Tesi di Laurea

Pietro Bernabei - matricola:6291312

Anno Accademico 2019/20

Contents

1 Introduzione

1.1 Motivazioni

Nel giro di un paio di secoli il mondo, è passato 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, diventandone dipendenti a tal punto da richiedere che questi siano continuativi, privi da possibili errori o malfunzionamenti.

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 con 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 riperquote nel sistema decisionale, errori non molto graditi al guidatore.

1.2 Obiettivo

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

1.3 Organizzazione del lavoro

Avendo definito il detector, come un sistema in grado di rilevare una variazione malevola nel flusso di immagini acquisite dalla fotocamera, questo dovrà essere in grado di classificare l'immagine sottoposta a

decisione. Per questo la soluzione descritta propone di impiegare la tecnologia dell'Intelligenza Artificiale, e nello specifico la rete neurale convuluzionale (NNC), per il suo sviluppo. Come verrà esposto meglio in seguito, una NNC è una particolare forma di supervisioned learning, rivolta alla ...(classificazione) . I supervisioned learning, e quindi anche NNC richiedono una fase di training, nella quale viene definita la propria conoscenza di base (Knoledge base), sulla quale poi prenderà le proprie decisioni. Detto ciò lo sviluppo del detector si divide in:

- Definizione e generazione Dataset
- Creazione CNN
- Training CNN

2 Fondamenti

2.1 Sistemi Critici

Ogni giorno, una persona usa infrastutture, mezzi, telecumicazioni, e servizi di qualsiasi genere, affidandosi totalmente al loro funzionamento, alla loro continuità, dando per scontato che non possano subire guasti o malfunzionamenti, perchè 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: [1].

- **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.[2]

Una definizione alternativa, che stabilisce un criterio per decidere se un determinato servizio è dependable, definisce la **dependability** di un sistema come la capacità di evitare fallimenti che siano più frequenti e più severi del limite accettabile [1].

Ciò di cui la definizione precedente non tiene conto, è che il comportamento del nostro sistema, sia sempre definito nella totalità, non ambiguo e precisa. Per questo la dependability può essere vista come una misura di quanta fiducia possiamo riporre in modo giustificato sul servizio erogato dal sistema stesso.[2]

Un'esposizione sistematica dei concetti relativi alla dependability consiste di tre parti:

- **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:[2]

- **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.

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" [4].

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, 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. Il grado con cui un sistema possiede questi attributi deve essere interpre- tato in senso probabilistico e non in senso assoluto, deterministico: a causa dell'inevitabile occorrenza dei guasti i sistemi non sono mai totalmente di- sponibili, affidabili, safe o secure. Per questo gli attributi di dependability possono essere definiti in senso probabilistico cosi da poterli trattare in modo quantitativo. Ad esempio:

- L'affidabilità può essere rappresentata dalla probabilità che il sistema non fallisca durante il periodo di missione del sistema. Se si assumono distribuzioni esponenziali, possiamo rappresentare l'affidabilità tramite il tasso di fallimenti (ad esempio, in numero medio di fallimenti all'ora).

- La disponibilità è la probabilità che il sistema sia operativo al tempo t, considerando l'alternanza fra gli stati di servizio corretto e servizio non corretto.

- La manutenibilità può essere rappresentata dalla velocità con cui viene ripristinato un servizio corretto dopo un fallimento.

**guarda se aggiungere ultima parte con le formule

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 gravita' 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. Queste tecniche comprendono ad esempio la programmazione strutturata e modulare, l'uso di linguaggi fortemente tipati, di editori guidati dalla sintassi e compilatori certificati per quanto riguarda il software, mentre per quanto riguarda l'hardware l'uso di rigorosi processi produttivi e di strumenti per la progettazione come linguaggi di alto livello (VHDL). Guasti di origine fisica vengono prevenuti tramite specifiche prote- zioni, ad esempio dalle radiazioni e interferenze elettromagnetiche. Guasti che originano da interazioni umane possono invece essere prevenuti tramite un'appropriata formazione del personale, o la creazione di procedure di manu- tenzione rigorose. Guasti originati da attacchi esterni possono essere prevenuti tramite firewall e dispositivi di sicurezza simili.

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 atti- vando componenti di riserva o ridistribuendo il carico tra i componenti funzionanti;
- **system reinitialization**, che esegue i controlli e gli aggiornamenti ne- cessari 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) e' 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 cosi 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 necessita' 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 comporta- mento del sistema rispetto all'occorrenza e attivazione dei guasti. La valuta- zione 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. Misu- re di dependability collegate alla qualita' del servizio erogato (performance) vengono solitamente riassunte nella nozione di performability [4]. 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 al'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**.[5] 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, come mani,gambe e bocca, interagisce con l'ambiente o altri agenti.(attuatori) Un agente fisico, come una macchina a guida autonoma, ha sensori come: telecamere, sensori a infrarossi,radar e tanti altri, mentre come attuatori: motori,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.[5] La totalità delle percezioni generate dall'agente in tutta la sua storia, è definità invece come **sequenza percettiva**, dove in generale la scelta dell'azione di un agente in un qualsiasi istante puo 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. [5]

2.2.2 Agenti che apprendono

Si definisce un agente sta imparando se migliora le proprie prestazioni nelle attività future dopo aver effettuato le osservazioni sul mondo. [5] E perchè deve imparare? Un agente deve impare 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 programmatori spesso non sanno come programmare loro la soluzione a un problema.

[5]

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

unsupervised learning , dove un agente apprende patterns dall'input senza feedback espliciti.

reinforcement learning , dove un agente apprende da una serie di rinforzi, punizioni o premi.

supervised learning , dove ha un agente sono forniti coppie di valori input-output e apprende una funzione che mappi dall'input all'output. Nello specifico, il compito di un apprendimento supervisionato è quello che dato un training set di N esempi della forma input-output

(x1.y1),(x2,y2),...(xn,yn), 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 del l'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 si definisce come **regressione**

2.2.3 Neural Network

Una particolare forma di supervised learning, come ci suggerisce il titolo del seguite 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.[5] La topologia e le propietà dei nodi determina 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 quando una combinazione lineare dei suoi input superano la sua hard o soft threshold. L'output dei neuroni finali sarà il risultato della task.

Collegamenti Un link tra una unita i e un unità j serve a propagare l'attivazione a_i da i a j. Oltretutto ogni collegamento ha anche un peso numerico $w_{i,j}$ associato, il quale determina la forza e il segno della connessione.

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:

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

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**.

feed-forward network Definito il modello matematico di ogni neurone, questi dovranno connettersi 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, cosi 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.

perceptron network : Una forma semplice di feed-forward network, è il perceptron network o anche chiamato single-layer neural networl. Questo prevede che i suoi neuroni di input siano direttamente connessi a quelli di output.

2.2.4 Convulutional Neural Network

3 Costruzione del dataset

Come predetto il detector ivi esposto impiega come sistema classificatore una rete neurale convuluzionale, ed essendo questo una forma di apprendimento supervisionato, richiede una fase di training prima del suo utilizzo, nel quale si va a fornirgli una conoscenza di base su cui si baserà. Detto ciò, la prima cosa che è necessario predisporre, è il dataset di dati, nel nostro caso, immagini, da fornirgli. Nel caso di questa tesi, i detector sono implementati attraverso singoli classicatori binari, dove ciascuno è stato allenato ad apprendere singoli casi di malfunzionamento. Come spiegato precedentemente, le reti neurali prevedono due diversi dataset: il training set e il validation set. Entrambi suddivisi in golden run e sporcate. Quindi il database avrà la seguente forma generale a tutte:

- Training set
  - Originale
  - Sporcata
- Validation set
  - Originale
  - Sporcata

Dovendo ottenere un simil risultato, si è proceduto con le due seguenti fasi:

- Acquisizione;
- Sporcatura;

3.1 Acquisizione

La prima fase del processo di costruzione del dataset, prevede l'acquisizione di immagini pulite dal simulatore CARLA. Questo avviene grazie a un progetto di "inserisci nome del progetto github", il quale avvia n simulazioni diverse di guida autonoma. Di ciascuna di queste salva un numero fisso di immagini, acquisite dal sensore della fotocamera RGB della macchina. Nel caso di questo progetto il numero di simulazioni avviate è stato di 500, da cui per ciascuno sono state prodotte 300 immagini (800*600)

3.2 Sporcatura

La seconda fase del processo di costruzione, prevede la "sporcatura" delle immagini salvate nella prima fase, immagazzinandole in maniera tale da essere poi usate dall'IA. Per "sporcatura" di una immagine, è inteso l'applicazione di un effetto all'immagine che ne simuli un guasto al sensore RGB della vettura. Per fare questo è stato usato il progetto github "progetto secci", adattando le funzioni al progetto. I diversi effetti utilizzati nel progetto sono le seguenti: - - - - - differenza tra death pixel 50 e death pixel 200

4 Costruzione del detector

La costruzione del detector, è costituita da due fasi:

References

[1] 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.

[2] 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.

[3] EN CENELEC. "50126-railway applications: The specification and demonstration of reliability, availability, maintainability and safety (rams)". In: *European Committee for Electrotechnical Standardization* 6 (1999).

[4] 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.

[5] Stuart J Russell and Peter Norvig. *Intelligenza artificiale. Un approccio moderno*. Vol. 1. Pearson Italia Spa, 2005.