



Scuola di Scienze

Matematiche, Fisiche e Naturali  
Corso di Laurea in Informatica  
Tesi di Laurea

Individuazione di immagine  
alterate da una telecamera RGB

mal funzionante attraverso un  
agente addestrato

Detection of failed images from  
an RGB camera through trained  
agents

Pietro Bernabei  
Anno Accademico 2019/20

# Contents

<b>1</b>	<b>Introduzione</b>	<b>4</b>
1.1	Motivazioni . . . . .	4
1.2	Obiettivo . . . . .	4
1.3	Organizzazione del lavoro . . . . .	5
<b>2</b>	<b>Fondamenti teorici</b>	<b>6</b>
2.1	Sistemi Critici . . . . .	6
2.1.1	Dependability . . . . .	6
2.1.2	Le Minacce: guasti, errori e fallimenti . . . . .	7
2.1.3	Gli attributi della dependability . . . . .	9
2.1.4	I mezzi per ottenere la dependability . . . . .	10
2.2	Artificial Intelligence . . . . .	14
2.2.1	Agente razionale,Misura di prestazione,Ambiente	14
2.2.2	Agenti che apprendono . . . . .	15
2.2.3	Neural Network . . . . .	17
2.3	Autonomous Driving . . . . .	18
2.3.1	Livelli di automazione della guida . . . . .	18
2.3.2	Visione artificiale . . . . .	20
2.3.3	Reti neurali applicate all'autonomus driving . .	22
2.3.4	Classificatori . . . . .	22
2.3.5	Object Recognition . . . . .	23
2.4	CARLA . . . . .	24
2.4.1	Sensori . . . . .	25
<b>3</b>	<b>Costruzione del dataset</b>	<b>26</b>
3.1	Definizione delle classi . . . . .	26
3.2	Acquisizione . . . . .	27
3.3	Sporcatura . . . . .	27
3.4	Suddivisione del dataset . . . . .	28

<b>4 Costruzione del detector</b>	<b>30</b>
4.1 Convolutional Neural Network . . . . .	30
4.1.1 L'architettura . . . . .	31
4.1.2 Convolutional layer . . . . .	32
4.1.3 Funzione di attivazione Relu . . . . .	34
4.1.4 Pooling layer . . . . .	35
4.1.5 Dense layer . . . . .	35
4.2 La rete . . . . .	36
4.3 Addestramento del rete convuluzionale . . . . .	36
4.3.1 Batch . . . . .	37
4.3.2 Epochs . . . . .	38
4.4 Testing della rete . . . . .	38
<b>5 Esecuzione e risultati</b>	<b>38</b>
<b>6 Conclusioni e lavori futuri</b>	<b>38</b>
<b>7 A Manuale utente</b>	<b>38</b>

# 1 Introduzione

## 1.1 Motivazioni

Nel giro di un paio di secoli il mondo, si è passati dal viaggiare in groppa a un cavallo, alla groppa di una macchina, dal essere il conducente, al condotto. Le macchine a guida autonoma parziali e totali stanno diventando ogni giorno che avanza, realtà. Questa rivoluzione sta permeando il nostro stile di vita, diventando sempre di più accessibili. Questa presenza sempre più forte nella quotidianità di tutti i giorni, fa sì che queste tecnologia debba rispettare degli standard sempre più stringenti, garantendo la continuità del servizio, l'assenza di possibili errori e malfunzionamento. Come per un essere umano, che soffre di miopia, guidare senza occhiali è pericoloso, anche per il sistema di guida autonoma, guidare con le telecamere affette da guasti porta a incidenti. Come verrà espresso in seguito, le telecamere negli attuali sistemi, ricoprono un ruolo decisivo nel momento decisionale della guida autonoma, e un loro malfunzionamento nel processo di acquisizione ripercuote nel sistema decisionale, errori non molto graditi al guidatore.

## 1.2 Obiettivo

Nella seguente tesi si propone un detector, ovvero un sistema software in grado di rilevare, nel flusso di immagini generate dalla telecamera di un mezzo a guida autonoma, la presenza di guasti nel sistema di acquisizione, come congelamento, pixel bruciati, e tanti altri, per poi notificare i possibili esiti al sistema decisionale del mezzo, prevenendo lo sviluppo di fallimenti.

## 1.3 Organizzazione del lavoro

Come verrà esposto in seguito un veicolo a guida autonoma, presenta una suite di sensori, attraverso i quali "vede" l'ambiente circostante. Su queste percezioni, il suo sistema decisionale guida il veicolo in sicurezza. Un particolare componente di questa suite di sensori, sono le fotocamere RGB le quali risultano le più impiegate e importanti tra le tutte, e proprio la loro importanza fa sì che un loro guasto, comporti un fallimento nel sistema decisionale. Per questo il detector è posto tra il sistema di acquisizione della fotocamera RGB e il sistema decisionale, al fine di rilevare la presenza dei guasti nel sensore e comunicare questa al sistema di guida su cui prenderà le sue decisioni. Il detector descritto nella seguente tesi impiega una particolare forma di intelligenza artificiale, rete neurale convoluzionale. Come verrà esposto meglio in seguito, una convolutional neural network è una particolare forma di supervised learning, particolarmente efficiente in ambienti grafici, soprattutto per la classificazione di intere immagini o identificazione di elementi al loro interno. I supervised learning, e quindi anche CNN richiedono una fase di training, nella quale viene definita la propria conoscenza di base (Knowledge base), sulla quale si baserà per prendere le decisioni. Detto ciò lo sviluppo del detector si divide in:

- Definizione e generazione Dataset
- Creazione del modello della ConvNet
- Training del modello della ConvNet

## 2 Fondamenti teorici

### 2.1 Sistemi Critici

Ogni giorno, una persona usa infrastrutture, mezzi, telecomunicazioni, e servizi di qualsiasi genere, affidandosi totalmente al loro funzionamento, alla loro continuità, dando per scontato che non possano subire guasti o malfunzionamenti, perché ove questi avvengano il risultato sarebbe devastante per tutto il nostro sistema di vita. Queste componenti si definiscono come **Sistemi Critici**, e il loro corretto funzionamento, dipende dalla nostra capacità di analizzarne aspetti quantitativi relativi sia a caratteristiche prestazionali, quali velocità di elaborazione o altre misure di efficienza, sia caratteristiche di sicurezza, disponibilità o affidabilità che dimostrino e ci convincano della adeguatezza dei nostri manufatti per i compiti sempre più critici e delicati per i quali li utilizziamo.

#### 2.1.1 Dependability

La **dependability** è una delle proprietà fondamentali dei sistemi informatici insieme a funzionalità, usabilità, performance e costo. Per fornirne una prima definizione, è necessario illustrare i concetti di: [2].

- **Servizio:** Il servizio fornito da un sistema è il comportamento del sistema stesso, così come viene percepito dai suoi utenti.
- **Utente:** Un utente di un sistema è un altro sistema che interagisce attraverso l'interfaccia del servizio.
- **Funzione di sistema:** La funzione di un sistema rappresenta che cosa ci attendiamo dal sistema; la descrizione della funzione di un sistema è fornita attraverso la sua specifica funzionale. Il servizio è detto corretto se realizza la funzione del sistema.

Detto ciò nella sua definizione originale **dependability** è la capacità di un sistema di fornire un servizio su cui è possibile fare affidamento in modo giustificato.[3] Un'esposizione sistematica dei concetti relativi alla dependability consiste di tre parti: La dependability, a sua volta è composta da tre diversi aspetti di cui è necessario tenere conto:

- **Le minacce o impedimenti alla dependability.** Gli impedimenti sono le cause potenziali di comportamenti non previsti.
- **Gli attributi della dependability.** Gli attributi ci permettono di esprimere e verificare il livello di dependability richiesto od ottenuto.
- **I mezzi per ottenere la dependability.** I mezzi sono le tecniche che permettono di ottenere comportamenti corretti, nonostante il verificarsi degli impedimenti.

### 2.1.2 Le Minacce: guasti, errori e fallimenti

Si definisce come:[3]

- **guasto:** la causa accertata o ipotizzata di un errore, derivante da malfunzionamenti di componenti, interferenze ambientali di natura fisica, sbagli dell'operatore o da una progettazione fallace.
- **errore:** è la parte dello stato del sistema che può causare un susseguente fallimento; in alternativa si definisce errore la manifestazione di un guasto all'interno di un programma o di una struttura dati.
- **fallimento:** di sistema è un evento che occorre quando un errore raggiunge l'interfaccia di servizio, alterando il servizio stesso. Quando un sistema viola la sua specifica di servizio si dice che è avvenuto un fallimento; il fallimento è quindi una transizione da

un servizio corretto a un servizio non corretto. La transizione inversa, da un servizio non corretto ad uno corretto, è detta ripristino.

**Il guasto** può rimanere dormiente per un certo periodo, fino alla sua attivazione. L'attivazione di un guasto porta ad un errore, che è la parte dello stato del sistema che può causare un successivo fallimento. I guasti di un sistema possono essere classificati secondo diversi punti di vista, ad esempio fisico, logico e di interazione. Un'altra suddivisione può essere fatta in base alla natura del guasto: un guasto può essere intenzionale o accidentale, malizioso oppure non malizioso; ed ancora in base alla persistenza dove abbiamo guasti permanenti, transienti ed intermittenti. Per una tassonomia completa si rimanda a [2] (riguarda)

**Il fallimento** di un componente si verifica quando il servizio fornito devia dalla sua specifica: si verifica nel momento in cui un errore del componente si manifesta alla sua interfaccia, e diventa quindi un guasto per il sistema. Il fallimento è quindi l'effetto, osservabile esternamente, di un errore nel sistema; gli errori sono in stato latente fino a che non vengono rilevati e/o non producono un fallimento.

La deviazione dal servizio corretto può assumere diverse forme, che vengono chiamate modi di fallimento e possono venire classificati secondo la loro gravità (severity).

Un sistema è formato da un insieme di componenti che interagiscono tra loro, perciò lo stato del sistema è l'insieme degli stati dei suoi componenti. Un guasto causa inizialmente un errore nello stato di uno (o più) componenti, ma il fallimento del sistema non si verifica fino a quanto l'errore non raggiunge l'interfaccia del servizio. La propagazione di errori può permettere ad un errore di raggiungere l'interfaccia di servizio. Questo insieme di meccanismi costituisce la

catena di impedimenti guasto-errore-fallimento (fault-error-failure) La propagazione all'interno di un componente (propagazione interna) è causata dal processo di elaborazione: un errore viene successivamente trasformato in altri errori.

### 2.1.3 Gli attributi della dependability

Il concetto di dependability è la sintesi di più attributi che forniscono misure quantitative o qualitative del sistema:

- **Affidabilità (reliability):** è la capacità del sistema di erogare un servizio corretto in modo continuo; misura la fornitura continua di un servizio corretto.
- **Manutenibilità (maintainability):** la capacità del sistema di subire modifiche e riparazioni; misura il tempo necessario per ristabilire un servizio corretto.
- **Disponibilità (availability):** è la prontezza del sistema nell'erogare un servizio corretto; misura la fornitura di servizio corretto, rispetto all'alternanza fra servizio corretto e non corretto.
- **Confidenzialità (confidentiality):** è l'assenza di diffusione non autorizzata di informazioni; misura l'assenza di esposizione non autorizzata di informazione.
- **Integrità (integrity):** descrive l'assenza di alterazioni improprie del sistema; misura l'assenza di alterazioni improprie dello stato del sistema.
- **La sicurezza (safety)** è poi l'assenza di conseguenze catastrofiche sugli utenti e sull'ambiente circostante. La safety può essere vista come l'affidabilità del sistema .

- **sicurezza (security)** può quindi essere vista come la contemporanea esistenza di availability solo per gli utenti autorizzati, confidentiality, e integrity, dove per “improprie” si intende “non autorizzate” [7].

Ciascuno di questi attributi può essere più o meno importante in base all’applicazione: la disponibilità del servizio è sempre richiesta, anche se può variare sia l’importanza relativa che il livello quantitativo richiesto mentre, la affidabilità, la safety, la confidenzialità e gli altri attributi possono essere richiesti o meno. Nella loro definizione, la disponibilità e la affidabilità evidenziano la capacità di evitare i fallimenti, mentre la safety e la security evidenziano la capacità di evitare specifiche classi di fallimenti come ad esempio fallimenti catastrofici e accesso non autorizzato alle informazioni.

I requisiti di dependability di un sistema sono forniti attraverso una descrizione degli obiettivi richiesti per uno o più degli attributi sopra descritti, rispetto alle modalità di fallimento previste per il sistema. Se i modi di fallimento previsti sono specificati e limitati, si parla di sistemi fail-controlled; un sistema i cui fallimenti sono limitati soltanto all’interruzione del servizio sono chiamati fail-stop o fail-silent. I sistemi fail-safe, invece, sono quelli per cui i fallimenti possibili sono solamente fallimenti non catastrofici.

#### 2.1.4 I mezzi per ottenere la dependability

Lo sviluppo di sistemi dependable richiede l’utilizzo combinato di quattro tipologie di tecniche:

- **prevenzione dei guasti**, per prevenire l’occorrenza o introduzione di guasti nel sistema;
- **tolleranza ai guasti**, per erogare un servizio corretto anche in presenza di guasti;

- **rimozione dei guasti**, per ridurre il numero o la gravità dei guasti;
- **previsione dei guasti**, per stimare il numero di guasti presenti nel sistema, la loro incidenza futura, o le loro probabili conseguenze.

**Prevenzione dei guasti** La “Fault Prevention” viene effettuata ricorrendo a tecniche e processi di controllo di qualita’ sia durante la progettazione del software che durante la produzione dei componenti hardware.

**Tolleranza ai guasti** La “Fault Tolerance” mira a preservare l’erogazione di un servizio corretto in presenza di guasti attivi. Essa viene solitamente implementata tramite rilevazione di errori (error detection) e conseguente recupero dello stato del sistema (system recovery). In particolare, la rilevazione degli errori origina un segnale di errore all’interno del sistema; esistono due classi di tecniche di rilevazione di errori: concurrent error detection viene effettuata durante l’erogazione del servizio, preemptive error detection viene effettuata quando l’erogazione del servizio è sospesa e controlla la presenza di errori latenti e guasti dormienti. Il recupero dello stato del sistema trasforma uno stato che contiene uno o più errori attivi (ed eventualmente guasti), in uno stato che non contiene errori rilevati e guasti che possono essere nuovamente attivati. Il recovery consiste in error handling e fault handling. Error handling elimina gli errori dallo stato del sistema e può assumere tre forme: rollback, dove la trasformazione consiste nel ritornare ad uno stato in cui si trovava il sistema prima della rilevazione dell’errore; rollforward, dove si porta il sistema in uno stato del tutto nuovo, e compensation, dove lo stato contiene abbastanza ridondanza per eliminare la parte erronea. Fault handling

impedisce che i guasti che sono stati localizzati vengano nuovamente attivati, attraverso quattro fasi:

- **fault diagnosis** identifica l'origine della cause degli errori, in termini di locazione e tipo;
- **fault isolation** isola logicamente o fisicamente il componente, impedendogli di partecipare all'erogazione del servizio, trasformando il guasto in un guasto dormiente;
- **system reconfiguration** che riconfigura il sistema, ad esempio attivando componenti di riserva o ridistribuendo il carico tra i componenti funzionanti;
- **system reinitialization**, che esegue i controlli e gli aggiornamenti necessari in seguito alla nuova configurazione.

Solitamente l'attività di fault handling è seguita da azioni di manutenzione correttiva, che rimuovono i guasti isolati dal fault handling, ad esempio sostituendo un componente segnalato come guasto. Il fattore che distingue la fault tolerance dalla manutenzione (maintenance) è che quest'ultima richiede l'intervento di un agente esterno.

**Rimozione dei guasti** La “Fault Removal” viene effettuata sia durante la fase di sviluppo, che durante la vita operazionale del sistema. La rimozione dei guasti è uno degli obiettivi del processo di verifica e validazione (V&V). La verifica è un processo attraverso cui si determina se il sistema soddisfa alcune proprietà determinate dalle specifiche o imposte all'inizio della fase di sviluppo; se così non è si cerca di individuare il guasto che impedisce di soddisfare tali proprietà e lo si corregge. La validazione consiste invece nel controllare se il sistema soddisfa le proprie specifiche e se le specifiche descrivono adeguatamente la funzione intesa per il sistema. Le tecniche di verifica

possono essere classificate in base alla necessità di esercitare il sistema. La verifica di un sistema senza la sua esecuzione è una verifica statica, altrimenti è una verifica dinamica. La rimozione dei guasti durante la sua vita operazionale è manutenzione correttiva o preventiva. La manutenzione correttiva ha l'obiettivo di rimuovere guasti che sono stati segnalati come la causa di uno o più errori, la manutenzione preventiva cerca di scovare e rimuovere i guasti prima che causino degli errori durante la normale operazione del sistema.

**Previsione dei guasti** La “Fault Forecasting” è condotta effettuando una valutazione del comportamento del sistema rispetto all’occorrenza e attivazione dei guasti. La valutazione puo’ essere di due tipi: qualitativa, che mira ad identificare, classificare e valutare i modi di fallimento o le combinazioni di eventi che porterebbero ad un fallimento del sistema; quantitativa (o probabilistica), che mira a valutare in termini probabilistici il grado con cui alcuni attributi vengono soddisfatti dal sistema; questi attributi sono in questo caso visti come misure. Alcuni metodi di analisi sono specifici per una valutazione qualitativa o quantitativa, mentre altri possono essere utilizzati per entrambi i tipi di analisi. I due approcci principali per il fault forecasting di tipo probabilistico sono la modellizzazione e il testing. Questi approcci sono complementari: la costruzione di un modello del sistema richiede delle informazioni su alcuni processi di base del sistema, che possono essere acquisite tramite testing. Generalmente, un sistema eroga diversi servizi, e spesso esistono due o più modi di erogazione del servizio, ad esempio da servizio a pieno regime, a servizio di emergenza. Questi modi distinguono la qualità o completezza del servizio erogato. Misure di dependability collegate alla qualità del servizio erogato (performance) vengono solitamente riassunte nella nozione di *performability* [10]. I due principali approcci alla previsione dei guasti quantitativa sono la costruzione di modelli e la loro soluzione (analitica o

tramite simulazione), in cui il comportamento del sistema è riprodotto tramite un modello (tipicamente un modello stato-transizione), e la osservazione e la valutazione (anche tramite test specifici) sperimentale. Le attività di fault injection sono un esempio di valutazione sperimentale del sistema ai fini della fault forecasting, in quanto permette di esaminare l’evoluzione e le conseguenze dei guasti in un sistema.

## 2.2 Artificial Intelligence

Nel pensiero comune quando si pensa all’intelligenza artificiale, si immagina un entità in grado di pensare, che prima o poi ci sostituirà. Nella realtà l’intelligenza artificiale è quella branca dell’Informatica che dato un determinato problema, definisce degli agenti che trovano in modo efficace le migliori soluzioni. Quindi il campo di applicazione dell’intelligenza artificiale non è unico ma è suddiviso in sottodiscipline, dove si va da aree più generali come l’apprendimento e la percezione, ad altre più specializzate come il gioco degli scacchi e la dimostrazioni di teoremi matematici.

### 2.2.1 Agente razionale,Misura di prestazione,Ambiente

Nello specifico l’intelligenza artificiale si occupa di progettare **agenti razionali**, che posti in un **ambiente**, riescano a massimizzare la propria **misura di prestazione**.<sup>[8]</sup> Un **agente** è definito come un sistema che percepisce il suo ambiente attraverso dei sensori e agisce su di esso mediante attuatori. Per farsi un’idea, si può assumere che l’essere umano sia un agente, che fornito di sensori, come occhi, orecchie e altri organi, sente l’ambiente circostante, e attraverso altrettanti organi (attuatori), come mani, gambe e bocca, interagisce con l’ambiente o altri agenti. Un agente fisico, come una macchina a guida autonoma, ha sensori come: telecamere, sensori a infrarossi, e radar, mentre come attuatori: motore, freni, e tanti altri. Un’altra cosa che accomuna tutti

gli agenti, sono le **percezioni**, termine usato per indicare gli input percettivi dell'agente in dato istante.[8] La totalità delle percezioni generate dall'agente in tutta la sua storia, è definita invece come **sequenza percettiva**, dove in generale la scelta dell'azione di un agente in un qualsiasi istante può dipendere dall'intera sequenza percettiva osservata fino a quel momento. In termini matematici la rappresentazione del comportamento di un agente, è descritto dalla **funzione agente**, che descrive la corrispondenza tra una qualsiasi sequenza percettiva e una specifica azione. Se la funzione agente è la rappresentazione matematica dell'agente, con il termine **programma agente** si indica la sua implementazione concreta in esecuzione sull'architettura dell'agente. [8]

### 2.2.2 Agenti che apprendono

Si definisce un agente che apprende se migliora le proprie prestazioni nelle attività future dopo aver effettuato le osservazioni sul mondo. [8] E perché deve imparare? Un agente deve imparare per tre motivi:

- il programmatore non può anticipare tutte le possibili situazioni che l'agente si potrà trovare davanti.
- il programmatore non può anticipare tutti i cambiamenti nel tempo.
- il programmatore spesso non sanno come programmare loro la soluzione a un problema.

[8]

Detto ciò, come esistono agenti diversi, per problemi diversi, esistono apprendimenti diversi per ogni agente, che possono essere applicati alle varie componenti dell'agente. Una loro classificazione si basa su 4 fattori:

- **componente** dell'agente migliorata.
- **rappresentazione** usata per i dati e i componenti
- **prior knowledge** dell'agente presente.
- **feedback** disponibili da apprendere.

Esistono tre tipologie di feedback che determinano i tre principali tipi di apprendimento:

**unsupervised learning:** l'agente apprende patterns dall'input senza feedback esplicativi.

**reinforcement learning:** l'agente apprende da una serie di rinförzi, punizioni o premi.

**supervised learning:** all'agente sono forniti coppie di valori input-output, da cui apprende una funzione che mappi il collegamento tra input e output.

Nello specifico, il compito di un apprendimento supervisionato è quello che dato un training set di N esempi della forma input-output:

$$(x_1, y_1), (x_2, y_2), \dots (x_n, y_n)$$

dove ogni valore  $y$  è generato da un funzione sconosciuta  $y = f(x)$ , questa scopre una funzione  $h$  che approssima la funzione  $f$ . Il learning problem si può definire in due maniere a seconda dell'output  $y$ : nel caso in cui  $y$  è uno di un insieme finito di valori il problema è definito **classificazione** (binaria o booleana, nel caso di due sole scelte), mentre se  $y$  è un numero è definito come **regressione**.

### 2.2.3 Neural Network

Una particolare forma di supervised learning, come ci suggerisce il titolo del seguente capitolo, sono le artificial neural network o più brevemente neural network. La **rete neurale** è composta da un insieme di nodi o unità, connesse da link orientati.[8] La topologia e le proprietà dei nodi determinano le proprietà della rete.

**Neurone** Ciascun neurone o nodo, ha un insieme di input e produce un singolo output che può essere inviato a un gruppo di altri neuroni. Questo si attiva quando una combinazione lineare dei suoi input superano la sua hard o soft, threshold. L'output dei neuroni finali sarà il risultato della task.

**Funzione di attivazione** In generale un neurone, calcola una somma pesata dei suoi input:

$$in_j = \sum_{i=0}^n w_{i,j}a_i$$

A questa è applicata la **funzione di attivazione**  $g$ , dalla quale si deriva l'output:

$$a_j = g(in_j) = g\left(\sum_{i=0}^n w_{i,j}a_i\right)$$

Nel caso in cui la funzione di attivazione è una hard threshold, l'unità è chiamato **perceptron**, nel caso in cui questa sia una funzione logica, si usa il termine **sigmoid perceptron**.

**Collegamenti** I neuroni sono interconnessi tra di loro, al fine di propagare il risultato della funzione di attivazione di un nodo padre

a un nodo figlio. Oltre tutto ogni collegamento ha anche un peso numerico  $w_{i,j}$  associato, il quale determina la forza e il segno della connessione.

**feed-forward network** Definito il modello matematico di ogni neurone, questi si connetteranno insieme, a formare una rete. Esistono due principali tipologie: la **feed-forward network**, prevede connessioni solo in una direzione, formando un DAG, directed acyclic graph. Ogni nodo riceve gli input dai nodi superiori e invia gli output a nodi inferiori. Non sono previsti cicli. Questa tipologia di rete, rappresenta una funzione del corrente input, senza mantenere al suo interno, nessun tipo di stato se non i pesi dei collegamenti. Il **recurrent network**, prevede che i suoi neuroni usino il proprio output come feedback per il proprio input, così che i livelli di attivazione della rete formano un sistema dinamico che raggiunge uno stato stabile o esibiscono oscillazioni o addirittura comportamenti caotici. Questa caratteristica, permette alla rete di supportare un memoria a breve termine.

## 2.3 Autonomous Driving

Nell'ultimo decennio, il settore dell'Automotive è stato in continua evoluzione. In particolar modo, le grandi aziende automobilistiche e molti gruppi di ricerca si sono mossi verso l'automazione della guida ed il miglioramento delle infrastrutture già esistenti. I motivi di questo interesse del settore verso la guida autonoma sono vari:

- riduzione del numero di incidenti, anche fino al 90
- riduzione dei consumi,
- riduzione delle emissioni di  $CO_2$ ,
- riduzione di tutte le varie problematiche della sicurezza stradale.

### **2.3.1 Livelli di automazione della guida**

La SAE International, un ente di standardizzazione automobilistica, nel 2014, pubblica un nuovo standard internazionale, il J3016 ("Levels of Driving Automation"). In questo sistema di classificazione, all'aumentare del livello di automazione, la responsabilità del conducente passa dalla guida alle attività di supervisione. Lo standard suddivide l'automazione in 6 livelli, dal livello 0, in cui è assente qualsiasi tipologia di automazione, al livello 5, dove il sistema guida in maniera completamente autonoma.

- Livelli 0: Veicoli privi di qualsiasi sistema di automazione.
- Livello 1: Veicoli i quali mettono a disposizione dell'autista, almeno un sistema di assistenza, per esempio, l'assistenza alla frenata.
- Livello 2: Veicoli i quali combinano due o più sistemi avanzati di assistenza alla guida, come l'adaptive cruise control, Lane-Keeping assist, e l'Automatic Emergency Braking. Tutti questi rientrano sotto la dicitura ADAS (Advanced Driver-Assistance Systems).
- Livello 3: Veicoli definiti a guida semi-autonoma, siccome sono in grado di gestire situazioni variegate, anche al di fuori di un contesto autostradale. Nel senso che si prendono la responsabilità di gestire tratti del tragitto, richiedendo però alla fine di questi la ripresa del controllo da parte del conducente. Nel caso questo non avvenisse, arresteranno in sicurezza il veicolo.
- Livello 4: Veicoli definiti a guida autonoma, sono dotati di un set di tecnologie in grado di procedere per lunghi tragitti, superando una varietà di ostacoli, come i caselli autostradali. Tuttavia il loro funzionamento sarà limitato:

- ad una determinata area geografica,
  - condizioni meteo avverse,
  - da una velocità massima e così via.
- Livello 5 Veicoli privi di qualsiasi limite e conducente, capaci di procedere in qualsiasi condizione e luogo.

### 2.3.2 Visione artificiale

Come definito nella sezione precedente dedicata all'intelligenza artificiale, ogni agente prende in input una sensazione, la elabora, prende una decisione, e la attua con gli attuatori. Applicando questo concetto all'autonomus driving, anche questi hanno una serie di sensori, che gli permettono di prendere coscienza dell'ambiente circostante in cui è immersa la vettura. Le principali tecnologie, presenti al momento su questi sistemi sono[4]:

- Fotocamere
- Radar
- Lidar
- GPS

che insieme compongono la visione artificiale avanzata dell'autonomus driving. Tutte insieme, forniscono un modello del ambiente circostante (3D), attraverso immagini bidimensionali(2D). Lo scopo della visione artificiale è proprio quella di riprodurre la vista umana.[12]

**LIDAR - Light Detection And Ranging** Il Lidar è solito essere installato sul tettuccio del veicolo, è un sensore che attraverso l'emissione e ricezione di segnali luminosi, laser, ottiene informazioni fisiche dell'ambiente e cinematiche sul veicolo.

**RADAR - RAdio Detection And Ranging** E' un sistema che utilizza onde radio trasmesse nell'ambiente, al fine di raccogliere informazioni sugli ostacoli intorno al veicolo e aumentare la consapevolezza del posizionamento degli altri veicoli presenti. Questo sensore tiene d'occhio le altre auto e indica al veicolo autonomo di accelerare o rallentare a seconda del comportamento degli altri conducenti [4].

**Videocamera** Le telecamere sono necessarie nel sistema di trasporto intelligente affinché si riconoscano gli ostacoli rispetto alla posizione e alla velocità del veicolo in considerazione. Le immagini bidimensionali derivanti da una singola fotocamera o le mappe 3D risultanti dall'utilizzo della doppia fotocamera, potrebbero individuare in modo stereoscopico lo spazio disponibile per i movimenti autonomi del veicolo. Queste immagini o mappe vengono utilizzate per estrarre informazioni quantitative dalle scene e per tracciare gli obiettivi del veicolo. Le immagini sono segmentate in un certo numero di pixel. Ogni pixel viene elaborato e memorizzato, il che richiede un'elevata velocità di calcolo ed un elevato spazio di archiviazione [4].

**GPS - Global Positioning System** Il Sistema di Posizionamento Globale, attraverso una rete dedicata di satelliti artificiali in orbita, fornisce a un terminale mobile o ad un ricevitore GPS informazioni sulle sue coordinate geografiche e sul suo orario in quasi tutte le condizioni atmosferiche ed ovunque non vi siano ostacoli che potrebbero impedirne l'invio e la ricezione dei segnali (edifici molto alti, gallerie, ecc.), generando errori dell'ordine di metri. La scelta di sviluppare un detector sulle fotocamera dei veicoli, è data dalla loro attuale, alta presenza in tutti i veicoli.

### **2.3.3 Reti neurali applicate all'autonomus driving**

Le Reti Neurali sono oggi ampiamente accettate come soluzioni dominanti per la visione artificiale, per il riconoscimento vocale e per l'elaborazione del linguaggio naturale. Nel campo della visione artificiale, le neural network elaborano le informazioni ottenute dai sensori, e ne estraggono informazioni utili al fine di prendere decisioni sul come muoversi, garantendo la sicurezza dei viaggiatori, e degli utenti esterni al veicolo. Questo sistema presenta comunque delle criticità, come una non corretta interpretazione dell'ambiente a causa di guasti, che minano la sicurezza dei passeggeri, ma anche tutto ciò che circonda il veicolo. Per quanto riguarda le prestazioni misurate su questa tecnologia, sono stati registrati punteggi di accuratezza spesso corrispondenti a quelli di un individuo umano, su benchmark chiave. Ben diversi, invece, sono stati i punteggi osservati quando si considerano i casi peggiori. Con casi peggiori si vuole intendere quei casi definiti Adversarial Examples [5] o input perturbati, anche solo leggermente, che dal punto di vista umano risultano sempre comprensibili, mentre da quello della macchina, cambiano totalmente di significato. Naturalmente, se queste Reti vengono utilizzate su veicoli che devono muoversi in un ambiente nel quale si trova un'eterogeneità di soggetti e comportamenti, si parla di contesti nei quali affidabilità, sicurezza e tutti gli altri attributi che definiscono la dependability di un sistema sono al primo posto in ordine di importanza.

### **2.3.4 Classificatori**

Una attività importante delle reti neurali nella guida autonoma, è il processo di classificazione; ovvero l'identificazione di particolari caratteristiche intrinseche di un insieme di immagini o fotogrammi, sulle quali prenderà delle decisioni. Come detto in precedenza, le reti neurali presentano delle criticità in presenza di immagini prese in input

trasformate spazialmente [5]: con questo termine si intende rappresentare tutte le possibili trasformazioni che un’immagine può subire, ad esempio ruotandola, tagliandola, scalandola ecc. Queste sono trasformazioni naturali e sono utilizzate in vari contesti:

- verificare il livello di completezza di un classificatore,
- allenare una rete neurale,
- ricreare scenari o situazioni con cui il veicolo su cui è installata una determinata rete può avere a che fare.

Con il termine ”trasformata spazialmente” non ci si deve immaginare una modifica profonda del fotogramma in questione, ma anzi, basta anche solo una piccola rotazione, di meno di un grado, per far sì che il classificatore confonda, ad esempio, un revolver con una trappola per topi [5], il che, pensando ad un utilizzo anche diverso da quello che se ne fa su un veicolo a guida autonoma, può far sorgere molte preoccupazioni e domande.

### 2.3.5 Object Recognition

In generale, ci si riconduce al problema dell’Object Recognition o riconoscimento artificiale degli oggetti. Questo richiede che ci sia una nozione di somiglianza visiva, difficile da far apprendere ad un sistema: significherebbe catturare e insegnare ad una macchina la nozione di percezione umana [5].

Tuttavia, oggi, le DNN (Deep Neural Network) hanno raggiunto prestazioni all’avanguardia, talvolta competitive con la percezione umana. Una delle sfide principali nella guida autonoma è il cambiamento ambientale che un qualsiasi sistema autonomo può incontrare, ovvero: le differenti distanze e angolazioni con cui viene percepito l’ambiente circostante dai sistemi di visione artificiale variano ogni volta che il

veicolo si muove nello spazio [6] . Purtroppo, nel contesto della guida autonoma, anche un piccolo errore di valutazione nella navigazione potrebbe causare incidenti altamente devastanti.

## 2.4 CARLA

**Autonomus driver simulator** Lo sviluppo della guida senza conducente è un argomento attuale e di grande interesse, su cui molte società automobilistiche stanno investendo dato che promette di essere un mercato da miliardi di dollari. Di contro, la guida in ambiente cittadino è una sfida molto più grande, per via del maggiore grado di variabilità degli scenari che si possono presentare e degli oggetti che devono essere riconosciuti, ed è necessario ancora molto lavoro prima che sia disponibile un sistema in grado di muoversi da solo in questi ambienti senza pericolo e in modo efficiente. Nell'ottica dello sviluppo di un sistema di guida autonoma efficiente e sicuro, si rende necessario lo sviluppo di metodi e strumenti per il miglioramento dell'efficienza del prodotto e per la validazione dello stesso. Tra le tecniche utilizzate per automobili a guida autonoma ci sono metodi di apprendimento automatico che richiedono una gran quantità di immagini, inoltre sia per lo sviluppo che soprattutto per la validazione, si deve considerare che esistono una quantità infinita di scenari (situazioni) che si possono presentare e che il sistema progettato deve essere in grado di agire nel modo più sicuro in ognuno di essi. Il problema è di per sé mal posto, visto che non è dato sapere se un'auto che reagisce correttamente in una certa situazione farà lo stesso cambiando qualche variabile dello scenario. Lo sviluppo e la validazione delle auto a guida autonoma si configura per quanto detto come molto complicato e costoso, visto che è necessario effettuare un gran numero di prove in diversi scenari. Per aiutare a risolvere il problema dei costi dello sviluppo con testing in ambiente fisico e della richiesta di dati che abbiamo presentato, sono stati sviluppati dei simulatori che permettono di sperimentare il soft-

ware sviluppato in un ambiente virtuale, tra questi NVIDIA DRIVE Constellation, AirSim, e CARLA Simulator. Questi simulatori riproducono scenari di guida cittadina in cui è possibile testare un auto a guida autonoma che raccoglie dati dall'ambiente. Per le considerazioni fatte in precedenza, un simulatore si presenta come un alleato fondamentale nel processo di sviluppo e di validazione. Infatti un simulatore in grado di riprodurre realisticamente il complicato ambiente cittadino può essere utilizzato per lo sviluppo con tecniche di machine learning avide di dati, per cui sono necessari centinaia di migliaia di riproduzioni per ottenere risultati, e per testare il sistema in altrettanti scenari, anche eticamente o fisicamente difficili da riprodurre, prima o congiuntamente allo sviluppo e al testing su strada.

In particolare CARLA è un simulatore grafico open source costituito da un'architettura client-server scalabile. dove il server è responsabile di tutto ciò che riguarda la simulazione dell'ambiente in tutti i suoi aspetti: rendering del sensore, calcolo della fisica, aggiornamenti sullo stato del mondo e sui suoi attori e molto altro. Mentre il lato client consiste in una somma di moduli client che controllano la logica degli attori sulla scena e impostano le condizioni del mondo.

#### 2.4.1 Sensori

Come un veicolo reale di guida autonoma, Carla predispone una suite di sensori preimpostasti, come fotocamere RGB e pseudo sensori che forniscono ground-truth depth e ground-truth semantic segmentation . La loro posizione, numero e direzione possono essere specificate dal client. Per una presentazione completa dei sensori si rimanda a [tesi niccolo galli] E proprio della fotocamera RGB di Carla si farà uso al fine di costruire il dataset di immagini.

## 3 Costruzione del dataset

Il detector utilizza come tecnologia una Convolutional Neural Network, una forma di rete neurale. Un vantaggio dell'utilizzo di software di simulazione come CARLA, oltre a permettere un abbattimento dei costi di ricerca, permette di avere a disposizione grandi quantità di dati, utili al training dell'agente. Infatti per l'apprendimento è stato usato il paradigma del supervised learning. L'addestramento di una generica rete neurale prevede la creazione di un dataset sulla base della procedura:

- Definizione delle classi
- Acquisizione immagini
- Sporcatura immagini
- Suddivisione del dataset

Ciascuno step è illustrato in dettaglio nei paragrafi seguenti.

### 3.1 Definizione delle classi

Il problema della classificazione delle immagini è un problema che come dice il verbo, deve assegnare a un elemento in input una classe. Quindi le classi, o etichette(label) sono gruppi di elementi che hanno determinate caratteristiche in comune. Più le differenze tra classi diverse, sono poco accentuate, più è difficile distinguerle. Nel caso del detector, le classi rispecchiano i diversi guasti che la fotocamera può subire. Per una esposizione completa dei guasti che sono stati presi in considerazione si rimanda a [9]. Le classi prese in considerazione per la costruzione dei vari dataset su cui sarà allenato l'agente, sono 18, di cui una rappresenta le immagini pulite(Golden Run), 16 rappresentano

ciascuna un determinato guasto, e l'ultima (All) contiene al suo interno tutte e 16 i guasti presi in considerazione

- Golden Run
- Blur
- Black
- Brightness
- 50 death pixels
- 200 death pixels
- Nodemos
- Noise
- Sharpness
- Brokenlen
- Icelens
- Banding
- Greyscale
- Condensation
- Dirty lens
- Chromatic aberration
- Rain
- All

## 3.2 Acquisizione

Definite le classi, la fase successiva nella costruzione dei dataset, è l'acquisizione delle immagini. Per questa fase entra in gioco CARLA. Avviato il suo lato server, il lato client è stato impostato per generare 517 simulazioni diverse, ovvero 517 ambienti generati casualmente, con condizioni atmosferiche diverse, e in localizzate in luoghi diversi sulla stessa mappa. In ciascuna delle simulazioni il veicolo si muove nelle vie della città, seguendo percorsi randomici. Durante il tragitto il Client è stato impostato per acquisire 300 immagini dal sensore della fotocamera frontale della vettura, nel formato .png e ciascuna dalle dimensioni di 800\*600 pixels. Si rimanda alla sezione "Manuale Utente" per le note sugli script utilizzati

## 3.3 Sporcatura

Ottenute queste grande mole di immagini pulite, prive di qualsiasi "errore", questa fase del processo, prevede l'applicazione dei guasti alle

immagini. Si intende l'applicazione di filtri, che simulino il guasto alla fotocamera, della corrispettiva classe. I filtri sono ripresi dal progetto github di Francesco Secci, Python Image Failures, i quali sono stati riadattati per l'uso nella seguente tesi.

**Rappresentazione dei vari guasti con breve descrizione** Per una illustrazione completa dei guasti si rimanda a [9], da cui è stato ripreso lo studio su l'effetto di determinati guasti alla fotocamera, portino a un rate di fallimenti più elevato. Di seguito si riporta una breve rappresentazione della stessa immagini, acquisita dalla fotocamera di un veicolo nel simulatore CARLA, a cui è applicata i diversi filtri che simulano i guasti impiegati nel progetto.

### 3.4 Suddivisione del dataset

La Convnet sviluppata, è un classificatore binario, e come tale distingue solamente tra due classi, immagini pulite e immagini sporche, ovvero: golden run e malfunzionamento. Detto ciò sono stati costruiti 17 dataset diversi, ciascuno per ogni classe. Ogni dataset è composto da 314400 elementi, di cui  $\simeq 75\%$  come training set,  $\simeq 20\%$  come validation set, e il restante  $\simeq 5\%$  come test set. Ognuno di questi 3, è a sua volta equamente suddiviso nelle due classi che lo compongono.

Questa suddivisione tra training set e validation set delle immagini, risulta molto importante nella fase di training dei modelli, come si vedrà in seguito. Uno schema di suddivisione (o immagine):

- Training set: 240000 elementi
  - Golden Run: 120000 elementi
  - Dirty Run: 120000 elementi
- Validation set: 60000 elementi

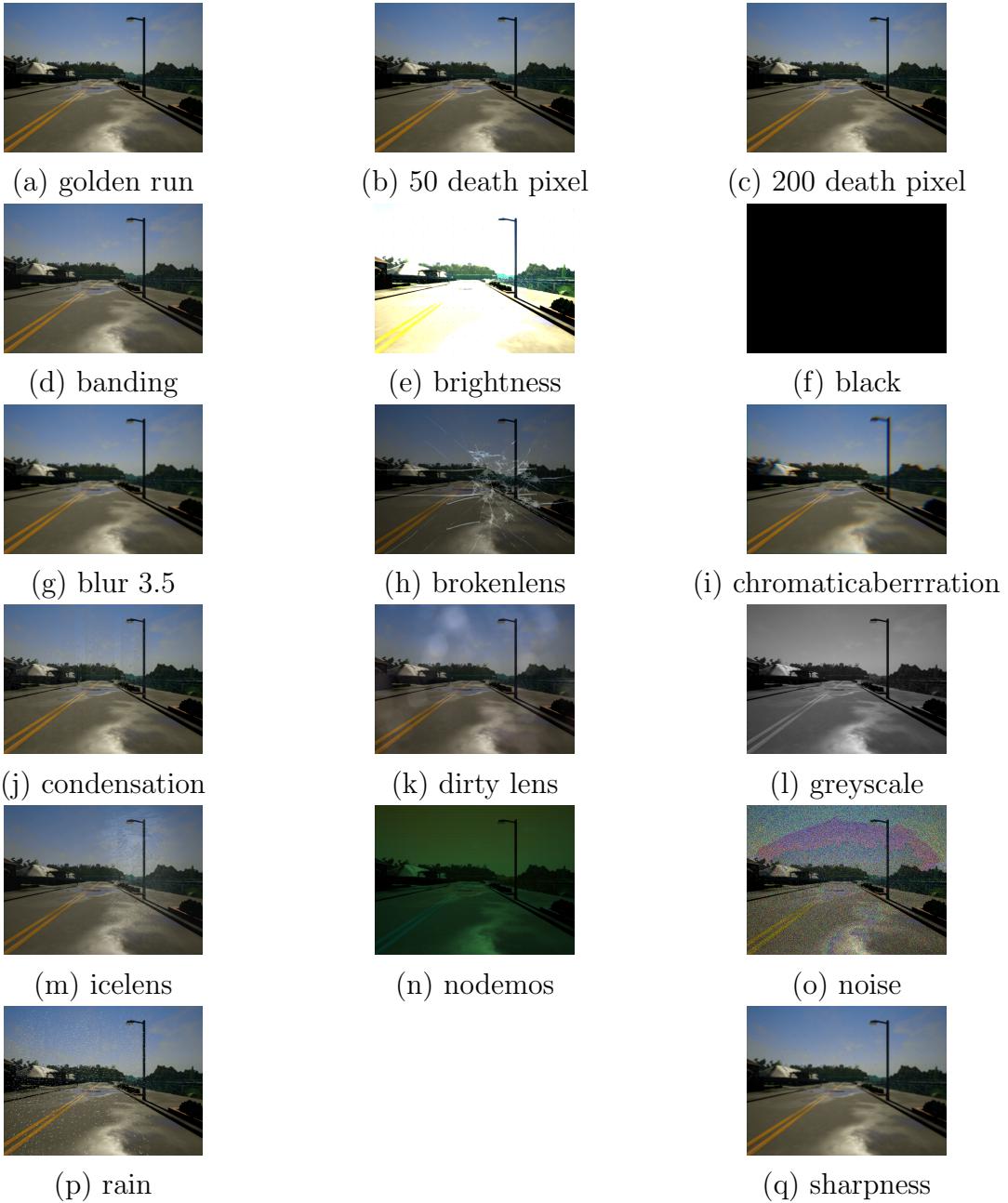


Figure 1: Esempi dei filtri applicati

- Golden Run: 30000 elementi
- Dirty Run: 30000 elementi
- Test set: 14400 elementi
  - Golden Run: 7200 elementi
  - Dirty Run: 7200 elementi

## 4 Costruzione del detector

Lo scopo del detector, è quello di individuare all'interno del flusso di immagini della fotocamera, la presenza di distorsioni, ovvero riuscire a prevenire che un guasto al sensore della fotocamera, arrivi al sistema decisionale, scaturendo un possibile fallimento. Per perseguire il suo scopo, il detector deve essere in grado di classificare ciascuna immagine del flusso, tra immagini pulite, ovvero immagini che non generano fallimenti nel sistema,e immagini sporche, generate da un guasto.Questa distinzione binaria, tra le due sole tipologie di immagini, scaturisce dal fatto che non è richiesta l'identificazione precisa di un singolo guasto,(multiclassificatore), ma si richiede solo la distinzione tra elemento buone e elemento cattive. L'elemento classificatore è svolto da una rete neurale convoluzionale o ConvNet. Come dice il nome, la ConvNet è una tecnologia basata sulle reti neurali più generiche, ed è già ampiamente impiegata nei sistemi di riconoscimento grafico, come classificatori, object detection oltre a vocal recognition.

### 4.1 Convolutional Neural Network

Una rete neurale convoluzionale è una rete neurale di tipo feed-forward network dove la strato convoluzionale compare almeno una volta. L'input

di una convolutional neurale network o ConvNet,, è una immagine, ossia una matrice di valori di ogni singolo pixel occupante una precisa posizione all'interno dell'immagine.Nel caso di immagini RGB, questa sarà descritta da un terna di matrici dell'intensità dei colori primari (rosso,verde,blu), nel caso di immagini in bianco e nero, queste sono descritte da una singola matrice. Le matrici che compongono le immagini, identificano i **canali** per la rete. Questa con le dimensioni dell'immagine,(height e width) compongono l'input della rete.

#### 4.1.1 L'architettura

Le reti neurali convoluzionali operano su strutture a griglia contraddistinte da relazioni spaziali tra pixel, ereditate da uno strato al successivo tramite valori che descrivono piccole regioni locali dello strato precedente. L'insieme di matrici degli strati nascosti (hidden), risultato della convoluzione o di altre operazioni, è definita feature map o activation map, i parametri addestrabili sono tensori denominati filtri o kernel. Una CNN è composta da un insieme di strati che si susseguono, illustrati in dettaglio nei paragrafi seguenti. Gli strati principali sono:

- convolutional layer;
- activation layer;
- pooling layer;
- dense layer;

[11] Di seguito la struttura sviluppata per il progetto:

- conv2d
- max pooling2d
- conv2d

- max pooling2d
- conv2d
- max pooling2d
- flatten
- dense
- dense

#### 4.1.2 Convolutional layer

La convoluzione è l'operazione fondamentale delle ConvNet. Essa dispone un filtro in ogni possibile posizione dell'immagine, coprendola interamente, e computa il prodotto scalare tra il filtro stesso e la matrice corrispondente del volume di input, avente eguale dimensioni. È possibile visualizzare la convoluzione come una sovrapposizione del kernel sull'immagine in input (o strato nascosto). [1] Un filtro è caratterizzato dai seguenti iperparametri (hyperparameters):

- altezza  $F_q$
- larghezza  $F_q$
- profondità  $d$
- numero  $q$

Solitamente i filtri hanno forma quadrata e profondità uguale a quella dello strato al quale sono applicati, nel caso di immagini RGB, 3. Il numero di possibili allineamenti tra filtro e immagine definisce altezza e larghezza della successiva feature map. Dal hidden layer  $q_1$ , il filtro produce uno nuovo hidden layer  $q_2$   $L_{q+1} = L_q - F_q + 1$   $B_{q+1} = B_q$

-  $F_q + 1$  Ad esempio, per immagini di dimensioni  $32 * 32$ , un filtro  $5 * 5$  genera uno strato nascosto  $28 \times 28$ . È opportuno fare distinzione tra profondità del filtro e profondità di strato nascosto/mappa di attivazione: la prima,  $d$  è la stessa dello strato al quale è applicato, la seconda deriva invece dal numero di filtri applicati. Il numero di filtri è un iperparametro definito sulla base della capacità di distinguere forme sempre più complesse che si vuole conferire alla rete. I filtri sono dunque i componenti a cui saranno associate le caratteristiche dei pattern delle immagini. I filtri dei primi strati individuano forme primitive, quelli successivi imparano a distinguere forme sempre più grandi e complesse. Una proprietà della convoluzione è l'**equivarianza alla traslazione**: immagini traslate sono interpretate allo stesso modo e i valori delle mappa di attivazione traslano con i valori di input [11]. Ciò significa che forme particolari generano feature maps simili, indipendentemente dalla loro collocazione nell'immagine. Una delle proprietà della convoluzione è la seguente: Una convoluzione sullo strato  $q$  incrementa il campo recettivo di una feature dallo strato  $q$  allo strato  $q + 1$ . In altre parole, ogni valore della mappa di attivazione dello strato successivo cattura una regione spaziale più ampia del precedente. Feature maps degli strati catturano aspetti caratteristici di regioni via via maggiori e questo è il motivo per cui le CNN possono essere definite “profonde”: per studiare l’intera immagine sono necessarie lunghe successioni di “blocchi” di strati.

**Padding** L’operazione di convoluzione comporta una contrazione dello strato  $q$  rispetto a  $q + 1$  ed una conseguente perdita di informazioni. Il problema può essere arginato utilizzando il cosiddetto padding, una tecnica che prevede l’aggiunta di  $\frac{(F_{meno1})}{2}$  pixel ai bordi delle mappe di attivazione per mantenere l’impronta spaziale. Ovviamente, per non alterare l’informazione, ai pixel sono assegnati valori nulli. Il risultato [1] è un incremento delle dimensioni (altezza e

larghezza) del volume di input di  $F - 1$ , esattamente la quantità di cui è ridotto a seguito della convoluzione. Essendo il prodotto zero, le regioni esterne soggette a padding non contribuiscono al risultato finale del prodotto scalare. Ciò che invece accade è permettere al filtro convoluzionale di scavalcare i bordi dello strato e computare il prodotto scalare solamente per le celle di valori diversi da 0. Questa tipologia di padding è definita “half-padding” in quanto circa metà del filtro oltrepassa i bordi, quando collocato alle estremità. L’half-padding è utilizzato per mantenere il “footprint” spaziale. Quando il padding non è utilizzato, si parla semplicemente di valid-padding e nella pratica non dà buoni risultati per il seguente motivo: mentre con l’half-padding le celle ai bordi contribuiscono all’informazione, nel caso di valid-padding, queste non vedono il passaggio del filtro e sono sottorappresentate. Un’altra forma di padding è il full-padding, con il quale si lascia che il filtro esuli completamente dal layer andando ad occupare celle di soli zeri. Così facendo si incrementa l’impronta spaziale dello strato, allo stesso modo in cui il valid-padding la riduce.[1]

#### 4.1.3 Funzione di attivazione Relu

L’operazione di attivazione non-lineare segue l’operazione di convoluzione. Per ogni strato  $L_q \times B_q \times d$ , la funzione di attivazione genera uno strato di eguale dimensione  $L_q \times B_q \times d$  di valori limitati da soglie: in quanto semplice mappatura uno-a-uno dei valori di attivazione, la funzione ReLU non altera l’impronta spaziale dello strato. L’attivazione avviene tramite funzioni matematiche. Mentre in passato la tangente iperbolica, la funzione sigmoide, softsign godevano di ampia diffusione, ora sono limitate a reti non-profonde e ormai rimpiazzate dalla funzione di attivazione ReLU (Rectified Linear Unit). Il motivo principale è che in reti neurali profonde, il gradiente di queste tre funzioni di attivazione si annulla durante la retropropagazione ed impedisce all’algoritmo di proseguire con l’addestramento. Inoltre, la funzione

ReLU è computazionalmente molto più efficiente.

#### 4.1.4 Pooling layer

Il max-pooling estrae il massimo valore contenuto in matrici  $P \times P$  di ogni mappa di attivazione e produce un altro hidden layer di eguale profondità. Anche in questo caso, come per la convoluzione, si fa uso di stride: se lo stride è 1 lo strato 2 così generato avrà dimensioni:  $L_2 = L_1 - P + 1$   $B_2 = B_1 - P + 1$   $d_2 = d_1$ . Per stride maggiori di 1, come solitamente si usa, altezza e larghezza saranno, rispettivamente:  $L_2 = L_1 - P/S + 1$   $B_2 = B_1 - P/S + 1$ . Rispetto alla convoluzione, il pooling è effettuato al livello di ciascuna mappa di attivazione, quindi il loro numero rimane alterato e l'output è uno strato della stessa profondità (ma di altezza e larghezza differenti).

Una configurazione tipica è dimensione 2x2 e stride 2: così facendo non c'è sovrapposizione tra regioni. L'uso dello stride nel pooling è importante per tre motivi. Il primo è la riduzione dell'impronta spaziale delle mappe di attivazione, il secondo è un certo grado di invarianza alla traslazione e il terzo è un incremento del campo ricettivo. Si osservi che per ridurre l'impronta spaziale possono essere utilizzati unicamente strati convoluzionali con stride maggiori di 1. Nonostante ciò, si preferisce tuttora utilizzare max-pooling o qualche altra variante dato il grado di non-linearietà e invarianza alla traslazione che introducono. [1]

#### 4.1.5 Dense layer

Connette ogni feature in input con la corrispondente feature in output. Presenta la struttura di una rete feed-forward tradizionale e aumenta a dismisura il numero di parametri addestrabili per effetto del numero di connessioni. Ad esempio, se due strati completamente connessi hanno 4096 unità ciascuno, il numero di connessioni (quindi di parametri) sarà superiore a 16 milioni. [1]

## 4.2 La rete

La struttura della rete sviluppata nel seguente progetto è mostrata in seguito: (temporanea)

```
Instructions for updating:  
Use tf.where in 2.0, which has the same broadcast rule as np.where  
Model: "sequential"  


| Layer (type)                 | Output Shape         | Param #   |
|------------------------------|----------------------|-----------|
| conv2d (Conv2D)              | (None, 800, 600, 16) | 448       |
| max_pooling2d (MaxPooling2D) | (None, 400, 300, 16) | 0         |
| conv2d_1 (Conv2D)            | (None, 400, 300, 32) | 4640      |
| max_pooling2d_1 (MaxPooling2 | (None, 200, 150, 32) | 0         |
| conv2d_2 (Conv2D)            | (None, 200, 150, 64) | 18496     |
| max_pooling2d_2 (MaxPooling2 | (None, 100, 75, 64)  | 0         |
| flatten (Flatten)            | (None, 480000)       | 0         |
| dense (Dense)                | (None, 512)          | 245760512 |
| dense_1 (Dense)              | (None, 1)            | 513       |


```
Total params: 245,784,609  
Trainable params: 245,784,609  
Non-trainable params: 0
```


```

## 4.3 Addestramento del rete convuluzionale

Dopo aver definito la struttura della rete, e avendo suddiviso il dataset in tre parti: training set, validation set e test set, la fase successiva prevede l'addestramento della rete. Per addestramento di una rete si intendo, elaborazione da parte della rete del training set, ovvero la modifica dei pesi dei collegamenti, al fine di modellare il meglio possibile la rete sui dati. In generale il sistema di apprendimento prevede

che sia utilizzato il training set per l’addestramento della regressione logica, ovvero una modifica dei pesi

Durante l’addestramento della CNN, il set di dati deve essere suddiviso in tre parti: il training set, il validation set e il test set. Il training set viene utilizzato per addestrare la CNN utilizzando la regressione logistica, che addestrerà la rete neurale convoluzionale, che sarà spiegato di seguito. Dopo ogni round di training, la rete è testata su una parte casuale del validation set. Il risultato viene utilizzato per valutare l’accuratezza della rete nel rilevare gli oggetti corretti, al fine di determinare se i pesi vengono regolati nel modo migliore. Il test set viene utilizzato per valutare le prestazioni della CNN dopo che il processo di formazione è terminato. Tuttavia, può essere pericoloso modificare la rete in base ai risultati utilizzando il set di test, a causa del rischio di overfitting. La dimensione esatta dei diversi set varia a seconda sul set di dati fornito, ma di solito il set di addestramento è composto da circa il 75-80% dei dati totali, mentre la convalida e i set di test utilizzano entrambi metà dei dati rimanenti, circa 10- 12,5% ciascuno [**guyon1997scaling**].

Avendo predisposto la rete, e i vari dataset, il passo successivo è il suo addestramento.

#### 4.3.1 Batch

Siccome le reti richiedono grosse quantità di dati per essere allenate, si definiscono due modi per fare ciò: la prima prevede l’uso di una infrastruttura hardware in grado di processare l’intera mole tutta insieme nello stesso momento, e questo può risultare effettivamente difficile, mentre il secondo modo prevede di dividere il dataset in porzioni, e fornirle una per una alla rete. Le dimensioni di queste porzioni, è definita come **Batch Size**. Il numero di batch necessari a processare l’intero dataset, definisce il numero di iterazioni.

### **4.3.2 Epochs**

Ultima componente fondamentale di un addestramento, sono le Epochs. Dato il dataset di 300000 elementi, e un batch size di 4, si avranno 75000 iterazioni per epoch, quindi un epoch è l'intero ciclo di elaborazione di un training set, e le epochs identificano quante epoch saranno eseguite. Questo elemento è importante, perchè ad ogni epoch mantenendo il dataset costante, modifica i pesi della rete, portando la curva da uno stato underfitting, a uno ottimale, a un overfitting. Il numero di epoch necessario a ottenere una curva ottimale varia da dataset a dataset

## **4.4 Testing della rete**

## **5 Esecuzione e risultati**

## **6 Conclusioni e lavori futuri**

## **7 A Manuale utente**

# References

- [1] Charu C Aggarwal et al. *Neural networks and deep learning*. Springer, 2018.
- [2] Algirdas Avizienis et al. “Basic concepts and taxonomy of dependable and secure computing”. In: *IEEE transactions on dependable and secure computing* 1.1 (2004), pp. 11–33.
- [3] Andrea Bondavalli. *L’analisi quantitativa dei sistemi critici: Fondamenti e Tecniche per la Valutazione-Analitica e Sperimentale-di Infrastrutture Critiche e Sistemi Affidabili*. Società Editrice Esculapio, 2011.
- [4] Plaban Das. “Risk analysis of autonomous vehicle and its safety impact on mixed traffic stream”. In: (2018).
- [5] Logan Engstrom et al. “Exploring the landscape of spatial robustness”. In: *International Conference on Machine Learning*. 2019, pp. 1802–1811.
- [6] Kevin Eykholt et al. “Robust physical-world attacks on deep learning visual classification”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1625–1634.
- [7] David M Nicol, William H Sanders, and Kishor S Trivedi. “Model-based evaluation: from dependability to security”. In: *IEEE Transactions on dependable and secure computing* 1.1 (2004), pp. 48–65.
- [8] Stuart J Russell and Peter Norvig. *Intelligenza artificiale. Un approccio moderno*. Vol. 1. Pearson Italia Spa, 2005.
- [9] Francesco Secci and Andrea Ceccarelli. “On failures of RGB cameras and their effects in autonomous driving applications”. In: *arXiv preprint arXiv:2008.05938* (2020).

- [10] RM Smith, Kishor S. Trivedi, and AV Ramesh. “Performability analysis: measures, an algorithm, and a case study”. In: *IEEE Transactions on Computers* 37.4 (1988), pp. 406–417.
- [11] Luca Torresin. “Sviluppo ed applicazione di reti neurali per segmentazione semantica a supporto della navigazione di rover marziani”. In: (2019).
- [12] Wikipedia. *Visione artificiale — Wikipedia, L'enciclopedia libera*. [Online; in data 15-ottobre-2020]. 2020. URL: [http://it.wikipedia.org/w/index.php?title=Visione\\_artificiale&oldid=111327669](http://it.wikipedia.org/w/index.php?title=Visione_artificiale&oldid=111327669).