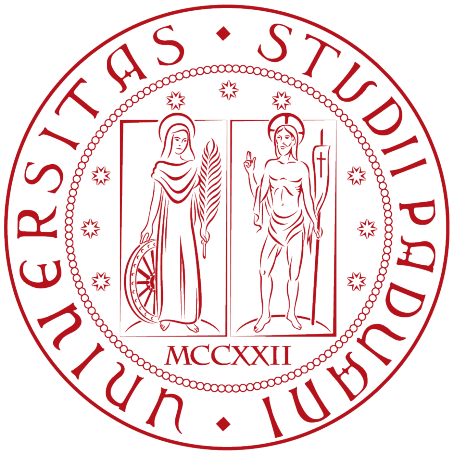
# Università degli Studi di Padova

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA


# Classificazione di segnali stradali tramite ensemble di reti neurali

*Tesi di Laurea, 21 Luglio 2021*


*Relatore*

Prof.Loris Nanni

*Laureando*

Filippo Manzardo


---


ANNO ACCADEMICO 2020-2021

*"Magic's just science that we don't understand yet"*

— Arthur C. Clarke

Dedicato alla mia famiglia ed ai miei amici.

# Sommario

Il presente documento descrive un approccio Deep Learning per la classificazione di cartelli stradali. A causa dell'ambiente di applicazione spesso limitato in risorse, è cruciale trovare una soluzione in grado di ottenere prestazioni ottimali in tempo reale. Lo scopo della ricerca è determinare se limitando le risorse, cioè utilizzando modelli risolutivi efficaci in termini computazionali, si riesca comunque a raggiungere ottime prestazioni.

Introduciamo una tecnica di ensemble di reti convoluzionali leggere, cioè un insieme di reti neurali profonde caratterizzate da un peso e tempi computazionali ridotti. La base di partenza e metro di giudizio è una rete convoluzionale denominata ResNet-201, che richiede tempi e forza lavoro elevati nella fase di classificazione. L'ensemble consiste di reti leggere unite tramite concatenazione delle attivazioni finali e classificate tramite una Support Vector Machine. I risultati ottenuti suggeriscono un'accuratezza tendente alla saturazione, surclassando lo stato dell'arte di reti pesanti ed ottenendo tempi reali.

# Indice

# Elenco delle figure

# Elenco delle tabelle

# Capitolo 1

# Introduzione

Gli ADAS, Automatic Driver-Assistance System, sono dei sistemi elettronici che aiutano e sostituiscono i guidatori, fornendo svariate funzionalità, come l'assistenza al parcheggio, il controllo di trazione adattivo e, sempre più popolare negli ultimi anni, il pilota automatico. La classificazione di cartelli stradali tramite software è uno dei processi più importanti che caratterizzano l'ADAS. Questo processo pone le basi per la maggior parte dei comportamenti che il sistema automatico di bordo deve attuare.
Limiti di velocità, segnali di pericolo e segnali d'obbligo sono tutti variabili che permettono al sistema automatico di effettuare scelte ed operazioni corrette dalla partenza all'arrivo. Per questi motivi, il riconoscimento di ogni cartello stradale è essenziale alla sicurezza del conducente.

## 1.1 Classificazione tramite Deep Learning

Da anni, sistemi Deep Learning vengono utilizzati per replicare la capacità del cervello umano di identificare simboli tramite memoria, analisi dei colori, delle forme e di schemi ricorrenti. Nei sistemi moderni si è raggiunto lo stato dell'arte, superando anche prestazioni umane [29] grazie all'utilizzo di CNN, reti neurali convoluzionali, una classe di reti neurali profonde che si ispirano al funzionamento della corteccia cerebrale e con un'ottima capacità di indipendenza spaziale, caratteristica cruciale nel riconoscimento di simboli che si ripresentano in varie forme. I risultati ottenuti sono dovuti principalmente alla grande mole di dati disponibili (i.e. GTS Dataset) ed ai nuovi dispositivi hardware

sempre più performanti, come GPU, che hanno ridotto i tempi di calcolo nell'allenamento da interi anni a poche ore.

Il funzionamento di una rete neurale si basa sui principi del Machine Learning: apprendimento automatico e generalizzazione delle abilità apprese. L'obiettivo principale è quello di creare un modello in grado di apprendere da un insieme di dati, detto training set, per poi riuscire ad applicare le stesse capacità su ulteriori dati dello stesso dominio applicativo.
Le reti convoluzionali hanno bisogno di un enorme numero di campioni di allenamento, detti pattern, per riuscire a generalizzare ed evitare ciò che viene definito 'overfit': la situazione nella quale il modello apprende a memoria tutti i dati dell'insieme d'allenamento, perdendo o diminuendo l'abilità di estendere la propria esperienza anche a nuovi dati.
Un'ulteriore caratteristica delle CNN (ed in generale delle reti neurali) è la rapidità di classificazione. Se da un lato l'allenamento può durare anche settimane, una volta allenata, la rete neurale, riesce a classificare in tempi relativamente ristretti.
Si possono comunque distinguere reti veloci da reti più lente, o pesanti, principalmente dal numero di parametri e dal tipo di connessioni tra livelli.

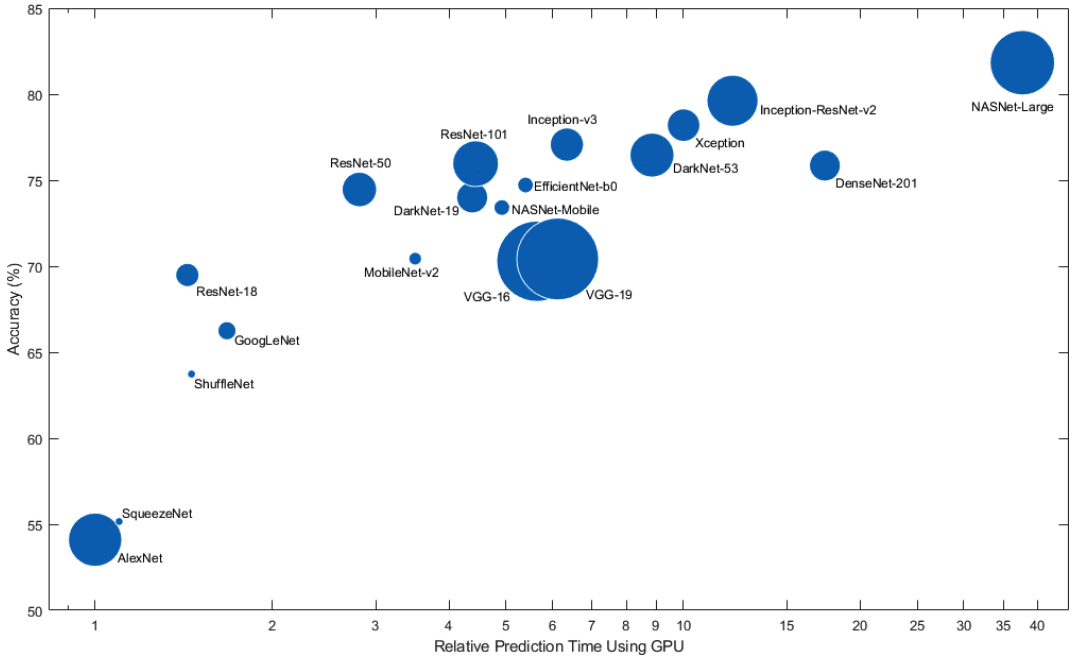



**Figura 1.1:** Esempio di velocità relativa di diverse CNN

La Figura 1.1 mostra come diverse tipologie di reti convoluzionali possono differire in velocità di predizione contro precisione, con rispetto alla rete più veloce. L'area di ogni modello è direttamente proporzionale alla dimensione su disco della rete.
Questa differenza di velocità complica la scelta della rete da utilizzare. Nel nostro caso abbiamo bisogno di reti leggere e veloci per classificare le immagini stradali nel minor tempo possibile ma al contempo dobbiamo ottenere la miglior accuratezza possibile. In che modi possiamo utilizzare reti leggere ma meno accurate per ottenere, o superare, i risultati di una singola rete pesante?

## 1.2 Il German Traffic Sign Dataset

Il *German Traffic Sign Dataset* è un insieme di 39209 immagini per allenamento suddivise in 43 classi e di 12630 immagini utilizzate per la valutazione delle prestazioni. Le immagini contengono cartelli stradali presenti in Germania, fotografate in diverse locazioni e periodi della giornata, in modo tale da variare in colori, contrasti e posizioni. Le immagini sono suddivise in diverse macro categorie: limiti di velocità, segnali di pericolo, precedenze e segnali obbligatori. Inoltre, il numero di pattern per classe non è omogeneo.


**Figura 1.2:** Il German Traffic Sign Recognition Dataset

Il Dataset è utilizzato nel German Traffic Sign Recognition and Detection Benchmark come base per competizioni di classificazione e riconoscimento.

Classificare e riconoscere sono due problematiche diverse [32][34][22], le quali possono utilizzare procedimenti diversi ed ottenere risultati e performance diverse. Il riconoscimento consiste nell'individuare un cartello in un'immagine più complessa, dove potrebbero essere presenti diversi oggetti come macchine, edifici o pedoni, mentre nella classificazione si etichetta il cartello individuato associandolo ad una delle classi presenti. La problematica che viene analizzata in questa ricerca è la *classificazione*.

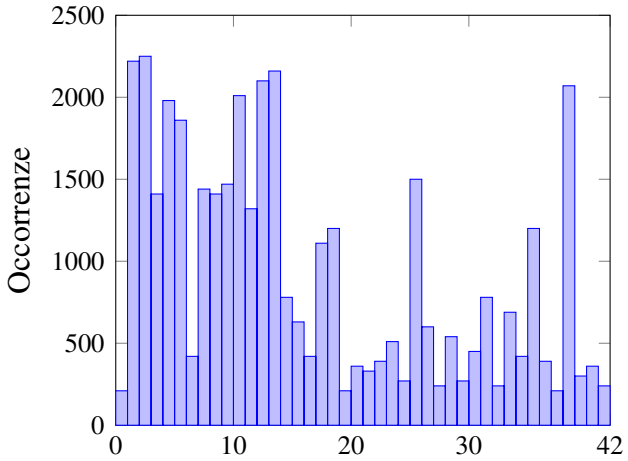



**Figura 1.3:** Numero di occorrenze in funzione della classe di appartenenza

Nel grafico superiore notiamo come le diverse classi differiscono in numero di pattern. Questa eterogeneità potrebbe portare a situazioni nelle quali immagini meno presenti vengono svalutate dal modello. Tuttavia, questo dataset è abbastanza numeroso da consentirci di trascurare questa evenienza [17].

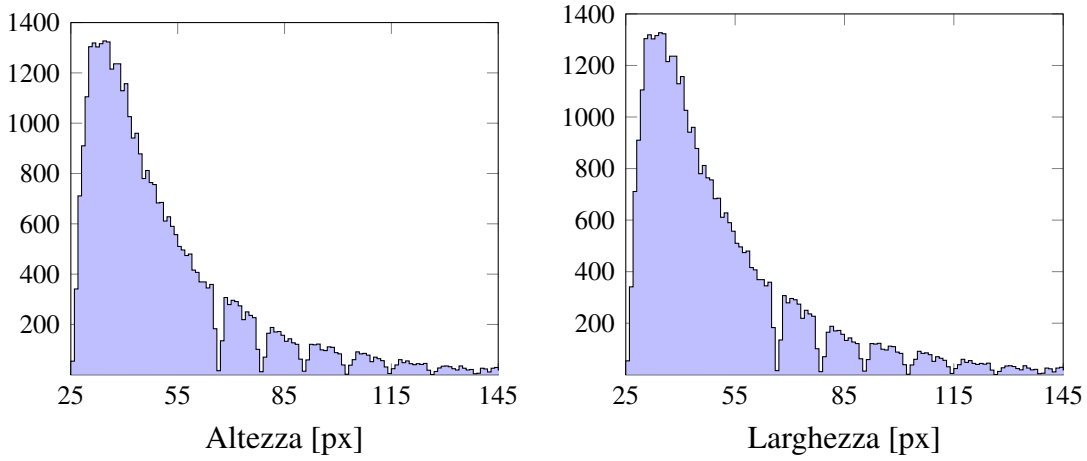



**Figura 1.4:** Numero di occorrenze in funzione della dimensione dell'immagine

## 1.3 Lavori correlati

Il problema del riconoscimento e classificazione di immagini, ed in particolare di cartelli stradali, è molto presente nel panorama scientifico. Gli approcci sviluppati hanno permesso di ottenere accuratezze super-umane [29], cioè sono in grado di classificare con meno errori di quanti ne farebbe una persona.
Le prime soluzioni furono sviluppate seguendo schemi di colori e forme. Le soluzioni basate sui colori utilizzano diversi spazi, cioè descrittori, per individuare un cartello, ad esempio lo spazio RGB [4] e HSV [25], riconoscendo cartelli in base a zone di colore e gradienti variabili.
L'utilizzo di metodi basati su forme è anch'esso un approccio spesso utilizzato nel riconoscimento di cartelli stradali. I cartelli stradali sono spesso molto differenti in forme, che ne identificano la natura: obbligatorio, pericolo, ecc. Loy et al. [20] utilizza la simmetria di segnali triangolari, quadrati e ottagonali per individuare cartelli in una immagine. Queste metodologie assumo a priori condizioni non sempre presenti nella realtà, per esempio l'ortogonalità dei cartelli rispetto alla strada ed i contrasti di colore costanti ed indipendenti dalla quantità di luce presente.

Per quanto riguarda la classificazione, sistemi Machine Learning hanno preso piede nella comunità scientifica e sono diventati la tecnica risolutiva più utilizzata.
SVM [23], Random Forest [33] e Nearest Neighbor [9] sono metodologie classiche applicate al problema, che non sempre ne rappresentano la soluzione ottimale. Negli ultimi anni, le CNN, reti convoluzionali, dette anche ConvNet, sono i modelli Deep Learning che hanno ottenuto le più alte accuratezze nella classificazione di immagini [1] [2], diventando quindi lo standard e stato dell'arte. Ogni anno si cerca comunque di migliorare la struttura e le prestazioni dello stato corrente. Sun et al. [30] sviluppano una tecnica basata su algoritmi genetici per strutturare una CNN in base alle proprie necessità.

Per quanto riguarda il problema delle risorse, vengono spesso ideate nuove soluzioni 'lightweight'. Sikiric et al. [28] studiano quanto si possano limitare i descrittori di un'immagine, quindi la chiarezza dell'immagine, senza comprometterne le prestazioni. Zhang et al. [35] utilizzano due reti, una 'maestra' che rappresenta un modello allenato e

complesso, che ha lo scopo di trasmettere le proprie informazioni ad un modello inferiore e leggero, lo studente. Nonostante ciò, spesso viene analizzato il tempo di allenamento, più che l'utilizzo effettivo del modello.

Passando ora al dataset di riferimento, anch'esso molto presente in letteratura, è stata stilata una classifica che ne elenca alcune risoluzioni e l'accuratezza raggiunta. Sul sito ufficiale, sono elencati i seguenti risultati:

1. CNN with 3 Spatial Trasformer [1]: 99.71/100
2. Committee of CNNs [2]: 99.45/100
3. Color-blob-based COSFIRE filters for object recognition [6]: 98.97/100

In [1] vengono raccolti numerosi dati da classificazioni diverse, confrontando l'utilizzo di diversi ottimizzatori e reti neurali, per ottenere la migliore combinazione di parametri necessaria a raggiungere le migliori accuratezze. Successivamente, vengono disposte combinazioni di Spatial Transformer [15] in diversi punti della rete neurale.

Il miglior classificatore da noi sviluppato ha raggiunto un'accuratezza del 99.69%, posizionandosi secondo in classifica.

# Capitolo 2

# Metodologie e processi

Lo scopo della ricerca è analizzare come i risultati di tecniche di classificazione possono variare in base alle risorse a loro disposizione. Per fare ciò, utilizziamo da un lato lo stato dell'arte delle CNN pesanti, confrontandole con CNN leggere o un insieme di quest'ultime. Questa necessità nasce perché i sistemi di bordo sono spesso limitati in capacità computazionali, quindi è essenziale ottenere il real-time e la leggerezza nei calcoli.
In questo capitolo, analizzeremo gli strumenti utilizzati nella ricerca.
Inizialmente, verranno introdotte le reti neurali ed alcune tecniche risolutive utilizzate. Poi, verrà descritto il metodo di ensemble utilizzato. Infine, verrà illustrato il linguaggio di programmazione utilizzato ed alcuni riferimenti utili.

## 2.1 Reti neurali convoluzionali

Le reti neurali convoluzionali sono una tipologia di ANN, Artificial Neural Networks, caratterizzate da livelli di convoluzione spaziale.
Per rete neurale intendiamo un modello artificiale composto di interconnessioni di informazioni costituite da *neuroni* artificiali, ispirato al funzionamento del cervello umano. Nella maggior parte dei casi una rete neurale artificiale è un sistema adattivo che cambia la propria struttura in base a informazioni esterne o interne che scorrono attraverso la rete stessa durante la fase di apprendimento.

Le CNN si ispirano a processi biologici [11] [5], poiché la struttura delle connessioni di neuroni richiama quella della corteccia cerebrale animale, i cui neuroni rispondono a stimoli solamente da una regione del campo visivo ristretta conosciuta come "campo recettivo".
Al contrario di altri algoritmi di classificazione immagini, le reti convoluzionali utilizzano poca pre-elaborazione, differendo da modelli che necessitano di ritocchi a mano.

Le reti neurali convoluzionali vengono introdotte alla comunità scientifica già nei primi anni ’90; nel 1998 LeCun [18] sviluppa una rete neurale convoluzionale composta da 7 livelli capace di riconoscere cifre numeriche, che venne utilizzata da numerose banche per leggere assegni e valori automaticamente da immagini di dimensione $32 \times 32$ pixel.
L’abilità di processare immagini più dettagliate necessita di una grande capacità di calcolo, che prima del secondo decennio del ventunesimo secolo non era presente. Le reti convoluzionali hanno ottenuto una grande popolarità nel 2012 grazie ai risultati ottenuti da Krizhevsky [16] con una rete convoluzionale denominata *AlexNet* nella competizione di classificazione ImageNet LSVRC-2012 [14], raggiungendo prestazioni di gran lunga superiori a metodologie classiche. AlexNet era composta da 8 livelli, 5 livelli Convoluzionali, poi di 3 livelli Fully Connected e con funzioni di Pool e Attivazione.

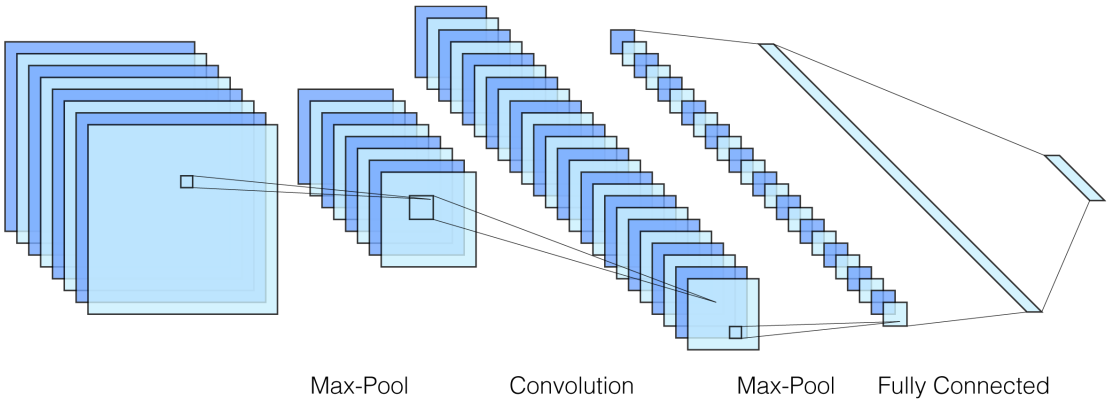



**Figura 2.1:** Struttura di una rete convoluzionale

### Livello di Convoluzione

In matematica, la convoluzione è un operatore che utilizza due funzioni, $f$ e $g$, restituendone una terza che rappresenta come la forma di una influisce sull'altra.

$$f(t) * g(t) := \int_{-\infty}^{\infty} f(t)g(\tau - t)\,\mathrm{d}\tau$$

In una CNN, un livello convoluzionale prende in *Input* un tensore, per esempio la nostra immagine, di *(Input Heigth)* $\times$ *(Input Width)* $\times$ *(Input Channels)* e restituisce una feature map definita da *(Feature map Heigth)* $\times$ *(Feature map Width)* $\times$ *(Feature map Channels)*. Una feature map, o activation map, è l'output di un livello convoluzionale dato un filtro. Rappresenta il risultato dei calcoli matematici avvenuti all'interno del livello e, più in generale, le caratteristiche raccolte dalla CNN.

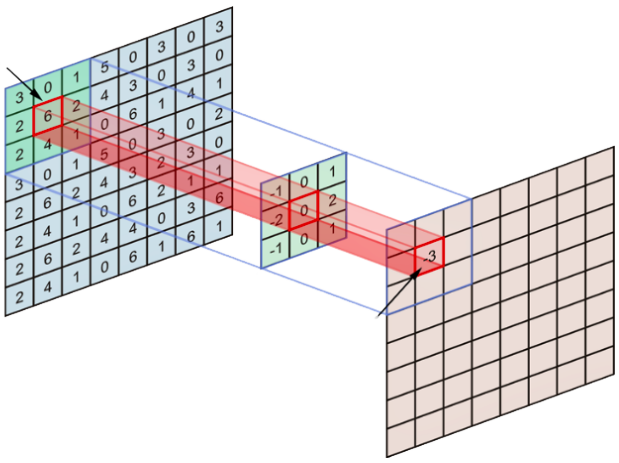


**Figura 2.2:** Esempio di livello convoluzionale

Le operazioni all'interno del livello vengono determinate da dei filtri digitali $L \times L$, i quali vengono fatti scorrere attraverso ogni pixel di input, calcolandone il prodotto scalare. Si possono determinare diversi parametri: la dimensione del filtro, l'utilizzo di padding e di stride, per ottenere la miglior rappresentazione dell'immagine.
I primi livelli analizzano quelle che definiamo caratteristiche ad alto livello, che pian piano si specializzano nei livelli successivi.
Il loro funzionamento simula il comportamento dei neuroni della corteccia frontale dell'animale proprio perché ogni neurone analizza solo la zona locale determinata dal filtro.

### Livello di Pool

Un livello di Pool consiste nell'unione di un gruppo di neuroni secondo algoritmi di media o massimo per ridurre la dimensionalità dell'output finale. Inoltre, raggruppare gli output di un gruppo di neuroni adiacenti permette di ottenere un'invarianza spaziale negli input, in modo tale da riconoscere come uguali immagini leggermente diverse tra loro. La dimensione del filtro di pooling è un altro parametro da considerare per migliorare le prestazioni di una rete.

### Funzione di Attivazione

Tra un livello e l'altro viene utilizzata un'importante funzione detta di attivazione, anch'essa ispirata al funzionamento dei neuroni umani. Aiuta a decidere se un neurone viene attivato o soppresso.
Esistono molte funzioni di attivazione, la più comune è la ReLU

$$f(x) = \max(0, x)$$

ReLU aiuta inoltre la fase di correzione degli errori.

### Livello Fully Connected

Un livello Fully Connected connette ogni neurone di input con ogni neurone di output. In particolare, l'ultimo livello di una rete è spesso Fully Connected ed ha tanti neuroni quanti il numero di classi del problema di classificazione associato.

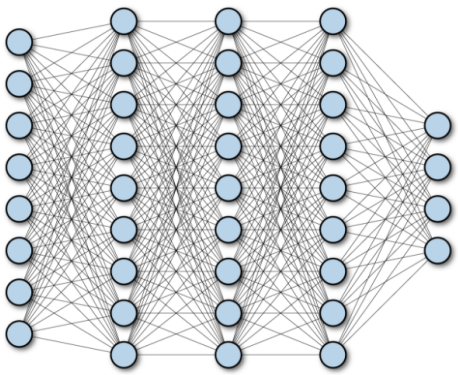


**Figura 2.3:** Esempio di livelli Fully Connected

### 2.1.1 Diverse tipologie di reti

Successivamente al grande successo ottenuto da AlexNet, più e più tipologie di CNN sono state sviluppate, raggiungendo lo stato dell'arte e superando anche le capacità umane nei problemi di classificazione di immagine. In questa ricerca, utilizzeremo alcune delle più famose:

#### GoogleNet

GoogleNet [31], introdotta nel 2014 da Christian Szegedy et Al., raggiungendo lo stato dell'arte nella competizione di ImageNet ILSVRC14. Definiscono una nuova architettura denominata *Inception*, che si distacca dalla classica struttura introdotta con AlexNet, concentrandosi in ridurre la dimensionalità e costi computazionali, utilizzando filtri diversi nello stesso livello.

#### MobileNet

MobileNet [10] è una rete presentata nel 2017, successivamente migliorata alla versione MobileNetv2 [24]. Come suggerisce il nome, MobileNet nasce con l'obiettivo di essere idonea all'utilizzo da dispositivi mobile. È basata su un'architettura snella che utilizza convoluzioni separabili in profondità per creare una rete neurale profonda leggera.

#### ShuffleNet

ShuffleNet [36] è una rete sviluppata nel 2018 da Xiangyu Zhang et al., progettata per essere estremamente efficiente e per l'utilizzo nei dispositivi mobile. Utilizza due nuove operazioni, una convoluzione point-wise e un canale di Shuffle, con lo scopo di ridurre i calcoli senza perdere in accuratezza.

#### SqueezeNet

Squeezenet [13] è la rete più leggera e veloce di quelle utilizzate in questo paper, raggiunge gli stessi risultati di AlexNet [16] ma utilizzando 50x parametri in meno, con un peso su disco di 0.5 MB. Introduce un nuovo modulo, definito *Fire Module*, al cui interno sono presenti un livello convoluzionale squeeze, caratterizzato da filtri 1 x 1, che sfocia

in un *expand layer* dove sono presenti filtri convoluzionali 1 x 1 e 3 x 3. Questo modulo è stato sviluppato per ridurre i parametri di un classico livello convoluzionale.

#### NASNet-Mobile

NASNet-Mobile [39] è la versione leggera della rete NASNet-Large progettata utilizzando un l'algoritmo genetico che opera sul cosiddetto *NASNet search space*. L'architettura della rete si sviluppa operando direttamente nel dataset, dove si cerca il miglior livello convoluzionale per poi impilare questi livelli. Essendo una versione mobile, può essere considerata leggera, ma tra le reti utilizzate in questa ricerca è sicuramente la più lenta.

#### DenseNet

DenseNet [12], a differenza delle precedenti, è considerata una rete neurale pesante. Non punta a limitare le computazioni ma ha come obiettivo primario la miglior accuratezza possibile. Introduce un nuovo blocco, il *DenseBlock*, che a differenza delle architetture classiche, ogni livello del blocco ottiene informazioni 'globali' da tutti i livelli precedenti.

### 2.1.2 Transfer Learning

Il Transfer Learning è una strategia che permette di adattare al proprio problema reti neurali precedentemente allenate su dataset diversi dal nostro.

> Transfer learning and domain adaptation refer to the situation where what has been learned in one setting is exploited to improve generalization in another setting. - Deep Learning, Goodfellow[8]

Quando una CNN viene allenata da zero, tutti i suoi parametri sono inizializzati casualmente, spesso secondo delle distribuzioni Gaussiane. Questo ambiente stocastico potrebbe sfociare in situazioni svantaggiose alla convergenza della rete, spesso allungando i tempi di allenamento.

Grishick et al. [7], scopre nel 2016 che applicando una AlexNet allenata per ImageNet su un dataset diverso, le performance aumentavano rispetto a nuove reti. Similmente, Razavian et al. [26] e Zhou et al. [37] dimostrano come reti più performanti vengono ottenute grazie al Transfer Learning, applicandole su dataset di dimensione inferiore. Secondo Shin et al. [27], il Transfer Learning aiuta reti su Dataset con pochi pattern di allenamento a raggiungere una soluzione soddisfacente.

Questa vantaggiosa qualità è da riportare alla struttura delle reti neurali: se da un lato i primi livelli estraggono le caratteristiche generali dell'immagine, senza specificazione, gli ultimi livelli agiscono da discriminante. Ci basterà quindi adattare gli ultimi livelli della rete al nostro dataset per ottenere numerosi vantaggi.

Tutte le reti utilizzate in questa ricerca sono state pre-allenate su ImageNet, un dataset con milioni di immagini e 1000 classi, verranno adattate sul nostro dataset con 43 classi.

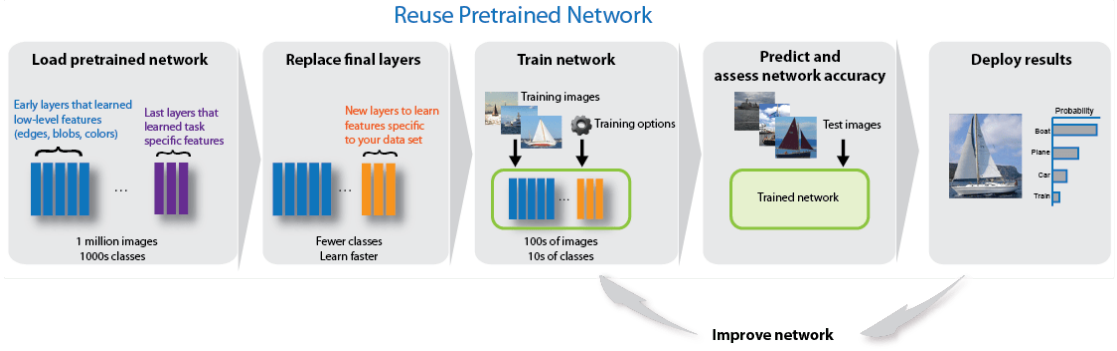



**Figura 2.4:** Funzionamento del Transfer Learning - Matlab

### 2.1.3 Ensemble Learning

L'Ensemble Learning è un paradigma del Deep Learning che si basa sull'utilizzo di più reti, o modelli, per risolvere un problema di classificazione. L'idea di base è che utilizzando più prospettive ed opinioni, unite secondo una regola definita, si possano raggiungere accuratezze più alte con errori meno frequenti [38].
Esistono molte tipologie di Ensemble, il cui obiettivo principale è sviluppare, o utilizzare, dei modelli statisticamente indipendenti tra di loro, in modo tale che non facciano gli stessi errori. L'invarianza è raggiunta in vari modi: allenando modelli su sottoinsiemi diversi [19], utilizzando diversi iperparametri, ecc. Le modalità e le tempistiche di fusione possono influire di molto nel risultato.

#### Stochastic ensemble

Stochastic Ensemble [21] è una tecnica per modellare reti neurali che è stata dimostrata efficace nell'ottenere maggiore indipendenza tra di esse. Consiste nel sostituire, casualmente, le funzioni di attivazione presenti all'interno della rete scelta. Spesso queste funzioni sono esclusivamente ReLU, cambiandole vengono inserite delle non-linearità tra i livelli. Alcune delle reti utilizzate in questa ricerca sono state allenate con Stochastic.

#### Sum Rule

Sum Rule è una metodologia di ensembling molto comune ed utilizzata. Si pone al livello di decisione. Ogni rete neurale, a fine classificazione, restituisce un vettore numerico che rappresenta la confidenza della classificazione. Sum Rule consiste nella sommatoria, preceduta da una normalizzazione standard, di queste confidenze. Alla fine, la classe che riceve il valore più alto sarà il risultato della classificazione.

#### Majority Voting

Majority Voting è un'altra metodologia di ensembling molto comune ed utilizzata. Si pone anch'essa al livello di decisione. Ogni rete ha un voto e lo attribuisce alla classe che considera corretta per ogni immagine del testing set. Alla fine, viene scelta la classe più votata e l'immagine viene etichettata di conseguenza.

## 2.2 Support Vector Machine

I Support Vector Machine sono dei modelli di apprendimento supervisionato utilizzati in problemi di classificazione e regressione. Nascono come classificatori binari. Vengono introdotti da Vapnik nel 1995 [3], anche se la teoria del loro funzionamento era già presente dagli anni '60.

SVM utilizza dei vettori, detti Support Vectors, per determinare una superficie decisionale lineare in uno spazio multi-dimensionale. Inizialmente, venne introdotta una soluzione per pattern linearmente separabili attraverso l'utilizzo di iper-piani. Successivamente venne estesa alla non-linearità con l'utilizzo di funzioni iper-quadratiche.

Se il numero di identificatori è abbastanza elevato (nel nostro caso lo è) le classi possono essere divise da un iperpiano, si parla allora di SVM lineare:

Dato il dataset composto da $(x_1, y_1)...(x_n, y_n)$ punti, dove $x_i \in \mathbb{R}^d$ è il vettore features del pattern $i$ e $y_i$ la sua classe, la teoria SVM si preoccupa di trovare l'iperpiano

$$w^T x - b = 0$$

con $w$ il vettore normale al piano e $\frac{b}{||w||}$ indica la distanza dall'origine. Per determinare il piano (quindi $w$), vengono selezionati due iperpiani che separano le classi in modo tale che la distanza tra di esse sia la massima possibile. Questi iperpiani possono essere descritti dall'equazione:

$$w^T x - b = +1$$
$$w^T x - b = -1$$

Notiamo che sono entrambi paralleli visto che sono determinati dallo stesso vettore normale.

La Figura 2.5 mostra una rappresentazione grafica degli iperpiani con $d = 2$.

Tutti i punti nel o sopra l'iperpiano +1 sono classificati positivamente, mentre tutti i punti nel o sotto l'iperpiano -1 sono classificati negativamente.

La distanza che separa questi due iperpiani è $\frac{2}{||w||}$, quindi per massimizzare la distanza dobbiamo minimizzare $||w||$.

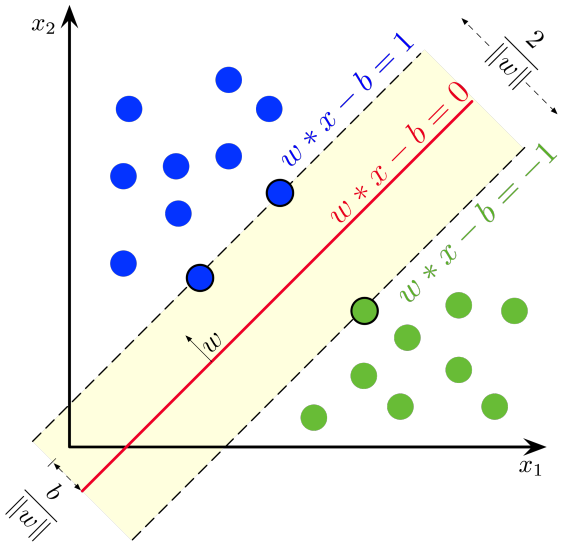



**Figura 2.5:** Visualizzazione iperpiano SVM 2-dimensionale

Un'importante conseguenza è che l'iperpiano può essere descritto interamente dai vettori che giacciono sul margine, che vengono chiamati *support vectors*, limitando quindi il numero di pattern del training set da memorizzare.
Questa metodologia viene definita "Hard-Margin", cioè gli iperpiani sono definiti e fissi. Si può estendere la classificazione ad una soluzione più permissiva, "Soft-Margin", introducendo delle variabili di slack $\xi$ positive che permettano ad alcuni pattern di superare il margine, modificando i margini di separazione $w^T x - b = \pm 1 - \xi_i$ per ogni pattern $i$ del dataset d'allenamento.
I vantaggi nell'utilizzare il classificatore SVM sono:

1. Rappresentazione della superficie decisionale mediante pochi vettori.
2. Dimensionalità del problema relativamente poco influente, più determinante il numero di pattern.
3. Grande velocità di classificazione.

Data la natura binaria degli SVM, l'estensione multi-classe è un ulteriore parametro di decisione. Esistono diverse tecniche di coding: OnevsAll, OnevsOne, SparseRandom... In questa ricerca viene utilizzata OnevsAll che, dato un problema di classificazione con $K$ classi, la codifica OnevsAll genera $K$ classificatori binari, dove per il classificatore i-esimo la classe i-esima è positiva, mentre le $(K-1)$ classi rimanenti (resto del mondo) sono negative.

## 2.3 Soluzione Proposta

La ricerca si basa sull'utilizzo di reti leggere, quindi per aumentare i risultati di classificazione è stato sviluppato un metodo di Ensemble a livello di decisione che ha ottenuto accuratezze immediatamente superiori anche con solo 2 reti.
La soluzione utilizza le reti convoluzionali come estrattrici di caratteristiche, ricavando le attivazioni di un livello avanzato di ogni rete neurale. Queste attivazioni vengono poi concatenate formando un vettore che verrà successivamente classificato da una Support Vectore Machine.
Qualche esempio di livelli utilizzati in questa ricerca sono:

- MobileNetv2: Layer 151 - Global Average Pooling
- GoogleNet: Layer 140 - Global Average Pooling
- ShuffleNet: Layer 169 - Global Average Pooling

Dove 'Layer' è inteso l'oggetto 'Layer' di Matlab.
Spesso questi livelli sono layer di pooling visto che è standard applicarli prima dei livelli finali di classificazione. Ogni rete ha un numero di attivazioni differente, che sono l'output del livello, ma sono tutte vettori numerici.
Una volta calcolate le attivazioni di training vengono concatenate per formare un unico vettore per pattern.

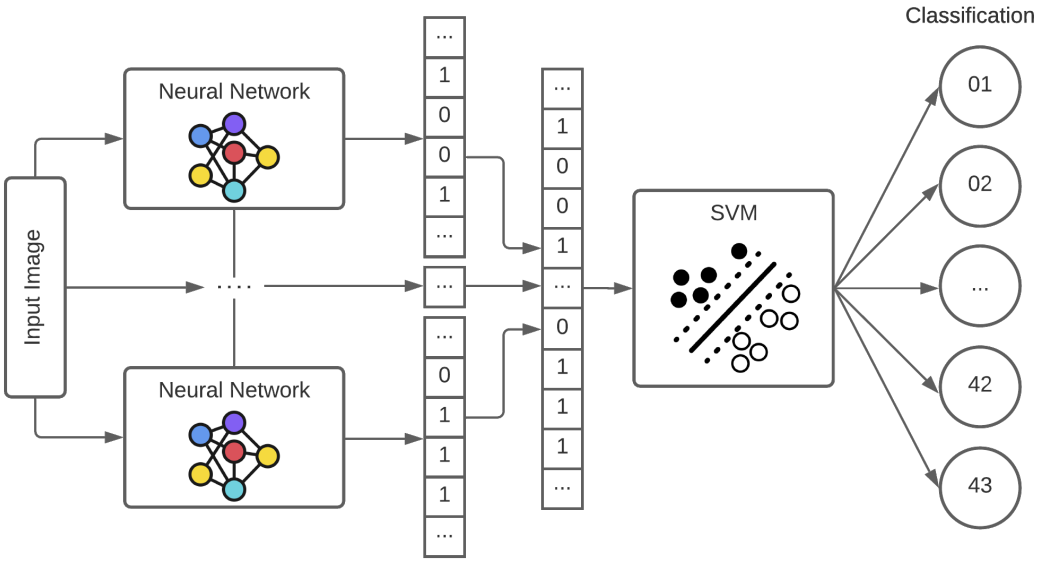



**Figura 2.6:** Activations Ensemble

**Algoritmo 1:** Creazione Pattern SVM

**Data:** TrainingPatterns[1,...,T],NeuralNets[1,..N], Layers[1,..,N]

**Result:** TrainingActivations

```
1: begin
2:     for t ∈ TrainingPatterns do
3:         for net ∈ NeuralNets do
4:             Activation ⟵ CalculateActivations(net, t, Layers[net])
5:             TrainingActivations[t] ⟵ Append(Activation)
6:         end
7:     end
8: end
```

Detto ciò, i vettori numerici possono essere anche molto consistenti, raggiungendo anche le 10'000 componenti. Questa numerosità non influisce di molto nella rapidità di classificazione, che è trascurabile rispetto all'estrazione delle features da parte dei classificatori iniziali. SVM è un modello che scala ottimamente con la dimensione, ma che risente di più nel numero di pattern usati nell'allenamento. Si possono comunque applicare degli algoritmi di riduzione di dimensionalità come PCA, Principal Component Analysis, e LDA, Linear Discriminant Analysis, che in questa ricerca non trattiamo.
Per migliorare le prestazioni dell'SVM viene eseguita una standardizzazione dei dati. Questa standardizzazione consiste in centrare e scalare i dati secondo la media pesata e deviazione standard del vettore.

## 2.4 Dettagli Tecnici

Tutti gli esperimenti sono stati effettuati su Matlab R2020b, utilizzando il Deep Learning Toolbox. Il codice è presente in https://github.com/filippomanzardo. Le sperimentazioni sono state effettuate su un computer fisso, con Windows 10, CPU: i5 4690, GPU: GTX 1660S, 16GB di RAM. Le reti stochastic sono state allenate in remoto con GPU: Titan X. I risultati potrebbero cambiare in base all'architettura utilizzata.

# Capitolo 3

# Esperimenti

In questo capitolo, illustriamo i processi svolti nella ricerca e presentiamo i risultati ottenuti.
Inizialmente, verrà esposto l'allenamento delle reti neurali, base di partenza della ricerca. Successivamente, verranno mostrate le tipologie di ensemble utilizzate e le prestazioni ottenute. Infine, verrà esibito il risultato migliore ottenuto, confrontandolo con la letteratura.

## 3.1 Database Immagini

Le immagini per l'intera fase di allenamento ed analisi sono state scaricate dal sito ufficiale del GTSR Benchmark. L'archivio contenente le immagini di Training pesa ~276 MB mentre l'archivio di Testing ~99 MB. All'interno del primo, le immagini sono suddivise in sottocartelle che determinano le classi di ognuna; è inoltre presente un file ".csv" contenente le informazioni di ogni immagine. Nel secondo, invece, le immagini sono tutte all'interno una cartella dove un file ".csv" ne elenca anche la classe originale. Nel set di dati d'allenamento, 7 immagini erano corrotte e sono state scartate.
Le immagini, a causa delle varie dimensioni, sono state tutte ridimensionate in base alla necessità della rete utilizzata: alcune utilizzano immagini di dimensione ($227 \times 227 \times 3$) pixel, mentre altre ($224 \times 224 \times 3$) pixel.
Non è stato effettuato del Data Augmentation in quanto il dataset era sufficientemente numeroso e le immagini erano già state aumentate e pre-elaborate in precedenza.
Tutte le immagini sono state gestite con l'oggetto 'imageDataStore' nativo di Matlab.

## 3.2 Allenamento reti

La sperimentazione inizia con l'allenamento delle reti neurali convoluzionali. Grazie al Transfer-Learning implementato su Matlab abbiamo risparmiato del tempo, ma è stata comunque la fase più lunga.
Ci sono diversi fattori da considerare nell'allenamento di una CNN, i più importanti sono chiamati iper-parametri. cioè le variabili che determinano la struttura della rete neurale. In particolare, dobbiamo determinare MiniBatch Size, Learning Rate e l'algoritmo di ottimizzazione.

La scelta viene fatta validando reti allenate con determinati parametri. Dopo varie prove, abbiamo scelto di utilizzare una *Mini Batch Size* di 8 immagini, il *Learning rate* di 0.001 e come algoritmo di ottimizzazione lo SGDM.
Una volta decisi gli iper-parametri, abbiamo allenato le reti neurali le cui tipologie sono elencate nella Tabella 3.1, di numerosità variabile.
Queste reti saranno la base di partenza e rappresentano l'approccio standard per risolvere problemi di classificazione.

| | | | |
|---|---|---|---|
| DenseNet-201 | MobileNetv2 | GoogleNet | ShuffleNet |
| NASNetMobile | Squeezenet | AlexNet | |

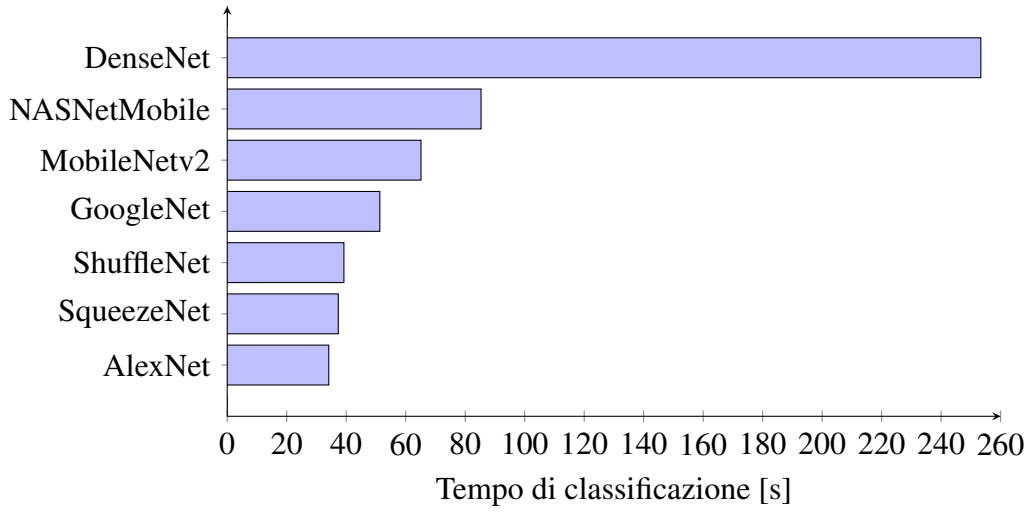
**Tabella 3.1:** Reti utilizzate

Tutte le reti sono state allenate con la funzione `trainNetwork()` di Matlab, che effettua una normalizzazione dell'input ed un rimescolamento delle immagini per epoca. Inoltre, una volta caricata la rete pre-allenata su ImageNet, abbiamo adattato i livelli finali per poter applicare il modello al nostro set di dati.
Le operazioni di adattamento sono due: aumento della velocità relativa di allenamento; cambio del numero di neuroni di output del livello di classificazione da 1000 a 43.
I risultati ottenuti, trascurando eventuali errori e scartando le reti che non hanno ottenuto convergenza nell'allenamento, sono elencati nella Tabella 3.2.

| Rete | Accuracy | | | Classificazione | | Peso |
|---|---|---|---|---|---|---|
| | Max | Min | Avg | A vuoto | In RAM | |
| DenseNet-201 | 99.34% | 98.41% | 99.00% | 288.45 s | 253.35 s | 68.26 MB |
| AlexNet | 98.12% | 96.29% | 97.28% | 35.79 s | 34.14 s | 207.3 MB |
| GoogleNet | 98.54% | 96.72% | 97.85% | 54.45 s | 51.31 s | 21.93 MB |
| ShuffleNet | 98.90% | 97.05% | 97.96% | 41.56 s | 39.23 s | 3.39 MB |
| MobileNetv2 | 98.89% | 95.24% | 97.67% | 68.39 s | 65.14 s | 8.77 MB |
| NASNetMobile | 98.27% | 98.14% | 98.20% | 93.19 s | 85.33 s | 16.1 MB |
| SqueezeNet | 98.23% | 95.19% | 96.04% | 37.90 s | 37.34 s | 2.73 MB |

**Tabella 3.2:** Risultati reti di partenza

Come ci aspettavamo, DenseNet-201, la nostra rete pesante, ottiene risultati maggiori rispetto alle altre reti più leggere sia nel caso migliore, sia nell'accuratezza media.
È importante notare come i tempi di classificazione di reti leggeri siano di molto inferiori a DenseNet-201: nello stesso intervallo di tempo si potrebbero utilizzare 8 AlexNet!
Inoltre, anche il peso della rete può influire nei tempi di classificazione. Come possiamo notare dalla tabella, quando una rete viene caricata in memoria primaria ottiene un incremento nelle prestazioni dovuto alla presenza delle strutture interne della rete che non devono più essere precaricate dalla memoria secondaria. Ovviamente questo aumento di prestazioni scala in base al peso, cioè dalla quantità di dati che devono essere caricati, ma è comunque importante sottolineare che i sistemi embedded di bordo possono essere molto limitati in quantità di memoria primaria disponibile.



## 3.3 Ensemble

Una volta determinate le performance delle reti singole, è arrivato il momento vedere quante reti possiamo utilizzare per superare le capacità di DenseNet.

### 3.3.1 Stochastic

Stochastic Ensembling è un metodo sviluppato da Nanni et al. [21] per sviluppare reti molto diverse tra loro. Consiste nella sostituzione casuale di livelli di attivazione ReLU classici in altri, come SReLU, ELU, PReLU, ecc. Questo processo ha lo scopo di utilizzare reti statisticamente indipendenti tra loro per ottenere un aumento delle performance nell'ensemble. Abbiamo allenato 10 reti MobileNetv2 e 10 reti GoogleNet per utilizzarle in commissione successivamente.
Sfortunatamente, questa modalità di sviluppo di reti causa un rallentamento nei tempi di classificazione, dovuto alla presenza di funzione di attivazione con parametri variabili, che si adattano agli errori.

| **Rete** | **Classificazione** | |
|---|---|---|
| | A vuoto | In RAM |
| Stochastic MobileNetv2 | 157.80 s | 112.34 s |
| Stochastic GoogleNet | 143.42 s | 101.76 s |

**Tabella 3.3:** Tempi reti Stochastic

Inoltre, questo rallentamento non è costante, infatti i tempi di classificazione a carico variano (nei nostri calcoli) da un minimo di 81 secondi ad un massimo di 188 secondi per GoogleNet. Questo limita all'utilizzo di al più 2/3 reti stochastic, per rimanere nei tempi impiegati da DenseNet. Analizzeremo comunque i risultati di più reti per scopi di ricerca e competizione sulle accuratezze raggiunte.

### 3.3.2 Activations

La nostra metodologia di ensemble mediante SVM consiste nella concatenazione delle attivazioni dei livelli finali delle reti. Questa tecnica, però, non necessita necessariamente di più reti, possiamo infatti allenare l'SVM sulle attivazioni della rete singola.

| **SVM** | **Accuracy** | **Varianza** |
|---|---|---|
| DenseNet-201 | 99.00% | -0.26% |
| AlexNet | 97.87% | -0.25% |
| GoogleNet | 98.46% | +0.44% |
| ShuffleNet | 98.61% | -0.29% |
| MobileNetv2 | 98.90% | +0.01% |
| NASNetMobile | 98.35% | +0.08% |

**Tabella 3.4:** Prestazioni SVM su miglior reti di partenza

Notiamo come tendenzialmente si peggiorino le prestazioni.
Prendiamo ora 5 reti di partenza, in particolare due MobileNetv2, due ShuffleNet e una GoogleNet, che hanno ottenuto in media migliori risultati sul dataset.
Calcoliamo le combinazioni di queste reti in funzione del numero di reti che vogliamo utilizzare e applichiamo Activations Ensemble su queste combinazioni.

| **Numero di** | **Accuratezza** | | | | **Classificazione (media)** | |
|---|---|---|---|---|---|---|
| **Reti** | Max | Min | Avg | Dev | Attivazioni | SVM |
| 2 | 99.41% | 99.10% | 99.26% | 0.0010 | 83.70 s | 6.15 s |
| 3 | 99.50% | 99.35% | 99.43% | 0.0005 | 125.55 s | 10.88 s |
| 4 | 99.53% | 99.47% | 99.50% | 0.0002 | 167.40 s | 12.36 s |
| 5 | 99.55% | | | | 209.25 s | 13.87 s |

**Tabella 3.5:** Risultati Activations Ensemble

Ne risulta un aumento delle prestazioni sostanziale, per di più la media della accuratezza esprime quanto sia affidabile questa metodologia di Ensembling. Inoltre, si nota come la classificazione SVM sia rapida. Infine, è importante sottolineare come i tempi di classificazione siano ridotti rispetto a DenseNet-201, che viene surclassata utilizzando solo 2 reti.

### 3.3.3 Confronto metodologie

Confrontiamo ora Activations Ensemble a comuni tecniche di Ensemble a livello di decisione: Sum Rule e Majority Voting.

Utilizziamo inizialmente le reti impiegate nella classificazione di Tabella 3.5.

| **Tipologia di Ensemble** | **Accuratezza** | | | | **Classificazione (media)** |
|---|---|---|---|---|---|
| | Max | Min | Avg | Dev | |
| Activations | **99.41%** | 99.10% | 99.26% | 0.0010 | **89.85** s |
| Sum Rule | 99.22% | **99.35%** | **99.28%** | **0.0004** | 90.99 s |
| M. Voting | 99.07% | 98.50% | 98.84% | 0.0017 | 91.68 s |

**Tabella 3.6:** Risultati di Ensemble con 2 Reti

| **Tipologia di Ensemble** | **Accuratezza** | | | | **Classificazione (media)** |
|---|---|---|---|---|---|
| | Max | Min | Avg | Dev | |
| Activations | **99.50%** | **99.35%** | **99.43%** | 0.0005 | 136.43 s |
| Sum Rule | 98.76% | 99.07% | 98.96% | 0.0013 | **136.10 s** |
| M. Voting | 99.25% | **99.35%** | 99.29% | **0.0003** | 136.85 s |

**Tabella 3.7:** Risultati di Ensemble con 3 Reti

| **Tipologia di Ensemble** | **Accuratezza** | | | | **Classificazione (media)** |
|---|---|---|---|---|---|
| | Max | Min | Avg | Dev | |
| Activations | **99.53%** | **99.47%** | **99.50%** | **0.0002** | **179.76 s** |
| Sum Rule | 98.74% | 99.03% | 98.97% | 0.0013 | 181.32 s |
| M. Voting | 99.37% | 99.48% | 99.44% | 0.0004 | 182.08 s |

**Tabella 3.8:** Risultati di Ensemble con 4 Reti

| **Tipologia di Ensemble** | **Accuratezza** | **Classificazione (media)** |
|---|---|---|
| Activations | **99.55%** | **223.12 s** |
| Sum Rule | 99.02% | 227.68 s |
| M. Voting | 99.49% | 226.42 s |

**Tabella 3.9:** Risultati di Ensemble con 5 Reti

Passando ora a Stochastic Ensemble, analizziamo le performance di committanza di 2/3 MobileNetv2, 2/3 GoogleNet:

| **Tipologia di** | **Accuratezza** | | | | **Classificazione** |
|---|---|---|---|---|---|
| **Ensemble** | Max | Min | Avg | Dev | **(media)** |
| Activations | 99.52% | **99.19%** | **99.37%** | **0.0012** | **223.64 s** |
| Sum Rule | **99.60%** | 99.13% | 99.35% | 0.00015 | 228.43 s |
| M. Voting | 99.39% | 98.80% | 99.06% | 0.0019 | 229.30 s |

**Tabella 3.10:** Risultati di 2 reti MobileNetv2

| **Tipologia di** | **Accuratezza** | | | | **Classificazione** |
|---|---|---|---|---|---|
| **Ensemble** | Max | Min | Avg | Dev | **(media)** |
| Activations | **99.58%** | **99.35%** | **99.48%** | **0.0007** | 324.50 s |
| Sum Rule | 99.44% | 99.09% | 99.31% | 0.0016 | **324.10 s** |
| M. Voting | 99.51% | 99.26% | 99.39% | 0.0008 | 325.85 s |

**Tabella 3.11:** Risultati di 3 reti MobileNetv2

| **Tipologia di** | **Accuratezza** | | | | **Classificazione** |
|---|---|---|---|---|---|
| **Ensemble** | Max | Min | Avg | Dev | **(media)** |
| Activations | **99.19%** | 98.17% | **98.80%** | 0.0039 | **204.54 s** |
| Sum Rule | 99.05% | **98.19%** | 98.71% | **0.0022** | 211.98 s |
| M. Voting | 98.57% | 98.08% | 98.28% | **0.0022** | 213.10 s |

**Tabella 3.12:** Risultati di 2 reti GoogleNet

| **Tipologia di** | **Accuratezza** | | | | **Classificazione** |
|---|---|---|---|---|---|
| **Ensemble** | Max | Min | Avg | Dev | **(media)** |
| Activations | **99.20%** | **98.28%** | **98.98%** | **0.0029** | **308.60 s** |
| Sum Rule | 99.04% | 98.14% | 98.69% | 0.0034 | 312.41 s |
| M. Voting | 98.99% | 98.17% | 98.55% | 0.0031 | 313.67 s |

**Tabella 3.13:** Risultati di 3 reti GoogleNet

I risultati esposti nelle tabelle precedenti evidenziano come Activations Ensemble raggiunga alte prestazioni senza sacrificare tempi computazionali.
Le Tabelle 3.10 - 3.13 dimostrano che MobileNetv2 Stochastic sia molto efficace in questo set di dati, tenendo prestazioni fisse $> 99\%$ già con due modelli. GoogleNet, invece, non raggiunge elevate prestazioni.

In ogni caso Activations Ensemble conferma la sua superiorità ed efficacia tra le scelte di Ensemble decisionali esposte: tempi relativamente ridotti e rendimenti costanti su più livelli.

## 3.4 Miglior Classificatore

Utilizzando la tecnica di ensemble esposta precedentemente, abbiamo definito un'insieme di 5 reti che ritenevamo competitive: GoogleNet Stochastic, MobileNetv2 Stochastic e ShuffleNet.
Allenato il classificatore SVM su 5 di queste reti abbiamo ottenuto i seguenti risultati:

| **Numero di Reti** | **Accuratezza** | | | |
|---|---|---|---|---|
| | Max | Min | Avg | Dev |
| 2 | 99.55% | 98.89% | 99.36% | 0.0021 |
| 3 | 99.64% | 99.28% | 99.55% | 0.0010 |
| 4 | 99.68% | 99.57% | 99.64% | 0.0005 |
| 5 | 99.69% | | | |

**Tabella 3.14:** Risultati miglior classificatore

In particolare, l'ensemble con 5 classificatori ha ottenuto un'accuratezza del 99.69%, raggiungendo il secondo posto in classifica ufficiale.

| **Accuracy** | **Recall** | **Precision** | **F-Score** |
|---|---|---|---|
| 0.9969 | 0.9967 | 0.9964 | 0.9965 |

**Tabella 3.15:** Statistiche miglior classificatore


**Figura 3.1:** Esempi di errore nella classificazione

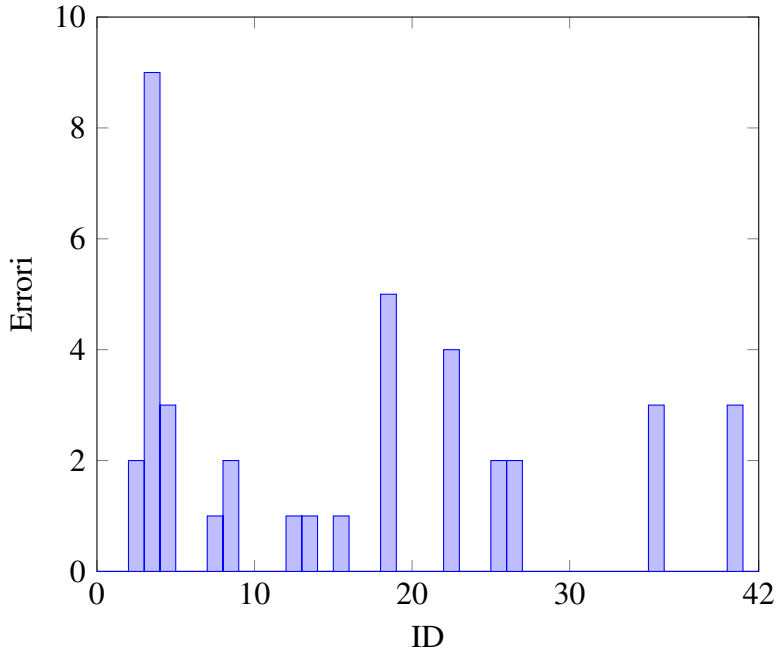



**Figura 3.2:** Numero di errori in funzione della classe di appartenenza

Come possiamo notare, ci sono 3 macro gruppi di errore. Il primo, con le classi da 0 a 10, rappresenta i limiti di velocità, dove sono state fatte più mis-classificazioni. Il secondo, dalla classe 11 alla classe 30 sono i segnali di pericolo, con molte immagini e forme diverse al loro interno. Infine, l'ultimo gruppo segnali rappresenta i cartelli stradali di obbligatori (blu).
In totale sono state classificate errate 39 immagini su 12630.

Nella Tabella 3.15 vengono esposti 4 parametri utilizzati nell'analisi delle prestazioni:

1. **Accuracy** indica quante classificazioni corrette sono state effettuate sul totale
2. **Recall** definisce quanti segnali corretti sono stati selezionati tra il totale dei corretti. È una misura di completezza. $R = \frac{T_p}{T_p+F_n}$
3. **Precision** rappresenta la percentuale di segnali che sono stati correttamente etichettati tra il numero delle etichette. È una misura di esattezza. $P = \frac{T_p}{T_p+F_p}$
4. **F-Score** raggruppa Precision e Recall in un rapporto che ne esprime la media armonica. $F = \frac{2}{\frac{1}{P}+\frac{1}{R}}$

Dove data una classe C, $T_p$, True Positive, indica le classificazioni corrette (per classe); $F_n$, False Negative, indica le immagini della classe C (detta positiva) che sono stati classificati diversamente; $F_p$, False Positive, rappresenta le immagini negative che sono state erroneamente classificate come classe C.
Mentre "Accuracy" è un indicatore globale di analisi delle prestazioni del modello, "Recall", "Precision" e "F-Score" sono intrinsechi alle classi, il valore che riportiamo è la media tra le classi.

Appuntiamo che nella ricerca di Stallkamp [29], il miglior individuo ha classificato correttamente il 99.22% delle immagini, mentre, in media, hanno classificato correttamente il 98.84%.

# Capitolo 4

# Conclusioni

## Riflessioni conclusive

Uno dei primi risultati esposti nel capitolo precedente è la superiorità della classificazione mediante Ensemble Learning rispetto all'utilizzo di singole reti più performanti.
Nella Tabella 3.5 saltano subito all'occhio le prestazioni eccellenti che si ottengono unendo semplicemente due reti.

Un altro importante risultato è la differenza di tempi computazionali evidente. Nei sistemi di bordo sono presenti telecamere che registrano da 24 fps a 60 fps, questo significa che il computer centrale dovrebbe analizzare fino ad 1 immagine ogni $0.0167\,s$. Questo vincolo è rispettabile solamente da reti che lavorano in tempi ristretti. La ricerca ha dimostrato come modelli leggeri utilizzati ed alcune combinazioni di reti riescono a ridurre i tempi guadagnando in prestazioni.

Infine, il metodo di Ensemble presentato in questa ricerca si è rivelato un ottimo e robusto alleato per problemi di classificazione, surclassando tipologie classiche di unione ed ottenendo risultati all'ordine della perfezione.

# Acronimi e abbreviazioni

**ADAS** Automatic Driver-Assistance System. 33

**ANN** Artificial Neural Networks. 33

**CNN** Convolutional Neural Networks. 33

**GPU** Graphics Processing Unit. 34

**HSV** Hue Saturation Value. 34

**LDA** Linear Discriminant Analysis. 34

**PCA** Principal Component Analysis. 34

**ReLU** Rectified Linear Unit. 35

**RGB** Red Green and Blue. 35

**SGDM** Stochastic Gradient Descent with Momentum. 35

# Glossario

**ADAS** Gli *ADAS, Automatic Driver-Asssistance System*,sono sistemi avanzati di assistenza alla guida, basati su sistemi elettronici, che aiutano i conducenti nelle funzioni di guida e di parcheggio. Attraverso un'interfaccia uomo-macchina sicura, ADAS aumenta la sicurezza di auto e strade. I sistemi ADAS utilizzano tecnologie automatiche, come sensori e telecamere, per riconoscere ostacoli nei dintorni o errori del conducente, per poi effettuare una corretta risposta agli eventi. 1

**ANN** Nel campo dell'apprendimento automatico, una *rete neurale artificiale* è un modello computazionale composto di "neuroni" artificiali, ispirato vagamente dalla semplificazione di una rete neurale biologica.
Questi modelli matematici sono troppo semplici per ottenere una comprensione delle reti neurali biologiche, ma sono utilizzati per tentare di risolvere problemi ingegneristici di intelligenza artificiale come quelli che si pongono in diversi ambiti tecnologici (in elettronica, informatica, simulazione, e altre discipline). 7

**CNN** Nell'apprendimento automatico, una *rete neurale convoluzionale* (CNN o ConvNet) è un tipo di rete neurale artificiale feed-forward in cui il pattern di connettività tra i neuroni è ispirato dall'organizzazione della corteccia visiva animale, i cui neuroni individuali sono disposti in maniera tale da rispondere alle regioni di sovrapposizione che tassellano il campo visivo. Le reti convoluzionali sono ispirate da processi biologici e sono variazioni di percettroni multistrato progettate per usare al minimo la pre-elaborazione. Hanno diverse applicazioni nel riconoscimento di immagini e video, nei sistemi di raccomandazione, nell'elaborazione del linguaggio naturale e, recentemente, in bioinformatica. 1, 2, 5, 7–9, 11

**GPU** L'unità di elaborazione grafica, in inglese *Graphics Processing Unit*, è un circuito elettronico progettato per accelerare l'elaborazione di immagini, destinato all'output su un dispositivo di visualizzazione o di calcolo. Le GPU vengono utilizzate in sistemi embedded come telefoni cellulari, personal computer e console di gioco. In un personal computer una GPU può essere presente su scheda video o incorporata sulla scheda madre, mentre in alcune CPU sono incorporate nel die della CPU stessa. Le GPU moderne, sebbene operino a frequenze più basse delle CPU, sono molto più veloci di esse nell'eseguire i compiti in cui sono specializzate. 2

**HSV** *Hue Saturation Value* (HSV), in inglese "tonalità, saturazione e valore", indica sia un metodo additivo di composizione dei colori, sia un modo per rappresentarli in un sistema digitale. La rappresentazione HSV modella come i colori vengono rappresentati sotto differenti luci. 5

**LDA** *L'analisi discriminante lineare* (in inglese linear discriminant analysis o abbreviata LDA) è una generalizzazione della discriminante lineare di Fisher, un metodo usato in statistica, nel riconoscimento di pattern, nell'apprendimento automatico per trovare una combinazione lineare di caratteristiche. La combinazione risultante può essere usata come un classificatore lineare, o più comunemente per una riduzione dimensionale prima di una classificazione statistica. È una tecnica supervisionata, cioè utilizza etichette dei dati, utilizzata per ridurre il numero più o meno elevato di variabili che descrivono un insieme di dati a un numero minore di variabili, mantenendo le caratteristiche che discriminano meglio i gruppi e classi di dati. 18

**PCA** *L'analisi delle componenti principali* (in inglese principal component analysis o abbreviata PCA), anche nota come trasformata di Karhunen-Loève, è una tecnica per la semplificazione dei dati utilizzata nell'ambito della statistica multivariata. Questo metodo fu proposto per la prima volta nel 1901 da Karl Pearson e sviluppato poi da Harold Hotelling nel 1933, e fa parte dell'analisi fattoriale. Lo scopo della tecnica è quello di ridurre il numero più o meno elevato di variabili che descrivono un insieme di dati a un numero minore di variabili latenti, limitando il più possibile la perdita di informazioni. 18

**ReLU** Nel contesto delle reti neurali artificiali, il *rettificatore* è una funzione di attivazione definita come la parte positiva del suo argomento. È anche annoverata tra le funzioni rampa ed è l'analogo di un raddrizzatore in elettronica. Questa funzione di attivazione fu introdotta per la prima volta in una rete dinamica da Hahnloser et al. adducendo motivazioni fortemente biologiche. Nel 2011 è stato dimostrato per la prima volta che la funzione rettificatore permette di ottenere un allenamento migliore per le reti neurali profonde (Deep Neural Network). Questo risultato è ottenuto comparando il rettificatore con le funzioni di attivazione più diffusamente usate fino al 2011: sigmoide logistica e la tangente iperbolica. Nel 2018, la funzione rettificatore è stata riconosciuta come la funzione di attivazione più largamente utilizzata nelle Deep Neural Network. 10, 14, 22

**RGB** *RGB* è un modello di colori di tipo additivo come somma dei tre colori Rosso (Red), Verde (Green) e Blu (Blue). Per le sue caratteristiche, l'RGB è particolarmente adatto nella rappresentazione e visualizzazione di immagini in dispositivi elettronici. Difatti, la maggior parte dei dispositivi, normalmente, usa combinazioni di Rosso, Verde e Blu per visualizzare i pixel di un'immagine. 5

**SGDM** La discesa stocastica del gradiente, *Stochastic Gradient Descent*, è un metodo iterativo per l'ottimizzazione di funzioni differenziabili, approssimazione stocastica del metodo di discesa del gradiente quando la funzione costo ha la forma di una somma. È ampiamente usato per l'allenamento di una varietà di modelli probabilistici e modelli di apprendimento automatico, come macchine a vettori di supporto, regressione logistica e modelli grafici. In combinazione con il metodo di retropropagazione dell'errore, è lo standard de facto per l'allenamento delle reti neurali artificiali. La variante con Momentum, *SGD with Momentum*, introduce un momento che è termine dipendente dalle iterazioni precedenti. Questo momento viene sommato al gradiente nel tentativo di regolarizzare il movimento nello spazio dei parametri. L'aggiornamento dei parametri usa una combinazione lineare del gradiente nell'iterazione corrente e dell'aggiornamento nell'iterazione precedente. Il metodo è stato usato a lungo nell'addestramento delle reti neurali e il nome deriva da un'analogia non del tutto accurata con il momento lineare in fisica, in

quanto il vettore dei parametri può essere visto come una particella che si muove nello spazio dei parametri, soggetta ad accelerazione da parte di una forza il cui potenziale è espresso dalla funzione costo. 20

# Bibliografia

## Riferimenti bibliografici


[1] Álvaro Arcos-García, Juan A. Álvarez-García e Luis M. Soria-Morillo. «Deep neural network for traffic sign recognition systems: An analysis of spatial transformers and stochastic optimisation methods». In: *Neural Networks* 99 (2018), pp. 158–165. ISSN: 0893-6080.

[2] Dan Cireşan et al. «Multi-column deep neural network for traffic sign classification». In: *Neural Networks* 32 (2012). Selected Papers from IJCNN 2011, pp. 333–338. ISSN: 0893-6080.

[3] Corinna Cortes e Vladimir Vapnik. «Support-vector networks». In: *Machine learning* 20.3 (1995), pp. 273–297.

[4] Arturo De La Escalera et al. «Road traffic sign detection and classification». In: *IEEE transactions on industrial electronics* 44.6 (1997), pp. 848–859.

[5] Kunihiko Fukushima e Sei Miyake. «Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition». In: *Competition and cooperation in neural nets*. Springer, 1982, pp. 267–285.

[6] Baris Gecer, George Azzopardi e Nicolai Petkov. «Color-blob-based COSFIRE filters for object recognition». In: *Image and Vision Computing* 57 (2017), pp. 165–174. ISSN: 0262-8856.

[7] Ross Girshick et al. «Region-based convolutional networks for accurate object detection and segmentation». In: *IEEE transactions on pattern analysis and machine intelligence* 38.1 (2015), pp. 142–158.

[8] Ian Goodfellow, Yoshua Bengio e Aaron Courville. *Deep Learning (Adaptive Computation and Machine Learning series)*. e MIT Press, Cambridge, England, 2016.

[9] Anjan Gudigar et al. «Local texture patterns for traffic sign recognition using higher order spectra». In: *Pattern Recognition Letters* 94 (2017), pp. 202–210.

[10] Andrew G Howard et al. «Mobilenets: Efficient convolutional neural networks for mobile vision applications». In: *arXiv preprint arXiv:1704.04861* (2017).

[11] David H Hubel e Torsten N Wiesel. «Receptive fields and functional architecture of monkey striate cortex». In: *The Journal of physiology* 195.1 (1968), pp. 215–243.

[12] Forrest Iandola et al. «Densenet: Implementing efficient convnet descriptor pyramids». In: *arXiv preprint arXiv:1404.1869* (2014).

[13] Forrest N Iandola et al. «SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size». In: *arXiv preprint arXiv:1602.07360* (2016).

[14] *ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012)*. https://image-net.org/challenges/LSVRC/2012/. Accessed: 2021-05-30.

[15] Max Jaderberg et al. «Spatial transformer networks». In: *arXiv preprint arXiv:1506.02025* (2015).

[16] Alex Krizhevsky, Ilya Sutskever e Geoffrey E Hinton. «Imagenet classification with deep convolutional neural networks». In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[17] Alexandra L'heureux et al. «Machine learning with big data: Challenges and approaches». In: *Ieee Access* 5 (2017), pp. 7776–7797.

[18] Yann LeCun et al. «Gradient-based learning applied to document recognition». In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[19] Kuang Liu, Mingmin Zhang e Zhigeng Pan. «Facial expression recognition with CNN ensemble». In: (2016), pp. 163–166.

[20] Gareth Loy e Nick Barnes. «Fast shape-based road sign detection for a driver assistance system». In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*. Vol. 1. IEEE. 2004, pp. 70–75.

[21] Loris Nanni et al. «Comparison of different convolutional neural network activation functions and methods for building ensembles». In: *arXiv preprint arXiv:2103.15898* (2021).

[22] Xishuai Peng et al. «Traffic sign recognition with transfer learning». In: (2017), pp. 1–7. DOI: 10.1109/SSCI.2017.8285332.

[23] Samuele Salti et al. «Traffic sign detection via interest region extraction». In: *Pattern Recognition* 48.4 (2015), pp. 1039–1049.

[24] Mark Sandler et al. «Mobilenetv2: Inverted residuals and linear bottlenecks». In: (2018), pp. 4510–4520.

[25] WG Shadeed, Dia I Abu-Al-Nadi e Mohammad Jamil Mismar. «Road traffic sign detection in color images». In: *10th IEEE International Conference on Electronics, Circuits and Systems, 2003. ICECS 2003. Proceedings of the 2003*. Vol. 2. IEEE. 2003, pp. 890–893.

[26] Ali Sharif Razavian et al. «CNN features off-the-shelf: an astounding baseline for recognition». In: (2014), pp. 806–813.

[27] Hoo-Chang Shin et al. «Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning». In: *IEEE Transactions on Medical Imaging* 35.5 (2016), pp. 1285–1298. DOI: 10.1109/TMI.2016.2528162.

[28] Ivan Sikirić et al. «Traffic scene classification on a representation budget». In: *IEEE Transactions on Intelligent Transportation Systems* 21.1 (2019), pp. 336–345.

[29] J. Stallkamp et al. «Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition». In: *Neural Networks* 32 (2012). Selected Papers from IJCNN 2011, pp. 323–332. ISSN: 0893-6080.

[30] Yanan Sun et al. «Automatically designing CNN architectures using the genetic algorithm for image classification». In: *IEEE transactions on cybernetics* 50.9 (2020), pp. 3840–3854.

[31] Christian Szegedy et al. «Going deeper with convolutions». In: (2015), pp. 1–9.

[32] Yi Yang et al. «Towards Real-Time Traffic Sign Detection and Classification». In: *IEEE Transactions on Intelligent Transportation Systems* 17.7 (2016), pp. 2022–2031. DOI: 10.1109/TITS.2015.2482461.

[33] Fatin Zaklouta, Bogdan Stanciulescu e Omar Hamdoun. «Traffic sign classification using kd trees and random forests». In: *The 2011 international joint conference on neural networks*. IEEE. 2011, pp. 2151–2155.

[34] Huibing Zhang et al. «Real-Time Detection Method for Small Traffic Signs Based on Yolov3». In: *IEEE Access* 8 (2020), pp. 64145–64156. DOI: 10.1109/ACCESS.2020.2984554.

[35] Jianming Zhang et al. «Lightweight deep network for traffic sign classification». In: *Annals of Telecommunications* 75.7 (2020), pp. 369–379.

[36] Xiangyu Zhang et al. «Shufflenet: An extremely efficient convolutional neural network for mobile devices». In: (2018), pp. 6848–6856.

[37] Bolei Zhou et al. «Learning deep features for scene recognition using places database». In: (2014).

[38] Zhi-Hua Zhou, Jianxin Wu e Wei Tang. «Ensembling neural networks: many could be better than all». In: *Artificial intelligence* 137.1-2 (2002), pp. 239–263.

[39] Barret Zoph et al. «Learning transferable architectures for scalable image recognition». In: (2018), pp. 8697–8710.