Tesi di Laurea

Pietro Bernabei - matricola:6291312

Anno Accademico 2019/20

Contents

1 Introduzione

1.1 Motivazioni

L'avanzamento tecnologico vissuto nell'ultimo periodo, ha portato ha un cambiamento nelle nostre vite, con l'introduzione di nuovi sistemi, applicazioni e infrastrutture, hardware e software, sulle quali si fa sempre più affidamento. Data questa particolare dipendenza dalla loro presenza, si richiede che questi siano continuativi, liberi da possibili errori o malfunzionamenti. Questi sono definiti Sistemi critici. Un particolare sistemi critico, che verrà trattato da questa tesi, è il sistema di guida autonoma di una macchina. Tecnologia in sviluppo negli ultimi anni prevede l'uso di sensori, per interagire e sentire il mondo esterno. Uno dei più importanti e comuni sensori usati all'interno dei sistemi a guida autonoma, sono le telecamere."Dispositivo elettronico in grado di acquisire immagini bidimensionali in sequenza a velocità di cattura prefissate, solitamente nella gamma visibile dello spettro elettromagnetico"["wikipedia-telecamere"]". Queste possono subire diversi malfunzionamenti, come la sfuocatura, il congelamente del vetro, la rottura dei pixels, e altri ancora, i quali causano errori nel momento decisionale.

1.2 Obiettivo

Detto ciò, l'obbiettivo della seguente tesi è di andare a definire un sistema software, detto Detector, in grado di andare a rilevare, nel flusso di immagini generate dalla telecamara di un mezzo a guida autonoma, la presenza di malfunzionamenti nel sistema di acquisizione.

1.3 Organizzazione del lavoro

Per lo sviluppo del seguente progetto si sono definite due fasi: Acquisizione immagini Sviluppo CNN Addestramento CNN **Essendo troppo costoso disporre di un mezzo fisico, per lo sviluppo e il testing di sistemi software per la tipologia di progetto, sono stati sviluppati simulatori software di sistemi a guida autonoma. Nello sviluppo del seguente progetto, è stata impiegata la piattaforma simulativa CARLA nella sua versione 0.8.4.

Dato questo dataset, **Il Detector sviluppato nella seguente tesi, è un software che si basa su di una intelligenza artificiale, CNN (convulutional Neureal Network), addestrata con dataset di immagini, da poi essere inserità poi all'interno

2 Fondamenti

2.1 Sistemi Critici

Il rapido avanzamento tecnologico a cui assistiamo ogni giorno ha introdotto nella nostra vita una serie di infrastrutture, sistemi e applicazioni su cui facciamo affidamento per svolgere le nostre attività quotidiane. L'informatizzazione si è introdotta in modo cosi pervasivo nelle attività umane e molti servizi sono percepiti come parte integrante dell'ambiente in cui siamo immersi. Una loro eventuale indisponibilità, interruzione o malfunzionamento ci lascia quantomeno sorpresi. Alcuni di questi sistemi e infrastrutture possono inoltre essere considerati critici perché la loro distruzione o un'interruzione del loro funzionamento potrebbe avere effetti catastrofici sulla sicurezza, sull'economia sull'ambiente o sulla salute, siano esse del singolo individuo o della società intera. In questo senso, sono ad esempio infrastrutture critiche le telecomunicazioni, i trasporti, il sistema elettrico, gli acquedotti, la finanza, la distribuzione del gas; sono esempi di sistemi critici il sis-

tema di pilotaggio di un treno o di un aereo, il controllo elettronico di stabilità di un'automobile, il sistema di telecomunicazioni di un satellite. A causa della loro criticità, è quindi necessario poter garantire un certo livello di affidabilità, sicurezza o disponibilità per questo tipo di servizi. Fondamentale per un funzionamento corretto ed efficace di tali sistemi è la 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. Di seguito si tratterà la dependability, di cui i concetti fondamentali nella descrizione di sistemi critici quali: guasto, errore, fallimento, attributi quali affidabilità, sicurezza ed altre metriche e mezzi e tecniche per l'ottenimento della dependability.

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 servizio, utente e funzione del sistema.

Definizione 1 (Servizio, Utente, Funzione).

Il servizio fornito da un sistema è il comportamento del sistema stesso, così come viene percepito dai suoi utenti.

Un utente di un sistema è un altro sistema che interagisce attraverso l'interfaccia del servizio.

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.♣

Possiamo ora fornire una definizione di dependability.

Definizione 2 (Dependability). Nella sua definizione originale, la dependability è la capacità di un sistema di fornire un servizio su cui è possibile fare affidamento in modo giustificato. ♣

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].

Questa definizione sottintende un'importante problematica: prevede infatti che una definizione del comportamento del sistema sia disponibile. Non sempre è semplice fornire una specifica precisa, completa e non ambigua di un sistema: infatti il comportamento corretto del sistema deve essere spesso ricavato a partire dai requisiti degli utenti, che sono solitamente impliciti e ambigui. Inoltre, per fornire una specifica completa del sistema, è necessario determinare anche quali sono le condizioni ambientali (esterne) richieste affinché il sistema fornisca il servizio specificato. In altre parole, la dependability puo essere vista come una misura di quanta fiducia possiamo riporre in modo giustificato sul servizio erogato dal sistema stesso. Un'esposizione sistematica dei concetti relativi alla dependability consiste di tre parti: i suoi attributi (attributes), le minacce (threats) e i mezzi (means) per ottenerla (Figura 1.1).

- **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

Definizione 3 (Guasto, Errore, Fallimento). Si definisce **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. Un **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. Un **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 (vedi Figura 1.2). ♣

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 [1]. **(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 siste- ma; 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). I modi di fallimento caratterizzano un servizio non corretto da quattro punti di vista:

- Il dominio dei fallimenti,
- La possibilità di rilevare i fallimenti,
- La consistenza dei fallimenti,
- Le conseguenze dei fallimenti.

Una completa analisi dei rischi (risk analysis, [2]) e delle modalità di fallimento ad essi associate è necessaria per lo sviluppo di sistemi critici; tali attività sono generalmente classificate come obbligatorie negli stessi standard per la certificazione del rispetto di requisiti di dependability (ad esempio in [2]). 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) mostrata in Figura 1.3. La propagazione all'interno di un componente (propagazione interna) è causata dal processo di elaborazione: un errore viene successivamente trasformato in altri errori. La propagazione da un componente A verso un componente B che riceve un servizio da A (propagazione esterna) (Figura 1.4) avviene quando un errore raggiunge l'interfaccia di servizio del componente A. A questo punto, il servizio che B riceve da A diventa non corretto e il fallimento di A appare a B come un guasto esterno, e si propaga come un errore all'interno di B.

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'al- ternanza fra servizio corretto e non corretto.
- **Confidenzialità (confidentiality):** è l'assenza di diffusione non autoriz- zata 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) 1 è poi l'assenza di conseguenze catastrofiche sugli utenti e sull'ambiente circostante. La safety può essere vista come l'affidabilità del sistema, considerando come corretti anche gli stati in cui il sistema subisce un fallimento benigno (possiamo quindi fondere in un unico stato sia gli stati corretti che i fallimenti benigni del sistema, e valutare in questo nuovo schema l'affidabilità). La sicurezza (security) 1 può quindi essere vista come la contempora- nea esistenza di availability solo per gli utenti autorizzati, confidentiality, e integrity, dove per "improprie" si intende "non autorizzate" [3]. Ciascuno di questi attributi può essere più o meno importante in base

al- l'applicazione: la disponibilità del servizio è sempre richiesta, anche se può variare sia l'importanza relativa che il livello quantitativo richiesto, la affida- bilità, 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 de- scrizione degli obiettivi richiesti per uno o più degli attributi sopra descritti, rispetto alle modalità di fallimento previste per il sistema. Se i modi di falli- mento 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 interpretato 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 si dedica alla definizione di algoritmi per trovare efficaciemente le migliori soluzioni a problemi di diversa tipologia. Quindi il campo 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. L'IA si occupa di sistematizzare e rendere automatiche alcune attività intellettive, e di conseguenza si può potenzialmente applicare a ogni sfera del pensiero umano. Detto ciò l'Intelligenza Artificiale si occupa di progettare **agenti razionali**, che posti in un ambiente, riescano a massimizzare la propria misura di prestazione.

Agente , è qualsiasi cosa possa essere vista come un sistema che percepisce il suo ambiente attraverso dei sensori e agisce su di esso mediante attuatori.

Misura di prestazione , rappresenta il criterio in base al quale valutare il successo del comportamento di un agente

Ambiente ,

2.2.1 Neural Network

2.2.2 Convulutional Neural Network

3 Costruzione del dataset

Per addestrare una intelligenza artificiale supervisionata è necessario predisporre un dataset, da usare sia per il training sia per la fase di validation. Tensorflow, richiede per l'addestramento di una cnn, un dataset composto da un training set e da un validation set. Entrambi a loro volta sono composti da dati di una categoria e l'altra divisi fra di loro. Nel caso del progetto, le due categorie di dati sono, immagini pulite e immagini modificate. Per la costruzione del dataset necessario al training del detector, sono state predisposte due fasi.

3.1 Acquisizione

La prima fase del processo di costruzione, 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:

4.1 Addestramento del rete convuluzionale

5 Esecuzione e risultati

6 Conclusioni e lavori futuri

7 A Manuale utente