Relazione di fine stage

Fabrizio D'Intinosante

Università degli Studi di Milano Bicocca — Monday 27th July, 2020

Premessa

Lo stage di tirocinio si è svolto presso il laboratorio **Imaging and Vision Laboratory** con sede in Viale Sarca 336, Edificio U14, a partire dal 17 Febbraio 2020 per una durata complessiva di tre mesi, concludendosi il 17 Maggio 2020.

Il progetto formativo approvato in fase di definizione dello stage prevedeva come obiettivi quelli di imparare ad utilizzare i framework per l'analisi dei segnali digitali monodimensionali, l'analisi delle migliori tecniche nello stato dell'arte per l'identificazione, l'ispezione e la rimozione di rumore in segnali monodimensionali, con particolare enfasi sui segnali audio.

L'obiettivo ultimo si è identificato nel realizzare un progetto di tesi che si pone lo scopo di delineare una strategia per lo sviluppo e l'implementazione di tecniche di machine learning/deep learning per l'identificazione, l'analisi e la rimozione di rumore da segnali audio, prevedendo inoltre la possibilità di specializzare ulteriormente il task (identificazione del tipo di rumore, pulizia selettiva dell'audio etc.).

NB: Nonostante in fase iniziale lo stage si sia svolto in presenza, con orari di ufficio 9:30-17:30, cinque giorni a settimana, a causa dell'emergenza sanitaria è stato per la maggior parte effettuato in *smart-working*, formalmente a partire dal 30 Marzo 2020, ma ufficiosamente fin dalla prima chiusura degli uffici e dei laboratori universitari.

1 Svolgimento

Come anticipato, l'obiettivo primario dello stage era quello di fornirmi una conoscenza di base degli strumenti e delle tecniche per l'analisi dei segnali audio. Per questa ragione fin dal principio si è scelto di optare per l'approfondimento di un *framework* per il *deep learning*, ovvero *PyTorch*[1], e di un toolkit molto specializzato per l'analisi dei segnali monodimensionali (nativamente pensato per assolvere esclusivamente al task di *Speech Recognition*), ovvero KALDI.[2] Per questa ragione, dati gli obiettivi, lo svolgimento dello stage si è basato sostanzialmente sull'analisi e la riproduzione di una *repository* Github[3], nello specifico `https://github.com/jefflai108/pytorch-kaldi-neural-speaker-embeddings`, basata sulla riproduzione del paper *Exploring the Encoding Layer and Loss Function in End-to-End Speaker and Language Recognition System*[4]. Questa repository contiene infatti una *pipeline* basata proprio sull'unione di *PyTorch* e KALDI per la

[1] `https://pytorch.org/`
[2] `https://kaldi-asr.org/`
[3] `https://github.com/`
[4] `https://arxiv.org/pdf/1804.05160.pdf`

Figure 1: Loghi di *PyTorch* e KALDI.

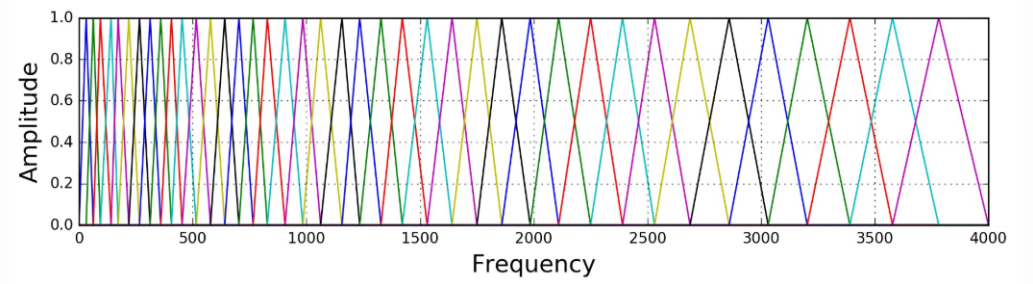

Figure 2: Filter Bank su scala di Mel

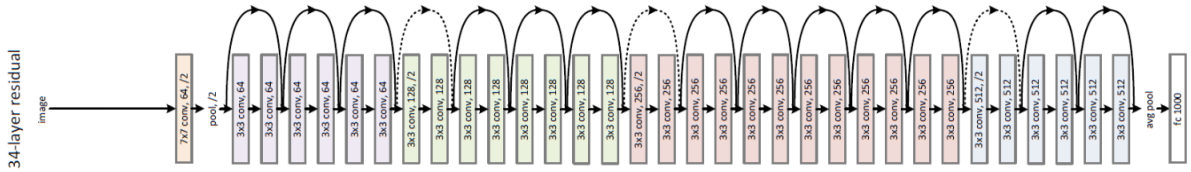

Figure 3: Rappresentazione schematica di una ResNet34

realizzazione di un sistema di *Speaker Recognition end-to-end*. Il sistema ideato si basa sostanzialmente sull'implementazione di una rete neurale addestrata utilizzando gli audio provenienti da due dataset noti, ovvero VoxCeleb1 e VoxCeleb2[5].

La *pipeline* realizzata dagli autori della *repository* consiste brevemente in:

- *Feature extraction*, in particolare *Filter Banks* di cui si ha un esempio in fig. 2 e *Mel-frequency cepstral coefficients* (MFCC) per effettuare *Voice Activity Detection* (VAD), attraverso il *toolkit* KALDI;
- *Data augmentation* attraverso l'inserimento di *noise* (prelevato dal dataset MUSAN[6]) in diversi audio così da arricchire ulteriormente la base dati e permettere alla rete di generalizzare al meglio e *train, test split*;
- Addestramento della rete e valutazione.

La rete neurale utilizzata di default per la riproduzione della *pipeline* è una **ResNet34**, un tipo di rete molto utilizzata per il *task* di *Speaker Recognition* il cui punto di forza risiede proprio nella sua grande profondità, resa possibile dalle *skip connection* che la caratterizzano e che la pongono quindi nella categoria delle reti residuali: uno schema rappresentativo di questo tipo di rete è presente in fig. 3.

2 Considerazioni

Nonostante come anticipato lo stage si sia svolto per la maggior parte del tempo da remoto mi ha comunque permesso di acquisire tutta una serie di conoscenze e competenze utili nel contesto della mia formazione e per quello che è il lavoro di tesi che mi appresto a svolgere.

La *repository* scelta per l'approfondimento è risultata estremamente utile per poter apprendere delle buone basi di *PyTorch* e le *best practices* che lo caratterizzano nella scrittura del codice.

L'ispezione del codice di KALDI, inoltre, mi ha permesso di acquisire per lo meno una buona capacità di comprendere altri linguaggi di programmazione per me fino a poco tempo fa completamente sconosciuti come Perl e C++. L'utilizzo stesso di questo toolkit ha richiesto l'impostazione di Linux come OS, e ciò mi ha quindi permesso di conoscere e utilizzare questo Sistema Operativo, sfruttando oltretutto una tecnologia nativa di Windows, ovvero Windows Subsystem for Linux (WSL), di cui è possibile vedere un esempio della *command line* in fig. 4.

[5] `http://www.robots.ox.ac.uk/~vgg/data/voxceleb/`
[6] `https://www.openslr.org/17/`

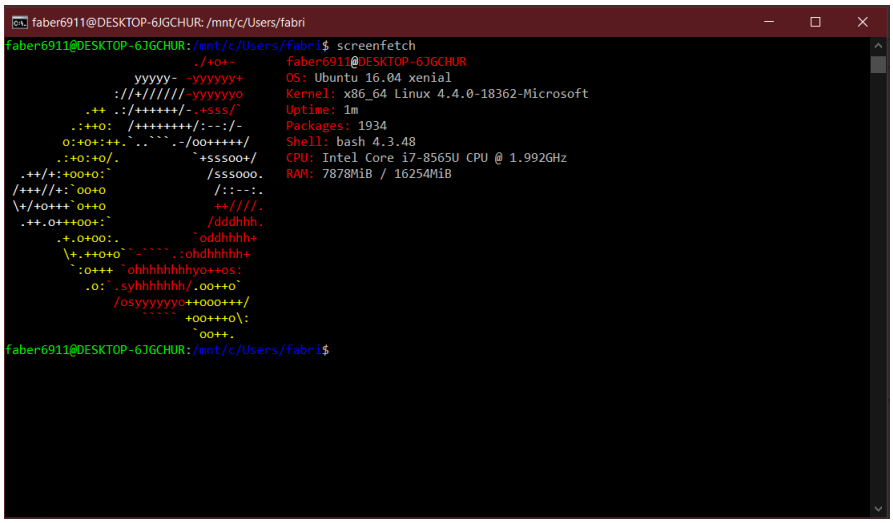

Figure 4: *Screenshot* della *command line* di WSL

L'analisi stessa della *repository*, data la sua grandezza e organizzazione, ha richiesto molto lavoro di *reverse engineering*, permettendomi di acquisire un metodo con cui approcciarmi a questo genere di "fonte di informazioni".

Oltre al lato squisitamente più tecnico e pratico, questo lavoro di approfondimento mi ha anche permesso di acquisire maggiori conoscenze sul lato teorico, anche grazie ad approfondimenti personali per comprendere al meglio le scelte e le opzioni presentate dagli autori della *repository*.

Nel complessso, quindi, reputo questa esperienza di stage molto positivamente poichè mi ha permesso di mettere a frutto quanto imparato fino ad ora nel CdLM in Data Science, ma anche di approfondire alcuni aspetti forse non particolarmente approfonditi all'interno del corso di studi.