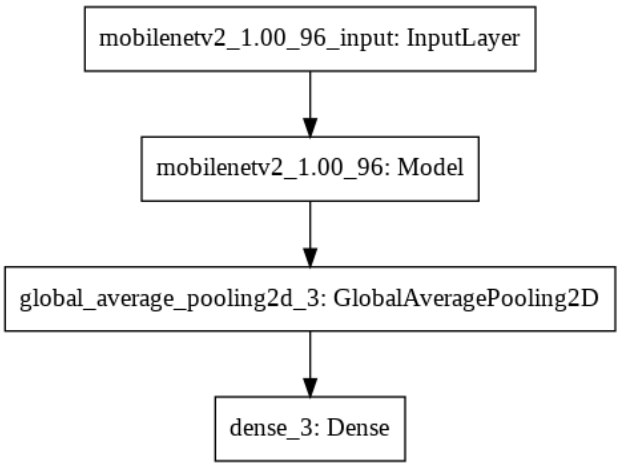
densità del training set, in giallo la densità del validation set e in verde la densità del test set:



METHOD

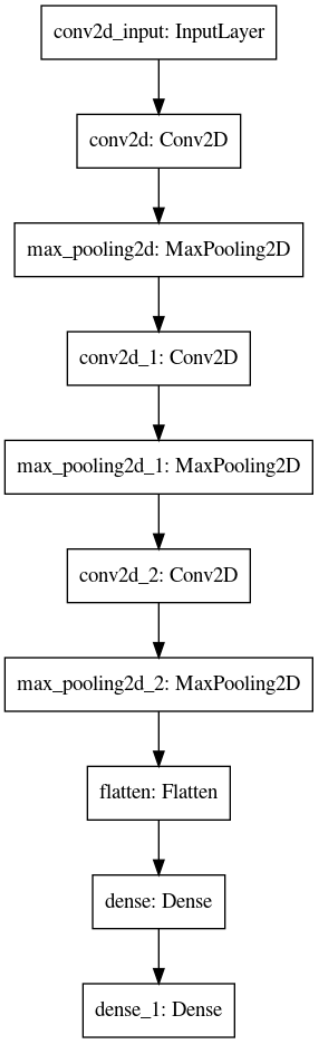
Avendo due reti distinte, i metodi sono diversi per ciascuna.

Come rete pre-trained è stata scelta MobileNetV2, la cui rappresentazione grafica della struttura è riportata nell'immagine qui a destra. Essa presenta un elevato quantitativo di layers definiti nel "mobilenetv2_1.00_96: Model" che alla fine convergono in due layers con le seguenti proprietà: il primo è il layer di keras "GlobalAveragePooling2D", il quale riduce la quantità di informazione totale eseguendo una media dei risultati convoluzionali calcolati dai layers precedenti tramite un kernel di dimensione 5x5, facendo infine convergere i dati



in un singolo vettore di 1280 elementi per immagine; il secondo e ultimo layer è un fully-connected di keras “Dense”, il quale converte le features nel numero di previsioni corrispondenti al numero di classi del dataset in esame. In questo layer è necessaria la funzione di attivazione softMax che trasformi i logits del layer precedente in valori di probabilità e che preveda poi la classe associata ad ogni immagine in base alla probabilità più alta stimata tra tutte. Infine, per ulteriori specifiche riguardo la struttura interna della rete MobileNetV2, rimandiamo al punto (4) delle references dove è presente un link al sito di tensorflow nel quale è disponibile una guida che illustra ulteriori caratteristiche della rete.

Invece, la struttura della semplice rete convoluzionale, chiamata PersonalCNN per semplicità, è composta da tre layers convoluzionali bidimensionali (Conv2D) alternati da tre layers di max pooling bidimensionali (MaxPooling2D) di dimensione 2x2. Tutti i layer Conv2D sono composti da 32 nodi e possono presentare come funzione di attivazione la “ReLU” (Rectified Linear Unit) o la “LeakyReLU”. Il primo layer Conv2D ha un kernel di dimensione 7x7, il secondo di dimensione 5x5 e il terzo di dimensione 3x3. Le dimensioni dei kernel nei layers diminuiscono gradualmente poiché è stato notato empiricamente come tale struttura tenda ad ottenere risultati più ottimi rispetto ad altre situazioni. Inoltre, il primo layer specifica la dimensione delle immagini che riceve in input, cioè 32x32 in 3 colori (RGB). Dopo il terzo layer di MaxPooling2D vi è un layer Flatten che permette di passare dalla struttura tridimensionale della matrice nello strato precedente ad un vettore monodimensionale che rappresenti ugualmente tutti i dati. Questo viene fatto perché i successivi due e ultimi layers sono fully-connected (Dense) che accettano in input solo vettori monodimensionali. Il primo dei due è costituito da 512 nodi e può adottare come funzione di attivazione sia la “ReLU” che la “LeakyReLU”. L’ultimo layer è formato da un numero di nodi pari al numero di classi e come funzione di attivazione utilizza la funzione softMax che trasforma i logits del layer precedente in valori di probabilità e che prevede poi la classe associata ad ogni immagine in base alla probabilità più alta stimata tra tutte.



La rete PersonalCNN è stata creata nella forma sopracitata per avere una rete dalla struttura efficiente e leggera, che fosse all’altezza dell’obiettivo di questo progetto. La rete MobileNetV2, invece, è stata scelta in quanto lo scopo di questo progetto è proprio confrontare la PersonalCNN con una rete pre-trained che fosse creata per lo stesso scopo e che al contempo fosse il più possibile leggera, proprio come lo è la rete PersonalCNN.

IMPLEMENTATION

In primis occorre notare che la rete pre-trained non ha subito modifiche rispetto alle specifiche standard disponibili online.

I dataset invece presentano già una parziale augmentation sia nel quantitativo di immagini che per quanto riguarda il contenuto di ogni sample. Sono presenti immagini di segnali stradali acquisite in momenti diversi della giornata, in modo da avere alcuni samples ritratte al buio, in condizioni di sovra-illuminazione o in condizioni atmosferiche avverse ad un loro immediato riconoscimento. In aggiunta sono presenti dei samples per ogni classe in cui è stata particolarmente accentuata la forma e le caratteristiche peculiari di ogni cartello tramite l'applicazione di una colorazione a contrasto blu/bianco e l'equalizzazione di ogni bordo dei soggetti in esame. Oltre a tali augmentation si è deciso poi di eseguire le ulteriori seguenti modifiche:

- shear_range: 0.1 nel test e 0.15 nel train/validation, che permette di inserire distorsione angolare per deformare leggermente l'immagine, simulando situazioni in cui i cartelli non sono perpendicolari alla strada, ma si presentano lievemente obliqui. Può rappresentare un caso realistico in cui i ganci lenti dei cartelli hanno permesso una lieve rotazione di quest'ultimi, per esempio per via del vento;
- zoom_range: 0.1 nel test e 0.15 nel train/validation. Aggiunge dello zoom nelle immagini. Può rappresentare situazioni in cui ci si trova molto vicino ai cartelli e quindi vengono ingranditi e/o tagliati rispetto ai margini degli scatti;
- width_shift_range: 0.1 nel test e 0.15 nel train/validation. Sposta l'immagine orizzontalmente, facendo valutare casi in cui l'immagine stessa venga tagliata a destra o a sinistra rispetto ai margini degli scatti;
- height_shift_range: 0.1 nel test e 0.15 nel train/validation. Sposta l'immagine verticalmente, facendo valutare casi in cui l'immagine stessa venga tagliata sopra o sotto rispetto ai margini degli scatti;
- fill_mode: è stata impostata a constant cosicché, una volta eseguite le precedenti distorsioni, le aree esterne vengano riempite con il colore nero. È sembrato il caso più realistico in quanto con l'opzione "nearest" di default gli spazi in eccesso sarebbero stati riempiti con l'immagine stessa specchiata, rappresentando dunque un caso poco realistico.

Sono state invece rigettate, e di conseguenza disattivate, alcune augmentation:

- vertical_flip: è impostata a false in quanto in cartelli non possono trovarsi invertiti verticalmente.
- horizontal_flip: è impostata a false in quanto i cartelli non possono trovarsi invertiti orizzontalmente e, in alcuni casi, possono persino cambiare significato, e quindi classe, come nel

caso dell'esempio riportato successivamente;

- rotation_range: è stata disattivata in quanto non sono presenti cartelli roteati. Essi infatti sono fissati verticalmente da due o più ganci ad un palo verticale, che quindi ne potrebbe permettere al massimo solo uno scorrimento verticale. Inoltre, ruotare i cartelli potrebbe far cambiare significato, e quindi classe, ai medesimi, come nel prossimo esempio degli indicatori di corsia:

Un'ultima fase di pre-processing ha coinvolto le immagini del dataset nel caso in cui siano utilizzate nella rete MobileNetV2. Le immagini del dataset hanno una dimensione di default di 32x32, ma la rete MobileNetV2 richiede che siano in una delle 5 dimensioni specifiche su cui la rete è stata precedentemente addestrata e per le quali esistono dei pesi specifici che la rete reperisce da "imagenet". Le dimensioni richieste sono 96x96, 128x128, 160x160, 192x192 o 224x224. Per arrivare ad una delle suddette dimensioni viene svolta una resize con interpolazione cubica per ogni immagine del dataset.

EXPERIMENTS / EXPERIMENTAL SETUP / EXPERIMENT PROCEDURES

Preliminarmente, è importante far notare come la PersonalCNN è in grado di essere addestrata offline abbastanza velocemente sia in GPU che in CPU. Con le convoluzioni della rete impostate a 32 nodi la GPU ha richiesto circa un secondo ad epoca, mentre in CPU all'incirca 20-30. Alzando i nodi a 64, la GPU è rimasta sugli 1-2 secondi, mentre la CPU a circa 40-50 secondi. Questi risultati hanno indotto ad addestrare principalmente con la GPU (NVIDIA).

Invece, l'addestramento della rete MobileNetV2 è risultato troppo lento in CPU. Settando i parametri nella maniera più semplice e leggera possibile, quest'ultima rete ha richiesto oltre 2 ore di addestramento in CPU, rispetto ai pochi secondi o minuti impiegati dalla GPU, quindi è stata sempre utilizzata sempre quest'ultima impostazione per eseguire i vari test d'addestramento. Occorre però specificare che la GPU NVIDIA a disposizione offline è risultata insufficiente per supportare tale lavoro, pertanto l'addestramento di tale rete è stato spostato sul servizio COLAB, che mette a disposizione delle GPU sufficientemente potenti e adeguate al compito richiesto.

Per quanto riguarda invece la fase di sperimentazione del progetto, essa è distinta in due fasi:

- ottimizzazione delle reti singolarmente;
- confronto delle reti risultanti in casistiche comuni.

Solo successivamente verranno svolte e confrontate casistiche comune, come l'augmentation. Per l'appunto, infatti, l'augmentation sarà disattivata di default durante l'ottimizzazione delle due reti separatamente

L'ottimizzazione della MobileNetV2 è proceduta per i seguenti punti:

1. Ricerca del valore di learning rate ottimo in base alla presenza o meno dell'early stopping;
2. Ricerca del numero di epoche adeguate;
3. Analisi dei risultati nel caso in cui il base model venga addestrato o meno;
4. Individuazione della resize più adeguata tra quelle disponibili;
5. Individuazione del miglior optimizer del modello.

Tuttavia, è necessario riportare che una delle principali problematiche affrontate durante tali test sulla rete MobileNetV2 è stata la gestione delle risorse a disposizione, più specificatamente della loro saturazione. Nonostante l'ausilio di COLAB, il ridimensionamento con interpolazione, che occorre applicare alle immagini per poter sfruttare le potenzialità della rete MobileNetV2, porta facilmente alla saturazione della memoria RAM. Questo fenomeno si presenta quando si cerca di ridimensionare il dataset "43Classes" in ognuno delle dimensioni disponibili e quando si cerca di ridimensionare il dataset "10RandomClasses" oltre la dimensione 160x160 (compresa). Tentando di creare delle versioni alternative del ridimensionamento con interpolazione che richiedessero meno memoria RAM, il problema originario non è stato risolto. Onde evitare l'esaurimento delle risorse, è stato necessario creare un metodo a concatenazione, ma che, all'aumentare delle immagini ridimensionate, tende a divenire sempre più lento computazionalmente in quanto di volta in volta effettua una copia dell'intero dataset già ridimensionato. Per ovviare a tale situazione, le analisi condotte sul dataset "43Classes" con la rete MobileNetV2 saranno esclusivamente condotte dal solo punto di vista teorico. Per quanto invece riguarda il dataset "10RandomClasses", le analisi con la rete MobileNetV2 saranno effettuate solo per ridimensionamenti inferiori a 128 (incluso), compensando i test per ridimensionamenti mancanti con delle analisi teoriche.

Infine, in riferimento ai test sopracitati, si è deciso di iniziare il primo test analizzando in primis solo il dataset "10RandomClasses" con i seguenti valori di default:

- Addestramento del base model: SI
- Dimensione della resize: 96

È stato scelto "10RandomClasses" come dataset di partenza in quanto è il dataset di riferimento principale e i valori sopra mostrati sono stati scelti in quanto consigliati dalle linee guida della rete pre-trained o per valutare casistiche abbastanza semplici. Successivamente anch'essi verranno cambiati e messi in discussione, con l'obiettivo di individuare le impostazioni adeguate a ottenere le massime

prestazioni della rete in tutte le casistiche.

L'ottimizzazione della PersonalCNN, invece, è proceduta per i seguenti punti:

1. Ricerca del miglior numero di nodi per i layers convoluzionali;
2. Ricerca del miglior numero di nodi per il penultimo layer fully-connected;
3. Analisi dell'early stopping;
4. Confronto tra i risultati delle funzioni di attivazione "ReLU" e "LeakyReLU";
5. Individuazione del miglior optimizer del modello.

Successivamente si sono confrontati i risultati ottenuti dalle due reti nei seguenti casi:

1. Train e Test augmentation non presente: questo rappresenta il caso più semplice dove i cartelli non sono distorti in nessuna maniera;
2. Train augmentation non presente e Test augmentation presente: questo rappresenta il caso in cui la rete non è stata addestrata a casi più realistici, che poi però si presentano nei test che rappresentano casi reali;
3. Train augmentation presente e Test augmentation non presente: questo rappresenta il caso in cui la rete viene addestrata alle possibili variazioni, ma che poi non si presentano nei test che rappresentano casi reali;
4. Train e Test augmentation presente: questo rappresenta il caso più reale dove i cartelli sono distorti sia durante l'addestramento che nel test set.

Per quanto riguarda delle impostazioni comuni alle due reti si vuol riportare all'attenzione del lettore che il set di validation è stato ottenuto come il 15% del train set, durante la generazione dei set tramite l'utility imageDataGenerator di TensorFlow. Inoltre, riguardo la dimensione del batch di addestramento, quest'ultima è stata impostata a 100 in modo da regolarizzarne la grandezza, velocizzando il calcolo e avendo allo stesso tempo un buon numero di immagini per epoca. È presente poi una seconda batch size, adottata riguarda l'ImageDataGenerator di TensorFlow: per una più semplice acquisizione del dataset nella forma <X, y> è stata impostata pari al numero di samples per dataset, in modo da consentirne una più immediata manipolazione.

Occorre precisare, infine, che i risultati riportati nella sezione seguente spesso presenteranno più rilevazioni per un singolo caso. Questo perché spesso è stato deciso di effettuare multipli test per accertare alcuni risultati prima di confermarli. Tali rilevazioni multiple sono divise dal separatore '/'.

RESULTS

Per quanto riguarda la ricerca della configurazione ottima della rete MobileNetV2, l'esperimento ha

seguito le fasi precedentemente esposte:

1. Ricerca del valore di learning rate ottimo in base alla presenza o meno dell'early stopping, con i seguenti parametri di default:
    a. Addestramento del base model: SI
    b. Dimensione della resize: 96
    c. Train e Test Augmentation: NO

DATASET	ACCURACY	LOSS	MAX EPOCHE	EARLY STOPPOING	TEMPO	LEARNING RATE
10Random	0.6494	1.8072	5	NO	2m 50s	0.0001
10Random	0.7644 / 0.7454	1.6954 / 1.7143	5	SI - TUTTE	2m 57s	0.0001
10Random	0.7504	1.7096	5	NO	2m 45s	0.0002
10Random	0.8199 / 0.8295	1.6392 / 1.6324	5	SI -TUTTE	2m 55s	0.0002
10Random	0.7214	1.7387	5	NO	2m 49s	0.0005
10Random	0.5509	1.9084	5	SI – 3 EPOCHE	1m 49s	0.0005
10Random	0.4637	1.9956	5	NO	2m 45s	0.001
10Random	0.3630	2.0976	5	SI – 3 EPOCHE	1m 47s	0.001
10Random	0.7149	1.7496	5	NO	2m 45s	0.00005
10Random	0.7647	1.7158	5	SI - TUTTE	2m 55s	0.00005

Una prima considerazione che si può evincere da questi test è che quando il valore di learning rate è pari a 0.00005, la rete non sembra ottenere prestazioni ottime come in altri casi, pertanto verrà scartato. Per i valori di learning rate 0.0005 e 0.001 invece l'early stopping termina l'addestramento 2 epoche prima e ottiene un risultato più basso di quanto già non ottenga quando non è presente. Come risaputo, l'early stopping è un metodo che ci permette di evitare l'overfitting analizzando l'andamento della loss e scegliendo poi di conseguenza i valori ottimi prima di raggiungere l'overfitting per via di una divergenza o di un addestramento forzato quando ci si trova in un plateau. Considerando i risultati scarsi e che nei due casi senza early stopping probabilmente si è verificato dell'overfitting, sono stati scartati entrambi i valori. Invece, per quanto riguarda i valori di learning rate 0.0001 e 0.0002, l'early stopping non ha fermato in anticipo l'addestramento, e in aggiunta è riuscito ad estrarre un risultato migliore di accuracy e di loss al termine dell'addestramento. Sono stati anche effettuati dei doppi test per appurare con maggior certezza i risultati, e pertanto da ciò si possono evincere due considerazioni: in primis si può notare come l'early stopping sembri fondamentale, non solo per evitare overfitting, ma anche per carpire il meglio dall'addestramento in sé per sé; in secundis attualmente è impossibile individuare il

valore ottimo di learning rate in quanto entrambi portano a risultati simili. Pertanto, al fine di scegliere il miglior valore di learning rate, bisogna effettuare un secondo test che ha l'obiettivo di valutare l'evoluzione delle accuracy all'aumentare delle epoche.

2. Ricerca del numero di epoche adeguate, con i seguenti parametri di default:
    a. Addestramento del base model: SI
    b. Dimensione della resize: 96
    c. Train e Test Augmentation: NO

DATASET	ACCURACY	LOSS	MAX EPOCHE	EARLY STOPPOING	TEMPO	LEARNING RATE
10Random	0.6494	1.8072	5	NO	2m 50s	0.0001
10Random	0.7644 / 0.7454	1.6954 / 1.7143	5	SI - TUTTE	2m 57s	0.0001
10Random	0.9455	1.5157	10	NO	4m 22s	0.0001
10Random	0.9240 0.9480	1.5395 1.5147	10	SI - TUTTE	4m 35s	0.0001
10Random	0.9776	1.4832	15	NO	6m 29s	0.0001
10Random	0.9579	1.5041	15	SI – 12 EPOCHE	5m 26s	0.0001
10Random	0.7504	1.7096	5	NO	2m 45s	0.0002
10Random	0.8199 / 0.8295	1.6392 / 1.6324	5	SI -TUTTE	2m 55s	0.0002
10Random	0.9797	1.4819	10	NO	4m 21s	0.0002
10Random	0.8454	1.6151	10	SI – 6 EPOCHE	2m 44s	0.0002
10Random	0.9691	1.4909	15	NO	6m 29s	0.0002
10Random	0.9215	1.5399	15	SI – 14 EPOCHE	6m 20s	0.0002
10Poorest	0.5085	1.9752	5	NO	53s	0.0001
10Poorest	0.4136	2.0544	5	SI - TUTTE	1m 11s	0.0001
10Poorest	0.4589	1.9878	10	NO	1m 32s	0.0001
10Poorest	0.4278	2.0477	10	SI – 6 EPOCHE	1m 21s	0.0001
10Poorest	0.6048	1.8676	15	NO	2m 12s	0.0001
10Poorest	0.5071	1.9632	15	SI – 7 EPOCHE	1m 33s	0.0001
10Poorest	0.4745	1.9965	5	NO	53s	0.0002
10Poorest	0.6289	1.8780	5	SI - TUTTE	1m 10s	0.0002
10Poorest	0.5071	1.9509	10	NO	1m 33s	0.0002
10Poorest	0.4547	2.0343	10	SI – 3 EPOCHE	47s	0.0002
10Poorest	0.3088	2.1523	15	NO	2m 12s	0.0002

10Poorest	0.5184	1.9485	15	SI – 4 EPOCHE	58s	0.0002

Analizzando i risultati ottenuti con il dataset "10RandomClasses", si osserva che quando il learning rate è impostato a 0.0001, l'early stopping non ferma l'addestramento né a 5 e né a 10 epoche. Quando invece sono impostate quindi 15 epoche l'early stopping interrompe l'addestramento anticipatamente. Eseguendo comunque l'addestramento a 15 epoche senza early stopping si nota un risultato leggermente migliore, ma questo potrebbe essere un falso risultato in quanto l'addestramento potrebbe essere andato in overfitting. Quando invece il learning rate è impostato a 0.0002, l'early stopping ferma l'addestramento sia a 10 che a 15 epoche. Inoltre, i risultati ottenuti rispettivamente con le medesime epoche e senza early stopping sono più alti, ma come nel caso precedente, potrebbero semplicemente essere overfitting. Le stesse identiche considerazioni si possono ottenere dall'analisi del caso con il dataset "10PoorestClasses". Pertanto, onde evitare overfitting, è quindi necessario mantenere sempre presente l'early stopping per tutti i futuri test che verranno svolti. Invece, per quanto riguarda il valore ottimo da assegnare al learning rate, quello che ottiene i risultati migliori a parità di epoche è il learning rate a 0.0001, che quindi da ora sarà considerato il valore di default. Un'ultima considerazione riguarda infine il numero massimo di epoche che conviene impostare e, come si evince dalle interruzioni dell'early stopping, è stato incrementato a 15 per ottenere un ottimo risultato di accuracy in tutti i casi.

Quindi riassumendo le scelte fatte prima del prossimo test:

- Early Stopping SI
- Learning Rate 0.0001
- Epoche 15

3. Analisi dei risultati nel caso in cui il base model venga addestrato o meno. Gli altri parametri sono settati di default come segue:
   a. Dimensione della resize: 96
   b. Train e Test Augmentation: NO
   c. Early Stopping: SI
   d. Epoche: 15
   e. Learning Rate: 0.0001

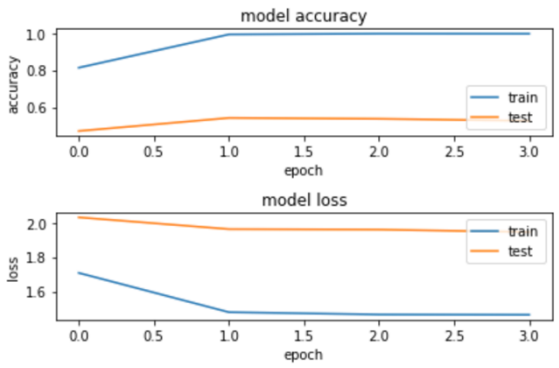
DATASET	ACCURACY	LOSS	EPOCHE – REALMENTE FATTE	ADDESTRAMENTO BASE MODEL	TEMPO
10Random	0.9579	1.5041	15 – 12 EPOCHE	SI	5m 26s
10Random	0.7423	1.7468	15 – 15 EPOCHE	NO	1m 57s

10Random	0.7280	1.7578	20 – 20 EPOCHE	NO	8m 37s
10Random	0.7691	1.7199	25 – 25 EPOCHE	NO	10m 44s
10Random	0.7744	1.7086	30 – 28 EPOCHE	NO	12m 33s
10Poorest	0.5071	1.9632	15 – 7 EPOCHE	SI	1m 33s
10Poorest	0.7861	1.7776	15 – 15 EPOCHE	NO	1m 03s
10Poorest	0.8003	1.7183	20 – 20 EPOCHE	NO	1m 22s
10Poorest	0.8187	1.6939	25 – 25 EPOCHE	NO	1m 40s
10Poorest	0.8414	1.6783	30 – 28 EPOCHE	NO	1m 44s

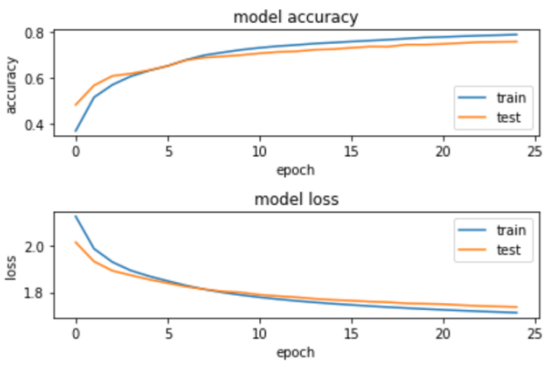
Analizzando l'accuracy del dataset "10PoorestClasses", possiamo notare come essa aumenta di circa il 30% quando l'addestramento del base model è disattivo. Il motivo per cui accade ciò è riconducibile al fatto che in questo dataset ci sono poche immagini, e quindi se bisogna addestrare anche i pesi, è facile ottenere un decadimento delle prestazioni. Nel caso invece del dataset "10RandomClasses", visto che ci sono molte più immagini, l'aggiornamento dei pesi del base model aiuta l'addestramento a trarre dei risultati migliori. Pertanto, da ora in poi si disattiverà l'addestramento dei pesi quando si tratterà il dataset più piccolo "10PoorestClasses", mentre per dataset più corposi come "10RandomClasses" e "43Classes" l'addestramento dei pesi sarà attivato. Al contempo si può notare come occorrano maggiori epoche quando il dataset in input è "10PoorestClasses" e l'addestramento del base model è disattivo, pertanto oltre a disattivare l'addestramento del base model, quando vi è in analisi il suddetto dataset verranno anche incrementate le epoche a 30.

Inoltre, risulta doveroso mostrare il diverso andamento che si ottiene durante l'addestramento dell'algoritmo con il dataset "10PoorestClasses":

Addestramento base model ATTIVO



Addestramento base model DISATTIVO



Come si può osservare, oltre a protrarsi per più epoche, l'algoritmo in cui l'aggiornamento dei pesi del base model è disattivo ha un andamento più "smooth" verso l'ottimo, mentre in caso contrario raggiunge quasi istantaneamente l'ottimo che ritiene raggiungibile, e che come analizzato in precedenza, non è il vero ottimo.

4. Individuazione della resize più adeguata tra quelle disponibili. Gli altri parametri sono settati di default come segue:
    a. Train e Test Augmentation: NO
    b. Early Stopping: SI
    c. Epoche: 30 per “10PoorestClasses”, 15 per gli altri
    d. Learning Rate: 0.0001
    e. Addestramento base model: NO per “10PoorestClasses”, SI per gli altri

DATASET	ACCURACY	LOSS	RESIZE	EPOCHE – REALMENTE FATTE	TEMPO
10Random	0.9579 / 0.9832	1.5041 / 1.4777	96	15 – 12 EPOCHE / 15 – 14 EPOCHE	5m 26s / 10m 50s
10Random	0.8404	1.6256	128	15 – 12 EPOCHE	14m 40s
10Poorest	0.8414	1.6783	96	30 – 28 EPOCHE	1m 44s
10Poorest	0.8442	1.6824	128	30 – 27 EPOCHE	2m 28s
10Poorest	0.8385	1.6690	160	30 – 27 EPOCHE	3m 38s
10Poorest	0.7776	1.7516	192	30 – 17 EPOCHE	3m 04s
10Poorest	0.7975	1.7169	224	30 – 20 EPOCHE	1m 32s

Analizzando l’andamento dei test in quest’ultimo caso si evince come incrementare la resize non comporta un aumento significativo dell’accuracy. Piuttosto porta ad una lieve diminuzione di questa e soprattutto viene richiesto molto più tempo per completare l’addestramento. Per quanto riguarda invece l’andamento del dataset “10RandomClasses”, si osserva che con un lieve aumento della resize ci sono leggeri peggioramenti nei risultati. Da questi è stato supposto che lo stesso ragionamento sia applicabile anche ai dataset “10RandomClasses” e “43Classes”, quindi la miglior resize è 96x96.

5. Individuazione del miglior optimizer del modello. Gli altri parametri sono settati di default come segue:
    a. Train e Test Augmentation: NO
    b. Early Stopping: SI
    c. Epoche: 30 per “10PoorestClasses”, 15 per gli altri
    d. Learning Rate: 0.0001
    e. Addestramento base model: NO per “10PoorestClasses”, SI per gli altri
    f. Dimensione della resize: 96

DATASET	ACCURACY	LOSS	OPTIMIZER	EPOCHE – REALMENTE FATTE	TEMPO
10Random	0.9579 / 0.9832	1.5041 / 1.4777	RMSProp	15 – 12 EPOCHE / 15 – 14 EPOCHE	5m 26s / 10m 50s
10Random	0.8373 / 0.8264	1.6277 / 1.6394	Adam	15 – 7 EPOCHE / 15 – 9 EPOCHE	3m 7s / 11m 12s
10Poorest	0.8414	1.6783	RMSProp	30 – 28 EPOCHE	1m 44s
10Poorest	0.7805	1.7356	Adam	30 – 22 EPOCHE	39s

Intuitivamente, la rete riporta risultati migliori quando viene utilizzato l'optimizer RMSProp, difatti quest'ultimo è l'optimizer di default per la rete, e che quindi di conseguenza resterà immutato.

Dopo quest'ultimo test, è riportato un riassunto delle scelte fatte:

- Early Stopping: SI
- Epoche: 30 per "10PoorestClasses", 15 per gli altri
- Learning Rate: 0.0001
- Addestramento base model: NO per "10PoorestClasses", SI per gli altri
- Dimensione della resize: 96
- Optimizer: RMSProp

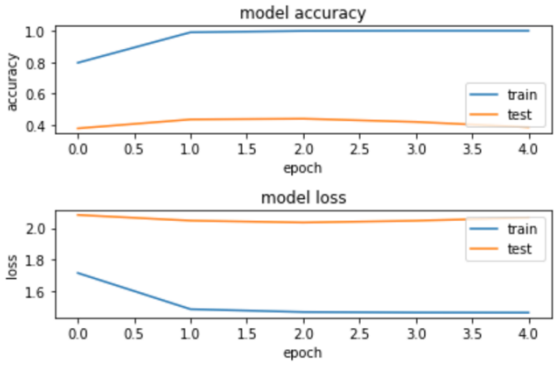
Graficando gli andamenti della configurazione iniziale e di quella ottima ottenuta alla fine della rete MobileNetV2 si ottiene:

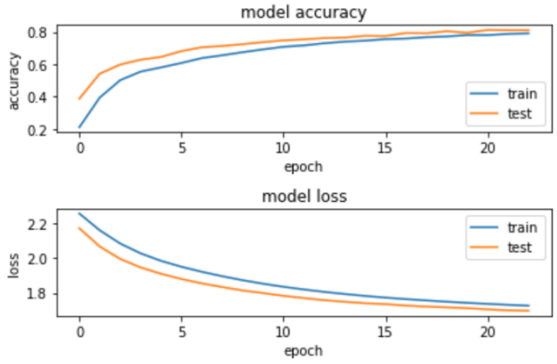
- Nel caso del dataset "10PoorestClasses":

TEST DI PARTENZA:
5 Epoche Massime; Learning Rate = 0.0001; Addestramento del Base Model = SI ed Early Stopping = SI.

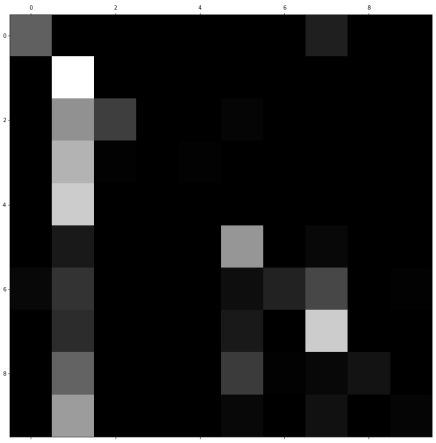
TEST FINALE ALL'OTTIMO:
30 Epoche Massime; Learning Rate = 0.0001; Addestramento del Base Model= NO ed Early Stopping = SI.

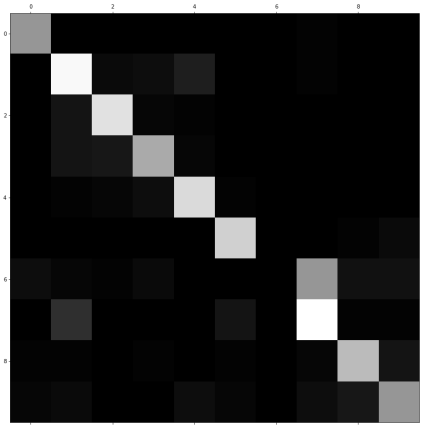
ANDAMENTO DELL'ADDESTRAMENTO





CONFUSION MATRIX





PREDIZIONI CORRETTE ED ERRATE

292 classificazioni corrette
414 classificazioni errate

532 classificazioni corrette
174 classificazioni errate

- Nel caso del dataset “10RandomClasses”:

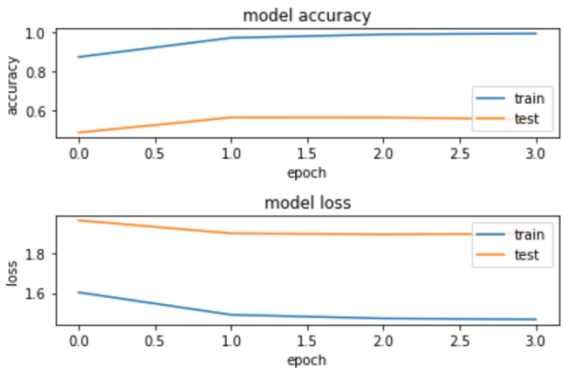
TEST DI PARTENZA:

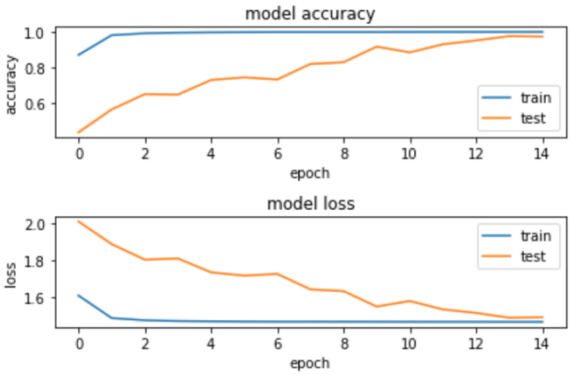
5 Epoche Massime; Learning Rate = 0.0001; Addestramento del Base Model = SI ed Early Stopping = SI.

TEST FINALE ALL’OTTIMO:

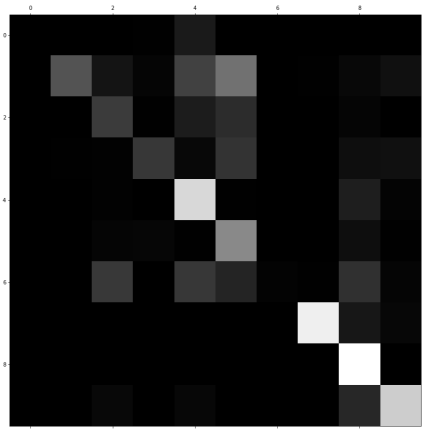
15 Epoche Massime; Learning Rate = 0.0001; Addestramento del Base Model = SI ed Early Stopping = SI.

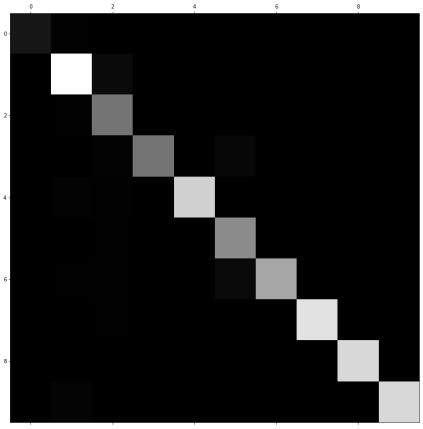
ANDAMENTO DELL’ADDESTRAMENTO





CONFUSION MATRIX





PREDIZIONI CORRETTE ED ERRATE

1961 classificazioni corrette
1248 classificazioni errate

3108 classificazioni corrette
101 assificazioni errate

Dopo aver esaurientemente migliorato la rete pre-trained MobileNetV2, l'analisi ora si è focalizzata sul miglioramento della PersonalCNN, tramite le fasi precedentemente esposte:

1. Ricerca del minor numero di nodi per i layers:

# nodi	Dataset	Epoche	Accuracy	Loss	Durata
32	10Random	10	0.9938	0.0211	01:06
64	10Random	10	0.9931	0.0198	01:42
128	10Random	10	0.9800	0.0737	03:02
32	10Poorest	10	0.9844	0.0637	00:16
64	10Poorest	10	0.9518	0.1697	00:20
128	10Poorest	10	0.9943	0.0263	00:33
32	43Classes	10	0.9837	0.0653	02:10
64	43Classes	10	0.9842	0.0619	03:27
128	43Classes	10	0.9737	0.110	06:15

Osservando le accuracy di test per i vari dataset a parità di nodi nei livelli convoluzionali, risulta evidente come la rete con 32 nodi per livello sia la più efficiente, coniugando una velocità prestazionale maggiore rispetto alle altre due e raggiungendo ugualmente precisione e loss molto soddisfacenti.

2. Ricerca del miglior numero di nodi per il penultimo layer fully-connected:

# nodi	Dataset	Epoche	Accuracy	Loss	Durata
1024	10Random	10	0.9936	0.0182	01:10
512	10Random	10	0.9935	0.0195	01:05
256	10Random	10	0.9928	0.0226	01:03

A parità di dataset e di epoche, i dati mostrano come sia preferibile un penultimo layer Dense costituito da 512 nodi. Le performance del modello con tale caratteristica raggiungono quelle in cui è adottato un livello da 1024 nodi, ma in aggiunta è presente un risparmio computazionale e temporale che, anche se lieve, comunque risulta significativo e vantaggioso per accelerare l'esecuzione dei futuri test da effettuare. Pertanto, la ricerca del miglior numero di nodi per il penultimo fully-connected ha evidenziato un ottimo compromesso tra precisione e velocità con 512 nodi.

3. Analisi dell'early stopping:

Pertanto, a partire dalla CNN con layer convoluzionali e penultimo Dense formati rispettivamente da 32 e 512 nodi, è stato testato l'early stopping per analizzarne i risultati. Il numero di epoche è aumentato a 15. Inoltre, nella colonna relativa all'early stopping in caso di suo utilizzo, è riportata l'epoca di fine addestramento.

Dataset	Epoche	Early Stopping	Accuracy	Loss	Durata
10Random	15	No	0.9916	0.0270	01:35
10Random	15	Si (11)	0.9956	0.0196	01:17
10Poorest	15	No	0.9958	0.0023	00:20
10Poorest	15	Si (14)	0.9844	0.0564	00:19
43Classes	15	No	0.9857	0.0582	03:14
43Classes	15	Si (12)	0.9800	0.0816	02:40

Dai risultati riportati è evidente come una rete di così piccole dimensioni non sfrutti tutti i benefici che normalmente sono introdotti con l'early stopping. La precisione acquisita non varia di molto con il suo utilizzo; tuttavia, si è ritenuto appropriato adottare tale tecnica come forma di prevenzione dall'overfitting e per ridurre la durata degli esperimenti, diminuendo anche di conseguenza il relativo costo computazionale.

4. Confronto tra i risultati delle funzioni di attivazione "ReLU" e "LeakyReLU":

Un terzo test effettuato ha coinvolto le funzioni di attivazione dei tre livelli convoluzionali e del penultimo layer di tipo Dense che compongono la rete. Finora è sempre stata utilizzata la ReLU (Rectified Linear Unit). Oltre a quest'ultima, sarà ora testata la LeakyReLU. Essa, a differenza delle ReLU, non comporta la disattivazione completa dei nodi, e quindi permette loro di addestrarsi nella speranza che successivamente possano portare beneficio alla rete. I risultati nella seguente tabella:

Dataset	Epoche	LeakyReLU	Accuracy	Loss	Durata
10Random	15 (11)	No	0.9956	0.0196	01:17
10Random	15 (9)	Si	0.9882	0.0457	01:11
10Poorest	15 (14)	No	0.9844	0.0564	00:19
10Poorest	15 (13)	Si	0.9915	0.0258	00:22
43Classes	15 (12)	No	0.9800	0.0816	02:40
43Classes	15 (9)	Si	0.9824	0.0685	02:17

Nonostante la funzione LeakyReLU sia leggermente più onerosa, l'early stopping ha permesso una riduzione della durata temporale dell'esperimento, dal momento che raggiunge l'ottimo prima rispetto ad una rete con funzione di attivazione ReLU. Tuttavia, i risultati conseguiti non sembrano differenziarsi in maniera significativa da quelli effettuati nei test precedenti. Di conseguenza, si è scelto di adottare entrambe le funzioni per poter effettuare una scelta discriminante nel seguito dell'esperimento.

Potendo riassumere i test svolti finora, una configurazione ottimale della semplice rete neurale convoluzionale sembra essere:

- Livelli convoluzionali a 32 nodi;
- Penultimo Dense a 512 nodi
- Early stopping.

5. Individuazione del miglior optimizer del modello

Come ultimo test per la rete PersonalCNN, si è studiato l'optimizer più appropriato tra "Adam", già utilizzato di default finora, e "RMSProp", adottato invece da MobileNetV2.

Dataset	Optimizer	Accuracy	Loss	Durata
10Random	Adam	0.9882	0.0457	01:11
10Random	RMSProp	0.7697	0.7652	01:57
10Poorest	Adam	0.9915	0.0258	00:22
10Poorest	RMSProp	0.4688	1.5224	00:24
43Classes	Adam	0.9824	0.0685	02:17
43Classes	RMSProp	0.6628	1.2204	03:48

Risulta dunque evidente come un optimizer di tipo RMSProp si allontani in maniera considerevole dai risultati ottenuti dalla rete con ottimizzazione Adam, di conseguenza tale elemento resterà invariato rispetto a prima.

Concludendo l'analisi della PersonalCNN, la configurazione ottima risultate è quella con:

- Numero nodi livelli convoluzionali: 32 nodi
- Numero nodi penultimo Dense: 512 nodi
- Early Stopping: SI
- Optimizer: Adam

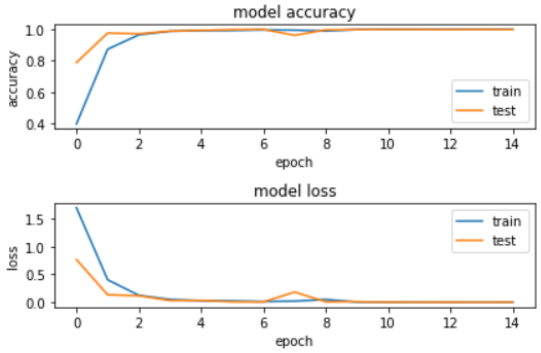
Graficando gli andamenti della configurazione iniziale e di quella ottima ottenuta alla fine della rete PersonalCNN si ottiene:

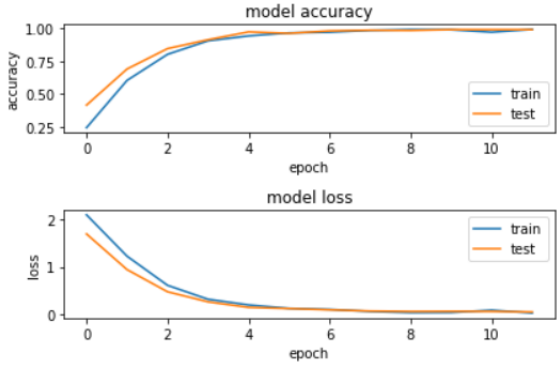
- Nel caso del dataset "10PoorestClasses":

TEST DI PARTENZA:
10 Epoche Massime; EarlyStopping = NO;
nodi livello convoluzionale = 128;
nodi penultimo Dense = 1024

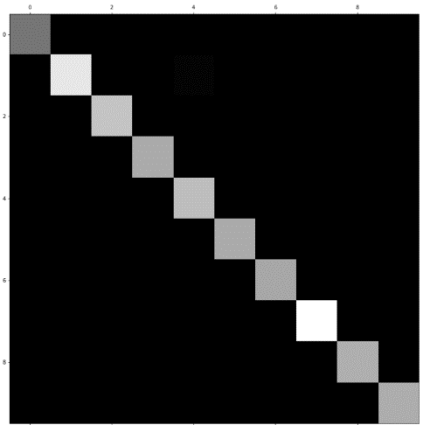
TEST FINALE ALL'OTTIMO:
15 Epoche Massime; EarlyStopping = SI;
nodi livello convoluzionale = 32;
nodi penultimo Dense = 512

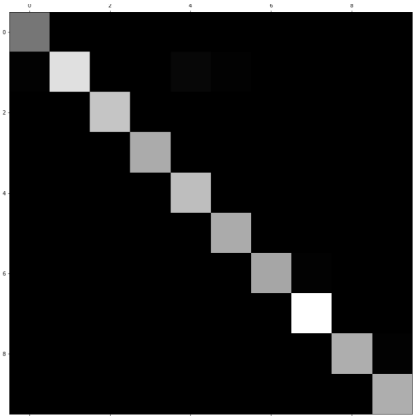
ANDAMENTO DELL'ADDESTRAMENTO





CONFUSION MATRIX





PREDIZIONI CORRETTE ED ERRATE

698 classificazioni corrette
6 classificazioni errate

699 classificazioni corrette
7 classificazioni errate

- Nel caso del dataset “10RandomClasses”:

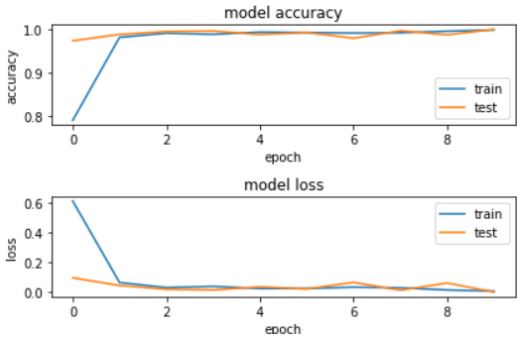
TEST DI PARTENZA:

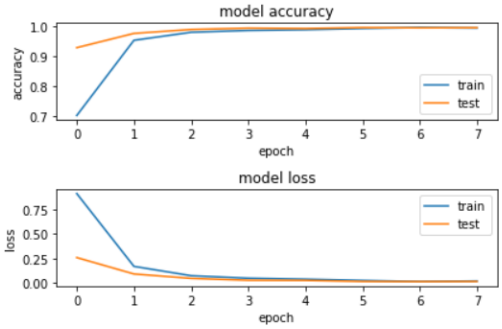
10 Epoche Massime; EarlyStopping = NO;
nodi livello convoluzionale = 128;
nodi penultimo Dense = 1024

TEST FINALE ALL’OTTIMO:

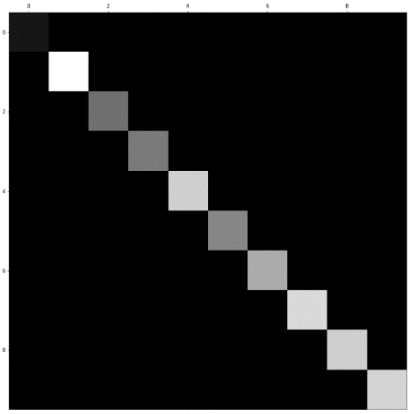
15 Epoche Massime; EarlyStopping = SI;
nodi livello convoluzionale = 32;
nodi penultimo Dense = 512

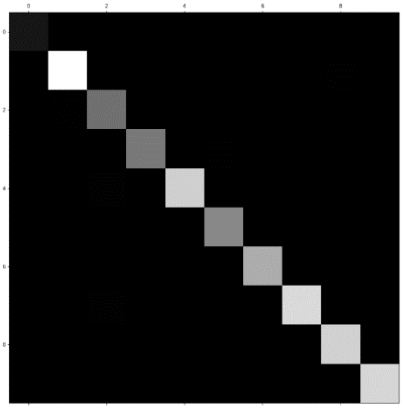
ANDAMENTO DELL’ADDESTRAMENTO





CONFUSION MATRIX





PREDIZIONI CORRETTE ED ERRATE

3206 classificazioni corrette
3 classificazioni errate

3187 classificazioni corrette
22 classificazioni errate

- Nel caso del dataset “43Classes”:

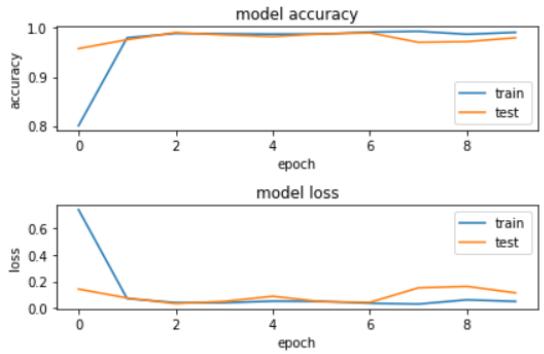
TEST DI PARTENZA:

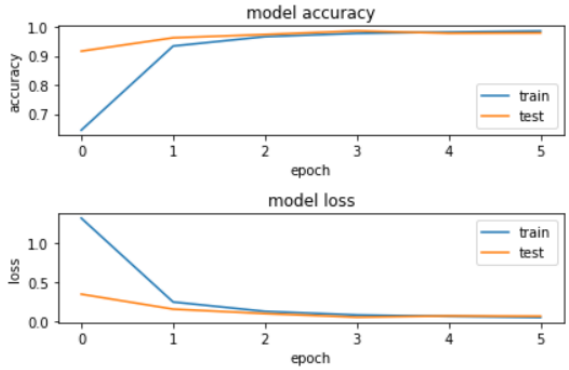
10 Epoche Massime; EarlyStopping = NO;
nodi livello convoluzionale = 128;
nodi penultimo Dense = 1024

TEST FINALE ALL’OTTIMO:

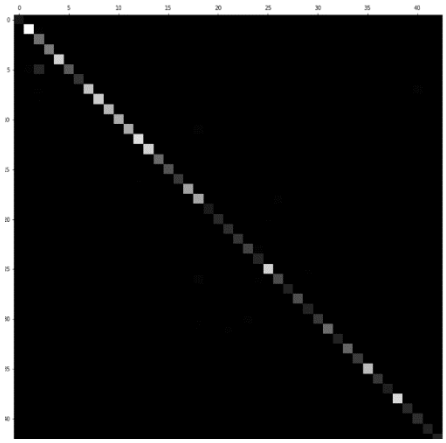
15 Epoche Massime; EarlyStopping = SI;
nodi livello convoluzionale = 32;
nodi penultimo Dense = 512

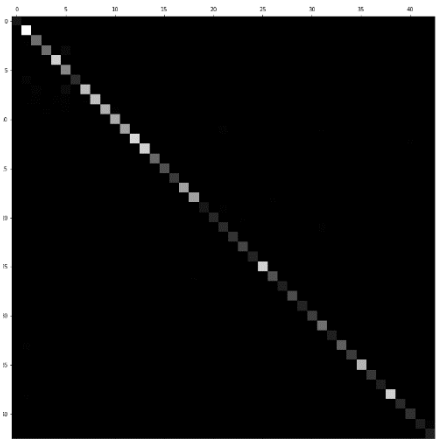
ANDAMENTO DELL’ADDESTRAMENTO





CONFUSION MATRIX





PREDIZIONI CORRETTE ED ERRATE

8581 classificazioni corrette
277 classificazioni errate

8585 classificazioni corrette
273 classificazioni errate

Risulta evidente come la PersonalCNN sia sempre stata vicina all’ottimo. Nei test casistici precedenti ha quasi sempre raggiunto una accuracy tra il 95-99%, precisione osservabile anche dalle sopra riportate confusion matrices e dai pochissimi samples classificati erroneamente. Tuttavia, appare chiaro come il processo di ottimizzazione della rete abbia permesso lo sviluppo di un addestramento molto più regolare. Analizzando i grafici sopra riportati, è mostrato come nei casi di partenza l’andamento fosse sempre caratterizzato da lunghi tratti costanti simili a plateau, ad accentuare lo spreco di risorse della

rete in quelle zone; in seguito, l'andamento è risultato estremamente più regolare, testimoniato dalla monotonicità della funzione e dalla scomparsa di zone dove il fit è risultato poco costante.

Terminata quindi ora anche l'ottimizzazione della PersonalCNN, l'analisi verterà sui risultati riportati dall'utilizzo o meno della augmentation nei generatori di train e test del dataset utilizzato. Una buona augmentation permette alla rete di osservare cartelli stradali in molteplici forme, simulandone una percezione più realistica durante la guida su strada.

Partendo dai valori ottimi ottenuti nei test precedenti, i risultati dei nuovi test con l'augmentation per la rete MobileNetV2 sono i seguenti:

Dataset	Train Augmentation	Test Augmentation	Accuracy	Loss	Durata
10Random	NO	NO	0.9685	1.4932	5m 26s
10Random	SI	NO	0.9648	1.4961	8m 32s
10Random	NO	SI	0.7121 / 0.8196	1.7477 / 1.6437	4m 06s / 10m 29s
10Random	SI	SI	0.7800 / 0.9626 / 0.8838 / 0.9785	1.6787 / 1.4997 / 1.5780 / 1.4823	9m 03s / 17m 53s / 3m 56s / 7m 11s
10Poorest	NO	NO	0.8414	1.6783	1m 44s
10Poorest	SI	NO	0.7507	1.7665	48s
10Poorest	NO	SI	0.5864	1.9097	47s
10Poorest	SI	SI	0.7450	1.7732	48s

Prevedibilmente, l'avvento dell'augmentation ha diminuito i livelli di accuracy e alzato i livelli di loss. La maggior perdita di accuracy (circa il 26%) si è verificata con il dataset "10PoorestClasses" nel caso in cui il train non è sottoposto ad augmentation e il test invece si. L'omologo caso con il dataset "10RandomClasses" invece ha riportato perdite minori (circa il 17%). Ad ogni modo la differenza tra i due casi di maggior interesse, cioè tra il caso ideale, dove l'augmentation risulta disattivata, e il caso reale, dove l'augmentation risulta attivata, riporta dei risultati interessanti. Sono stati effettuati vari test nella casistica di augmentation attiva e si sono registrati risultati contrastanti: alcuni hanno riportato una perdita di accuracy di quasi il 20%, mentre in altri la perdita è stata ininfluente. Un ulteriore caso invece ha registrato una perdita minima del 10%. Possiamo quindi appurare che l'algoritmo non è stabilmente ottimo ad ogni addestramento, ma che nella maggior parte dei casi fa registrare dei risultati davvero impressionanti, considerando le perdite minime rispetto al caso ideale. Saranno in effetti questi ultimi ad essere presi in esame nella successiva analisi delle predizioni corrette ed errate.

Un ulteriore analisi può essere svolta osservando le immagini predette correttamente e le immagini predette in maniera errata:

- Nel caso del dataset “10PoorestClasses” con augmentation non presente sia nel train che nel test si ottiene:

```
532  classified correctly
174  classified incorrectly

Considerando la seguente conversione digit => Nome della classe:

Digit: '0' corrisponde a 'Speed limit (20km/h)'
Digit: '1' corrisponde a 'Dangerous curve to the right'
Digit: '2' corrisponde a 'Road narrows on the right'
Digit: '3' corrisponde a 'Pedestrians'
Digit: '4' corrisponde a 'Bicycles crossing'
Digit: '5' corrisponde a 'End of all speed and passing limits'
Digit: '6' corrisponde a 'Go straight or left'
Digit: '7' corrisponde a 'Roundabout mandatory'
Digit: '8' corrisponde a 'End of no passing'
Digit: '9' corrisponde a 'End of no passing by vehicles over 3.5 metric
tons'
```

PREDIZIONI CORRETTE

PREDIZIONI ERRATE

Come si può notare, alcune predizioni errate sono considerate accettabili in quanto i cartelli sono molto distorti e difficilmente riconoscibili anche ad occhio umano. In altri casi, come nella digits 6, invece i cartelli sono ben distinguibili, quindi probabilmente in quel caso c’è stato un addestramento errato, il quale potrebbe essere dovuto alla scarsità di immagini.

- Nel caso del dataset “10PoorestClasses” con augmentation presente sia nel train che nel test si ottiene:

```
526  classified correctly
180  classified incorrectly

Considerando la seguente conversione digit => Nome della classe:

Digit: '0' corrisponde a 'Speed limit (20km/h)'
```

```
Digit: '1' corrisponde a 'Dangerous curve to the right'
Digit: '2' corrisponde a 'Road narrows on the right'
Digit: '3' corrisponde a 'Pedestrians'
Digit: '4' corrisponde a 'Bicycles crossing'
Digit: '6' corrisponde a 'Go straight or left'
Digit: '7' corrisponde a 'Roundabout mandatory'
Digit: '8' corrisponde a 'End of no passing'
Digit: '9' corrisponde a 'End of no passing by vehicles over 3.5 metric
tons'
```

PREDIZIONI CORRETTE

PREDIZIONI ERRATE

Come si può notare, la maggior parte delle predizioni errate sono da considerare accettabili in quanto i cartelli sono molto distorti e difficilmente riconoscibili anche ad occhio umano. Risalta invece il caso nelle predizioni corrette in cui il cartello di digits 6 venga riconosciuto correttamente. Riferendosi al caso precedente, ciò sta ad indicare che l'algoritmo sembra comunque essere in grado di imparare anche casi con poche immagini per classi.

Infine, un'ultima osservazione particolarmente rilevante vuole evidenziare come il numero di cartelli predetti erroneamente è aumentato solo di 6 rispetto al caso precedente. Questo è un buon risultato, considerando le distorsioni dell'augmentation.

- Nel caso del dataset "10RandomClasses" con augmentation non presente sia nel train che nel test si ottiene:

```
3108  classified correctly
101  classified incorrectly

Considerando la seguente conversione digit => Nome della classe:


Digit: '1' corrisponde a 'Speed limit (30km/h)'
Digit: '2' corrisponde a 'Speed limit (50km/h)'
```

```
Digit: '3' corrisponde a 'Speed limit (60km/h)'
Digit: '4' corrisponde a 'Speed limit (70km/h)'
Digit: '5' corrisponde a 'Speed limit (80km/h)'
Digit: '6' corrisponde a 'No passing for vehicles over 3.5 metric tons'
Digit: '7' corrisponde a 'Priority road'
Digit: '8' corrisponde a 'Yield'
Digit: '9' corrisponde a 'Keep right'
```

PREDIZIONI CORRETTE

PREDIZIONI ERRATE

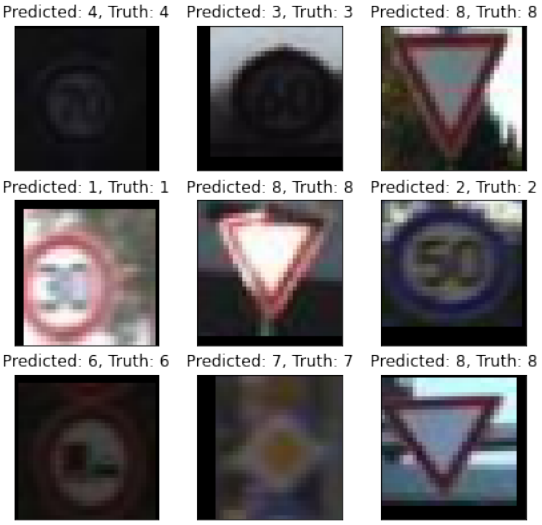
Anche in questo caso, come si può notare, la maggior parte delle predizioni errate possiamo considerarle accettabili in quanto gli errori derivano da cartelli eccessivamente distorti. Una predizione errata che però desta particolare preoccupazione è quella in posizione riga=2 e colonna=1, dove non viene riconosciuta la digit 4. Questo è un problema perché rappresenta una classica situazione di una strada buia in cui la fotocamera non riesce a vedere con chiarezza il valore sul cartello. Possiamo però osservare come l'errore compiuto ha portato al riconoscimento della digit 5, la quale altro non è che un cartello simile, ma con limite di velocità superiore. Questo ci fa pensare che quantomeno la "categoria" del cartello è stata individuata, ma ad ogni modo bisognerebbe potenziare lievemente l'addestramento per riconoscere con più precisione la classe corretta nel caso di immagini scure.

- Nel caso del dataset "10RandomClasses" con augmentation presente sia nel train che nel test si ottiene:

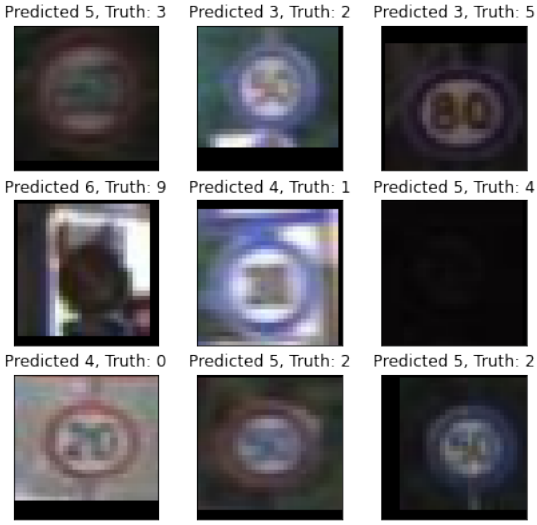
```
3089  classified correctly
120  classified incorrectly

Considerando la seguente conversione digit => Nome della classe:

Digit: '1' corrisponde a 'Speed limit (30km/h)'
Digit: '2' corrisponde a 'Speed limit (50km/h)'
```

```
Digit: '3' corrisponde a 'Speed limit (60km/h)'
Digit: '4' corrisponde a 'Speed limit (70km/h)'
Digit: '5' corrisponde a 'Speed limit (80km/h)'
Digit: '6' corrisponde a 'No passing for vehicles over 3.5 metric tons'
Digit: '7' corrisponde a 'Priority road'
Digit: '8' corrisponde a 'Yield'
Digit: '9' corrisponde a 'Keep right'
```

PREDIZIONI CORRETTE



PREDIZIONI ERRATE



Anche in questo caso, come si può notare, la maggior parte delle predizioni errate possiamo considerarle accettabili in quanto gli errori derivano da cartelli eccessivamente distorti. Anche qui, come nel caso precedente, una predizione errata desta particolare preoccupazione. Quest'ultima è la digits in posizione riga=1 e colonna=3, dove non viene riconosciuta la digit 5. Possiamo però anche qui osservare come l'errore compiuto ha portato al riconoscimento della digit 3, la quale altro non è che un cartello simile, ma con limite di velocità superiore. Questo ci fa pensare che quantomeno la "categoria" del cartello è stata individuata, ma ad ogni modo bisognerebbe potenziare lievemente l'addestramento per riconoscere con più precisione la classe corretta.

Infine, un'ultima osservazione particolarmente rilevante vuole portare alla luce come il numero di cartelli predetti erroneamente è aumentato solo di 19 rispetto al caso precedente. Questo è un buon risultato, considerando le distorsioni dell'augmentation.

Terminata quindi l'analisi dei risultati ottenuti nella rete MobileNetV2 al variare dell'augmentation, è possibile analizzare la medesima casistica nel caso della rete PersonaCNN. In quest'ultima, inoltre, i test effettuati porteranno anche all'interscambio delle funzioni di attivazione ReLU e LeakyReLU, per cercare di comprendere quale tra le due sia la funzione di attivazione più adeguata. I risultati di tali test sono:

Dataset	Activation Function	Train Augmentation	Test Augmentation	Accuracy	Loss	Durata
10Random	ReLU	No	No	0.9956	0.0196	01:17
10Random	LeakyReLU	No	No	0.9882	0.0457	01:11
10Random	ReLU	Si	No	0.9860	0.0453	01:14
10Random	LeakyReLU	Si	No	0.9919	0.0346	01:50
10Random	ReLU	No	Si	0.7722	1.4146	00:43
10Random	LeakyReLU	No	Si	0.8115	1.1947	00:48
10Random	ReLU	Si	Si	0.9869	0.0591	01:39
10Random	LeakyReLU	Si	Si	0.9754	0.0790	00:48
10Poorest	ReLU	No	No	0.9844	0.0564	00:19
10Poorest	LeakyReLU	No	No	0.9915	0.0258	00:22
10Poorest	ReLU	Si	No	0.9788	0.1204	00:14
10Poorest	LeakyReLU	Si	No	0.9632	0.1172	00:20
10Poorest	ReLU	No	Si	0.7691	1.3339	00:14
10Poorest	LeakyReLU	No	Si	0.7408	1.1679	00:10
10Poorest	ReLU	Si	Si	0.9419	0.1935	00:14
10Poorest	LeakyReLU	Si	Si	0.9773	0.1027	00:16
43Classes	ReLU	No	No	0.9800	0.0816	02:40
43Classes	LeakyReLU	No	No	0.9824	0.0685	02:17
43Classes	ReLU	Si	No	0.9730	0.0997	02:54
43Classes	LeakyReLU	Si	No	0.9761	0.0904	03:19
43Classes	ReLU	No	Si	0.6926	2.4078	02:00
43Classes	LeakyReLU	No	Si	0.6860	2.1190	01:23
43Classes	ReLU	Si	Si	0.9430	0.1952	01:38
43Classes	LeakyReLU	Si	Si	0.9579	0.1608	02:51

I valori qui riportati mostrano, come prevedibile, un lieve decremento prestazionale dovuto all'augmentation: cartelli stradali modificati sono più difficili da riconoscere. Tuttavia, si ritiene necessario effettuare tale fase di pre-processing in quanto è estremamente utile per un utilizzo pratico della rete su strada, rendendola in grado di raggiungere ottime prestazioni anche in applicazioni reali. Se così non fosse, ci si ricondurrebbe al caso in cui la augmentation è effettuata sul solo test set, casistica in cui appare immediato come la rete, non osservando e apprendendo mai in fase di addestramento i cartelli che più rispecchiano situazioni reali su strada, raggiunga mediocri risultati (conclusioni ovviamente da evitare in caso di utilizzo pratico). Inoltre, il test ha mostrato come una funzione di attivazione LeakyReLU, su cui prima vi era indecisione, sia preferibile ad una normale ReLU, raggiungendo, seppur di poco, risultati più soddisfacenti.

Concludendo i test casistici che hanno coinvolto la PersonalCNN, la configurazione ottima di tale rete risulta essere dunque quella con:

- 32 nodi nei livelli convoluzionali
- 512 nodi nel penultimo livello Dense

- Early stopping
- Funzione di attivazione LeakyReLU
- Optimizer Adam

Effettuando ora un discorso analogo a quello per la rete MobileNetV2 anche per PersonalCNN, si analizza il comportamento della configurazione ottima della rete e in particolare i suoi casi d'errore con e senza augmentation.

- Nel caso del dataset "10PoorestClasses" con augmentation non presente sia nel train che nel test si ottiene:

```
696  classified correctly
10  classified incorrectly

Considerando la seguente conversione digit => Nome della classe:

Digit: '0' corrisponde a 'Speed limit (20km/h)'
Digit: '1' corrisponde a 'Dangerous curve to the right'
Digit: '2' corrisponde a 'Road narrows on the right'
Digit: '3' corrisponde a 'Pedestrians'
Digit: '4' corrisponde a 'Bicycles crossing'
Digit: '5' corrisponde a 'End of all speed and passing limits'
Digit: '6' corrisponde a 'Go straight or left'
Digit: '7' corrisponde a 'Roundabout mandatory'
Digit: '8' corrisponde a 'End of no passing'
Digit: '9' corrisponde a 'End of no passing by vehicles over 3.5 metric
tons'
```

PREDIZIONI CORRETTE

PREDIZIONI ERRATE

Come testimoniato dal basso numero di predizioni errate, la rete si comporta in modo eccellente anche con carenza di samples per classe: gli errori sono dovuti principalmente a situazioni in cui i cartelli sono stati ripresi al buio o in condizione di sovra illuminazione. Di particolare interesse tra le predizioni corrette spicca il riconoscimento dell'immagine 8, rilevata durante il movimento.

- Nel caso del dataset “10PoorestClasses” con augmentation presente sia nel train che nel test si ottiene:

```
666  classified correctly
40  classified incorrectly

Considerando la seguente conversione digit => Nome della classe:

Digit: '0' corrisponde a 'Speed limit (20km/h)'
Digit: '1' corrisponde a 'Dangerous curve to the right'
Digit: '2' corrisponde a 'Road narrows on the right'
Digit: '3' corrisponde a 'Pedestrians'
Digit: '4' corrisponde a 'Bicycles crossing'
Digit: '6' corrisponde a 'Go straight or left'
Digit: '7' corrisponde a 'Roundabout mandatory'
Digit: '8' corrisponde a 'End of no passing'
Digit: '9' corrisponde a 'End of no passing by vehicles over 3.5 metric
tons'
```

PREDIZIONI CORRETTE

PREDIZIONI ERRATE

In linea con quanto analizzato precedentemente, l’augmentation ha ridotto lievemente la precisione del modello, incrementando il numero di samples classificate erroneamente. Tuttavia, è risultata necessaria per renderlo più robusto e capace di individuare correttamente segnali ruotati e in condizioni atmosferiche avverse, come la nebbia nelle predizioni corrette 5 e 2.

- Nel caso del dataset “10RandomClasses” con augmentation non presente sia nel train che nel test otteniamo:

```
3164  classified correctly
45  classified incorrectly

Considerando la seguente conversione digit => Nome della classe:
```

```
Digit: '1' corrisponde a 'Speed limit (30km/h)'
Digit: '2' corrisponde a 'Speed limit (50km/h)'
Digit: '3' corrisponde a 'Speed limit (60km/h)'
Digit: '4' corrisponde a 'Speed limit (70km/h)'
Digit: '5' corrisponde a 'Speed limit (80km/h)'
Digit: '6' corrisponde a 'No passing for vehicles over 3.5 metric tons'
Digit: '7' corrisponde a 'Priority road'
Digit: '8' corrisponde a 'Yield'
Digit: '9' corrisponde a 'Keep right'
```

PREDIZIONI CORRETTE

PREDIZIONI ERRATE

Analogamente al caso di “10PoorestClasses” senza augmentation, gli errori sono dovuti principalmente al buio o alla troppa luce presente al momento dell’acquisizione dei samples. Tra le predizioni sbagliate, spicca il caso in cui un cartello sia posizionato davanti ad un altro, rendendo impossibile alla rete di identificarlo correttamente, come mostrato nella predizione errata (0,4) in posizione riga=3 e colonna=2.

- Nel caso del dataset “10RandomClasses” con augmentation presente sia nel train che nel test otteniamo:

```
3133  classified correctly
76  classified incorrectly

Considerando la seguente conversione digit => Nome della classe:

Digit: '1' corrisponde a 'Speed limit (30km/h)'
Digit: '2' corrisponde a 'Speed limit (50km/h)'
Digit: '3' corrisponde a 'Speed limit (60km/h)'
Digit: '4' corrisponde a 'Speed limit (70km/h)'
Digit: '5' corrisponde a 'Speed limit (80km/h)'
```

```
Digit: '6' corrisponde a 'No passing for vehicles over 3.5 metric tons'
Digit: '7' corrisponde a 'Priority road'
Digit: '8' corrisponde a 'Yield'
Digit: '9' corrisponde a 'Keep right'
```

PREDIZIONI CORRETTE

PREDIZIONI ERRATE

Similmente al caso analogo con augmentation con "10PoorestRandom", la accuracy cala lievemente, tuttavia è bilanciata da una maggiore robustezza del modello, capace di adattarsi in modo estremamente preciso e riconoscere cartelli non sempre chiari o immediati.

Eseguendo infine un ultimo confronto tra i risultati ottenuti dalla rete PersonalCNN e la rete MobileNetV2 si osserva che:

Dataset	Rete	Train Augmentation	Test Augmentation	Accuracy	Loss	Durata
10Random	PersonalCNN	NO	NO	0.988	0.045	1m 11s
10Random	PersonalCNN	SI	SI	0.975	0.079	01m 39s
10Random	MobileNetV2	NO	NO	0.968	1.493	5m 26s
10Random	MobileNetV2	SI	SI	0.780 / 0.962 / 0.883 / 0.978	1.678 / 1.499 / 1.578 / 1.482	9m 03s / 17m 53s / 3m 56s / 7m 11s
10Poorest	PersonalCNN	NO	NO	0.991	0.025	22s
10Poorest	PersonalCNN	SI	SI	0.977	0.102	16s
10Poorest	MobileNetV2	NO	NO	0.841	1.678	1m 44s
10Poorest	MobileNetV2	SI	SI	0.745	1.773	48s
43Classes	PersonalCNN	NO	NO	0.982	0.068	02m 17s
43Classes	PersonalCNN	SI	SI	0.957	0.160	02m 51s

Risulta palese come la rete PersonalCNN ottenga dei risultati nettamente superiori con i dataset

“10PoorestClasses” e “10RandomClasses” rispetto alla rete MobileNetV2, non tanto in accuracy quanto nei valori di loss. Una precisazione che però occorre fare è che la rete MobileNetV2 è pensata per lavorare con un discreto, se non alto, numero di samples per classe, come infatti si può notare dalla differenza dei risultati tra il dataset “10PoorestClasses” e il “10RandomClasses”. Inoltre, la rete MobileNetV2 subisce meno effetti negativi con la presenta dell’augmentation quando ci sono abbastanza elementi per classe, come si evince dal caso nel dataset “10RandomClasses”. Pertanto, in conclusione è possibile supporre ottimisticamente che la rete MobileNetV2 possa ottenere dei risultati superiori con dataset molto corposi, come nel caso del “43Classes”.

CONCLUSIONS

I risultati sperimentali hanno permesso di raggiungere l’obiettivo prefissato in questo progetto: confrontando a livello prestazionale la prima rete di tipo pre-trained, importata da TensorFlow, e la seconda rete, la quale presenta solo pochi livelli convoluzionali, è possibile evincere che nel caso di dataset che non presentano un alto numero di samples per classe, la rete PersonalCNN si comporta in maniera molto più efficiente, mentre nel caso di dataset più corposi è possibile supporre che la rete MobileNetV2 ottenga risultati più soddisfacenti.

REFERENCES

1) Riferimenti ReLU, LeakyReLU e softMax:
    a. “MQI – LEZIONE 15” slide 5-6-22-23, lezione del 03 giugno 2020
    b. https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/
    c. https://medium.com/@himanshuxd/activation-functions-sigmoid-relu-leaky-relu-and-softmax-basics-for-neural-networks-and-deep-8d9c70eed91e
2) Riferimento per la loss: https://gombru.github.io/2018/05/23/cross_entropy_loss/
3) Riferimento per gli optimizer: “MQI – LEZIONE 15” slide 13-14-15-16, lezione del 03 giugno 2020
4) Link MobileNetV2:
    a. Rete: https://www.tensorflow.org/api_docs/python/tf/keras/applications/MobileNetV2
    b. Tutorial: https://www.tensorflow.org/tutorials/images/transfer_learning
5) Riferimenti alla documentazione di TensorFlow e in particolare:
    a. ImageDataGenerator: https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator
    b. Confusion Matrix: https://www.tensorflow.org/api_docs/python/tf/math/confusion_matrix
6) Plotting della Confusion Matrix ripreso dal capitolo 3 di “Hands on Machine Learning” di Aurélien Géron