**1. Introduzione**

In un’epoca dove i grandi colossi dell’informatica hanno ormai monopolizzato i dati, raccogliendone enormi quantità dagli utenti delle loro piattaforme e accaparrandosi un abnorme vantaggio su tutti i competitors, sviluppare un prodotto competitivo può risultare quasi impossibile per un’organizzazione con un bacino d’utenza non molto ampio e priva delle quantità di dati di cui dispone la concorrenza.

Proprio per risolvere il problema della carenza di dati, nascono i dataset sintetici: collezioni di dati generati da calcolatori in grado di simulare una raccolta in campo reale. Questi dataset rendono possibile per chiunque perfezionare i propri algoritmi in maniera efficace e veloce: non si devono, infatti, aspettare dati provenienti dal mondo reale per iniziare il training dei propri algoritmi.

Ovviamente, anche i dati sintetici hanno i loro difetti: il rischio, detto reality gap, di un algoritmo allenato con soli dati sintetici è quello di non sapersi adattare alle situazioni reali.   
Questo possibile problema è stato per anni un deterrente per l’uso dei synthetic data nelle applicazioni, tuttavia, con il miglioramento nel corso degli anni degli algoritmi di generazione di dati, il gioco ha cominciato a valere la candela grazie all’ormai ottima precisione che i dataset sintetici riescono a raggiungere a fronte di costi e tempi neanche minimamente comparabili a quelli di una raccolta dati sul campo.

Nell’ambito della computer vision, dove si colloca il lavoro di tesi che questa introduzione vuole presentare, un dataset sintetico può definirsi tale se composto da un’immagine, contenente gli oggetti la cui presenza è di interesse dell’algoritmo da allenare, e da un’etichetta che descriva quanto contenuto nell’immagine con l’obiettivo di “insegnarlo” all’algoritmo.

Questo lavoro di tesi si pone l’obiettivo di costruire, a tal proposito, un generatore di dati sintetici.   
Tale generatore verrà poi sfruttato per creare un dataset che andrà ad allenare un algoritmo di riconoscimento, con lo scopo di insegnargli ad individuare le mani che vede nelle immagini che andrà ad analizzare.

Per raggiungere tale scopo, sono stati utilizzati gli strumenti descritti nei prossimi paragrafi.

**1.1 Unity 3D 2019.2.0f1**

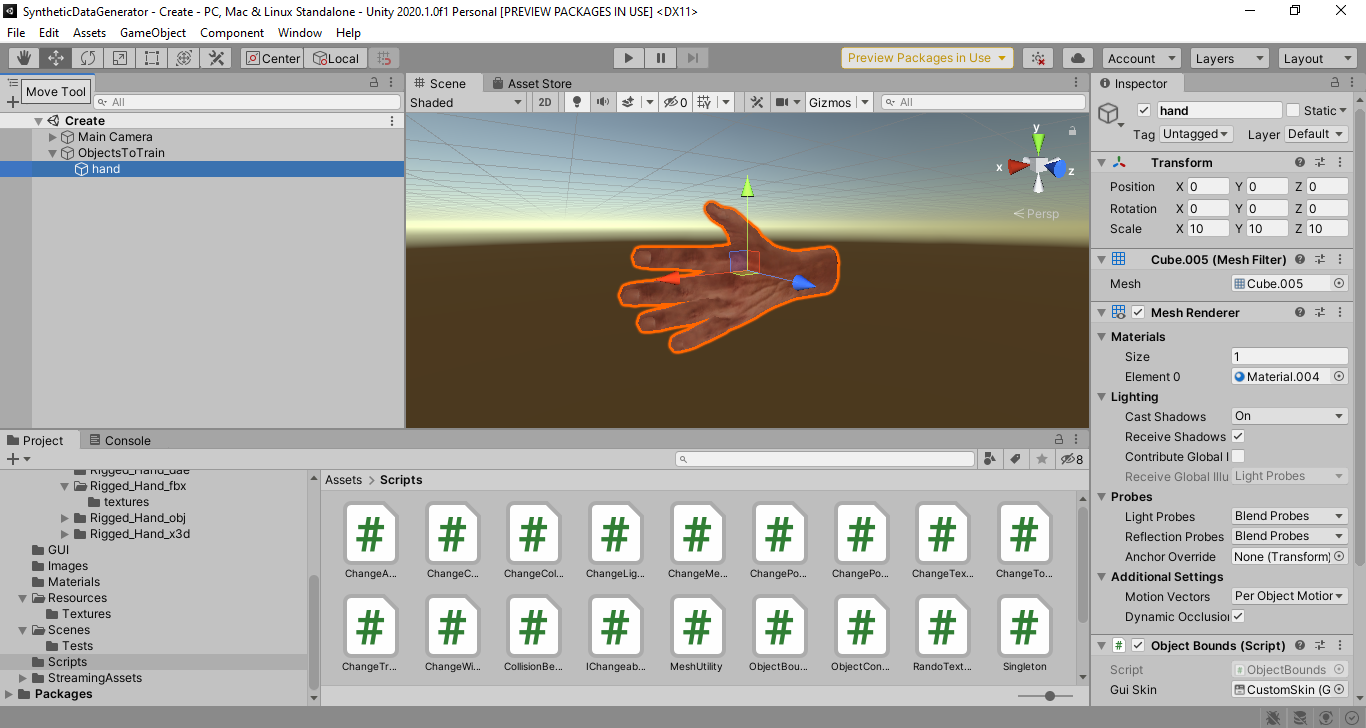
Unity 3D[[1]](#footnote-1) è un motore grafico progettato principalmente per lo sviluppo in C# di videogiochi ambientati in uno spazio tridimensionale.  
  
  
Essendo lo spazio tridimensionale, detto scena, offerto dall’engine popolabile e modificabile a proprio piacimento, Unity risulta molto versatile e le sue applicazioni toccano svariati settori, tra i quali è presente anche il machine learning.  
Gli oggetti 3D presenti nella scena possono essere modificati nelle loro caratteristiche mediante l’azione di scripts: soggetto della modifica possono essere elementi come la posizione dell’oggetto, la sua rotazione o la sua visibilità.

Figura 1.1 – schermata principale di Unity 3D

La versione di Unity utilizzata durante il lavoro di tesi supporta modelli 3D di oggetti in formato non proprietario *.fbx, .dae, .3ds, .dxf* e *.obj*. Sono supportati anche formati proprietari, previa conversione in *.fbx* presa in carico da Unity stesso.

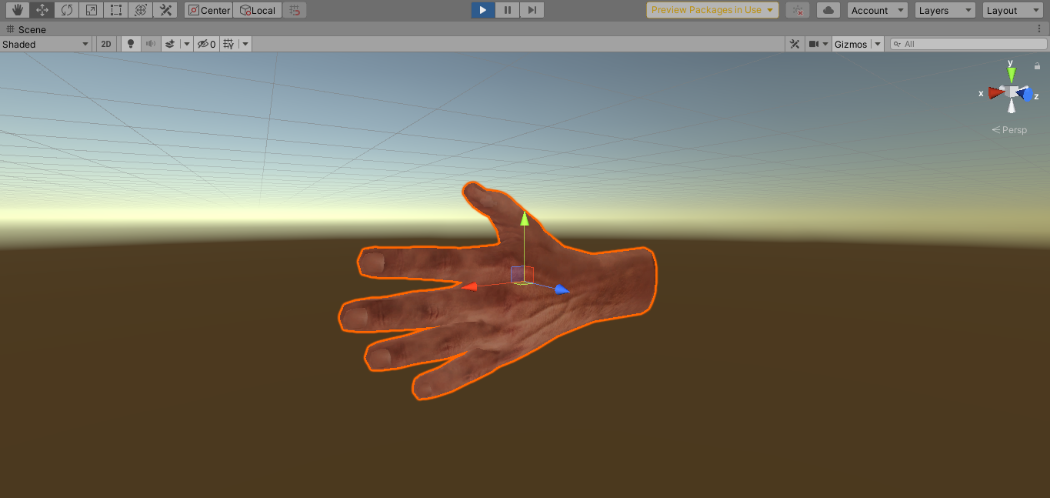


Figura 1.2 – esempio di scena utilizzata durante il lavoro di tesi

Al momento dell’importazione di un file nel suo workspace, Unity compila un documento omonimo in formato *.*meta, contenente informazioni su come l’importazione debba avvenire per quanto riguarda il file in questione.

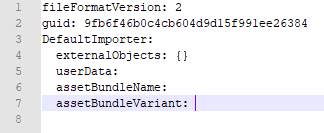


Figura 1.3 – esempio di file .meta

È importante precisare che l’importazione di un file all’interno del workspace Unity non implica il suo utilizzo durante l’esecuzione del software progettato mediante l’engine: per far sì che ciò avvenga, infatti, l’elemento deve essere importato nella scena.  
Va da sé che un file non può essere importato nella scena senza prima essere stato importato nel workspace.

Pur non essendo il miglior motore grafico in circolazione a livello di qualità dei render o di performance, Unity risulta perfetto per la generazione di dataset sintetici per la computer vision: se configurato correttamente, infatti, è in grado di generare numerose immagini adatte al training molto diverse tra loro e di etichettarle in pochi e non molto complessi passaggi.

**1.2 TensorFlow 1.15**

TensorFlow è una libreria open source sviluppata da Google Brain per lo sviluppo di modelli di machine learning: durante il lavoro di tesi è stata utilizzata la sua versione 1.15[[2]](#footnote-2) per Python 3.5.6 per allenare l’API per l’object detection offerta da TensorFlow stesso a individuare la presenza di mani nelle immagini proposte.  
Il progresso delle performance del detector è stato monitorato attraverso l’impiego del tool TensorBoard.

Si è tentato di migrare l’intero progetto a una versione di TensorFlow più recente (2.3), ma il training dell’object detection API non risultava essere compatibile con tale release.



Figura 1.4 – importazione di TensorFlow in uno script Python

**1.2.1 Object detection API**

Immagine che contiene spiaggia, acqua, uomo, inpiedi

Descrizione generata automaticamenteL’object detection API[[3]](#footnote-3) di TensorFlow è un framework che facilita lo sviluppo e il training di modelli per il riconoscimento di determinati oggetti all’interno di immagini.

Figura 1.5 – esempio di object detector fondato sull’object detection API di TensorFlow

**1.2.2 TensorBoard**

TensorBoard, autodefinitosi “TensorFlow’s visualization toolkit”, è un tool che permette di visualizzare, mediante un pannello contenuto in una pagina html all’indirizzo localhost:6006, grafi e grafici relativi al modello di cui si sta effettuando l’allenamento, per poterne osservare e valutare i progressi.

Immagine che contiene screenshot

Descrizione generata automaticamente

Figura 1.6 – una delle schede della dashboard di TensorBoard

**1.3 Fatkun Batch Download Image**

Immagine che contiene screenshot

Descrizione generata automaticamentePer scaricare le immagini dalle quali partire per popolare il dataset, si è scelto di utilizzare Fatkun Batch Download Image[[4]](#footnote-4), estensione di Google Chrome che permette di scaricare in lotto tutte le immagini presenti nelle schede del browser in un dato momento.

Figura 1.7 – schermata principale di Fatkun

1. Scaricabile al link https://store.unity.com/ [↑](#footnote-ref-1)
2. Installabile seguendo le istruzioni al link https://www.tensorflow.org/install [↑](#footnote-ref-2)
3. Installabile, insieme alle relative dependencies, seguendo le istruzioni al link https://github.com/tensorflow/models/blob/d530ac540b0103caa194b4824af353f1b073553b/research/object\_detection/g3doc/installation.md [↑](#footnote-ref-3)
4. Scaricabile al link https://chrome.google.com/webstore/detail/fatkun-batch-download-ima/nnjjahlikiabnchcpehcpkdeckfgnohf?hl=it [↑](#footnote-ref-4)