Appunti di Sistemi Digitali D

Sas

Filippo Lenzi e Guglielmo Palaferri

14 febbraio 2022

Indice

1 Intro **2**

1.1 22 settembre 2021 2

2 Modulo 1 **4**

3 Modulo 2 **5**

3.1 06 dicembre 2021 5

Capitolo 1

Intro

Versione attuale del doc: trascrizione videolezioni circa, da mettere meglio a posto in seguito

1.1 22 settembre 2021

Scopo del corso: introduzione a sistemi embedded

Diverse famiglie: risorse limitate con meno consumo, o viceversa

Metodologie di sviluppo per questi dispositivi

Più vicino a hardware con linguaggi di basso o alto livello, o anche smartphone con linguaggi di più alto livello

Si userà Python, linguaggio ad oggi più utilizzato (doubt)

In contesto anche applicativo: usato in applicazioni reali

Modulo 1: studio dispositivi (come FPGA) su cui si possono usare linguaggi di basso livello, vengono usati nel corso framework di più alto livello come HLS, che usa variante di C/C++, non Verilog e VHDL che sarebbero molto più vicini all'hardware

Inoltre si definiscono modalità di calcolo / progetto del sistema stesso in modulo 1

Modulo 2: come usare architetture esistenti, e anche board con bridge in Python per manipolare hardware sottostante

Python è interpretato ed è meno performante di C, avendo più overhead

Info esame

esame MASSIMO 2 VOLTE

Sistemi Embedded

Sistemi digitali includono tutti i dispositivi a ogni livello di consumo; i **sistemi embedded** sono compatti e a basso consumo energetico; quindi, la complessità delle funzioni da svolgere è un fattore limitante, dovendo spesso gestire una batteria limitata, e evitare altri problemi come surriscaldamento in base al caso d'uso.

Board: scheda con diversi accessori, con un piccolo dispositivo centrale. Più specializzata, architettura progettata ad hoc per la funzione. Hanno comunque delle librerie per interfacciare con accessori in usecase comuni (fotocamere, usb, ecc.).

Alcuni esempi di dispositivi:

- Robot di pulizia casalinga, con software di navigazione in base a sensori.
- Occhiali per persone ipovedenti, con fotocamera che elabora informazioni per fornire info su semafori, testo e altro in forma audio, che necessita di lunga autonomia e limitazione del calore.
- Droni contengono sistema embedded per ricevere il controllo remoto e tradurlo in movimento delle eliche.
- Sblocco del telefono con fotocamera: gestione batteria, fotocamera deve rimanere in "semi-idle" per poter rilevare volto, senza consumare troppa batteria.

Discorso di bilanciamento tra ottimizzare tempo performance mettendo cose in memoria, consumando però più memoria

Computer vision: elaborazione immagini. La mole di dati da elaborare può essere molto grande, essendo matrici. Spesso basato su machine learning. Esiste hardware dedicato al processamento di immagini.

Elaborazione di stream video: mole di dati dipende da framerate oltre che da risoluzione

Due strategie viste nel corso per elaborare video/immagini:

- Modulo 1: Board con ARM + FPGA, paradigma di programmazione classico di basso livello, anche con astrazione via Python in certi casi. Architettura progettata apposta per il problema. Non general purpose, programmazione più complessa.
- Modulo 2: Architettura hardware già progettata e immutabile (smartphone, GPU), con programmazione e tool di più alto livello, più simili a PC desktop. Controllo sull'hardware limitato da API sul dispositivo, e quindi limite alla complessità dei programmi dettato da esse.

Capitolo 2

Modulo 1

Capitolo 3

Modulo 2

3.1 06 dicembre 2021

Introduzione al Deep Learning

Vedremo una panoramica sul mondo del deep learning, in particolare dal punto di vista dello sviluppatore, senza soffermarsi troppo sui dettagli teorici.

Due framework principali: Tensorflow e PyTorch.

Due strade possibili:

- utilizzare una rete neurale già addestrata, integrandola all'interno del proprio progetto software, oppure
- addestrare autonomamente una rete neurale per poi andarla ad utilizzare

Per deep learning si intende una famiglia di metodi basati su reti neurali che tramite l'apprendimento da insiemi di dati etichettati possono consentire di risolvere problemi generalizzati su un qualsiasi insieme di dati.

Teorizzati inizialmente negli anni '80, hanno trovato ampia diffusione solo in tempi recenti, quando è stato possibile soddisfare due requisiti fondamentali per l'utilizzo di reti neurali:

- Elevata potenza di calcolo
- Grandi quantità di dati per l'addestramento

Le reti neurali convoluzionali (CNNs) sono tra i framework più popolari, consentono di risolvere problemi di classificazione e regressione nell'ambito dell'elaborazione delle immagini. Due fasi: Training e Testing

Una rete neurale è definita da una sequenza di moduli (livelli), ciascuno caratterizzato da un insieme di pesi (ognuno associato ad un neurone). I pesi definiscono il comportamento di ogni modulo e di conseguenza della rete nella sua interezza. Durante la fase di Training, la rete processa il dato fornito in input (nel nostro caso un'immagine) e predice un risultato. Tale risultato viene confrontato con l'etichetta reale assegnata all'immagine. Misurando la differenza tra il valore predetto dalla rete e l'etichetta possiamo

quantificare l'errore ed ottimizzare la rete per minimizzarlo, in particolare aggiustando i valori dei vari pesi della rete.

La sequenza di moduli che definiscono la rete può essere percorsa in avanti (**forward pass**) per ottenere l'output a partire dall'input, oppure all'indietro (**backward pass**) per ottenere l'espressione dell'uscita in funzione dei pesi della rete (poiché l'uscita è funzione composta delle uscite dei singoli moduli).

A livello matematico, la differenza tra il risultato della rete e l'etichetta viene modellata tramite una funzione L (*loss function*), calcolata come la differenza tra l'output (funzione degli N pesi della rete) e l'etichetta. La differenza viene minimizzata muovendo i pesi in direzione opposta rispetto al gradiente della funzione L.

espressione matematica del peso θ_j

Framework per il deep learning quali Tensorflow e PyTorch gestiscono automaticamente l'aggiornamento dei pesi nella fase di training della rete.

esempi di problemi risolvibili con una CNN: object detection, instance segmentation, depth estimation ecc.

Vedremo ora il funzionamento dei due framework citati, mostrando infine come sia possibile eseguire il porting su dispositivi mobili (in particolare Android).

Tensorflow

Sviluppato da Google, esistono due versioni principali: 1.x e 2.x. In Tensorflow 1.x, lo sviluppo del codice era diviso in due fasi: costruzione del grafo (trainig) ed esecuzione (testing). In Tensorflow 2.x al contrario, non è prevista la separazione tra le due fasi.

Tensorflow si basa sul concetto di **tensore**, ovvero un array multidimensionale: array (1-D), matrici (2-D), array di matrici (3-D) ecc. Possiamo pensare ad un'immagine come ad un tensore 3-D (altezza, larghezza e canale RGB).

La struttura della rete viene rappresentata tramite un **grafo** unidirezionale, il quale rappresenta l'esatta successione dei layer. Questo consente in modo efficiente sia di percorrere il grafo per ottenere l'output in fase di forward pass, sia di recuperare le dipendenze tra i diversi layer in fase di backward pass. In questo modo si modella la rete tramite un'astrazione ad alto livello.