Agenzia nazionale per le nuove tecnologie, l'energia e lo sviluppo economico sostenibile

Ministero della Transizione Ecologica

Ricerca di Sistema elettrico

Implementazione e test del modello per la simulazione della mobilità urbana veicolare

Stefano Chiesa, Vincenzo Nanni, Sergio Taraglio

Report RdS/PTR2020/059

IMPLEMENTAZIONE E TEST DEL MODELLO PER LA SIMULAZIONE DELLA MOBILITÀ URBANA VEICOLARE

Stefano Chiesa, Vincenzo Nanni, Sergio Taraglio

Aprile 2021

Report Ricerca di Sistema Elettrico

Accordo di Programma Ministero dello Sviluppo Economico - ENEA
Piano Triennale di Realizzazione 2019-2021 - II annualità
Obiettivo: Tecnologie
Progetto: Tecnologie per la penetrazione efficiente del vettore elettrico negli usi finali
Work package: Local Energy District
Linea di attività: 2.20 - Sviluppo di moduli di calcolo e simulazione della mobilità urbana e di protocolli di comunicazione IoT per la gestione real time di flotte elettriche
Responsabile del Progetto: Claudia Meloni, ENEA
Responsabile del Work package: Maria Pia Valentini, ENEA

Indice

Sommario

Questo Rapporto Tecnico presenta le attività del secondo anno del Piano Triennale 2019-2021 nella Linea di Attività 2.20, Sviluppo di moduli di calcolo e simulazione della mobilità urbana e di protocolli di comunicazione IoT per la gestione real time di flotte elettriche.

La parte principale delle attività è stata dedicata allo studio e alla realizzazione di un modello di simulazione di richiesta di traffico per la Città Metropolitana di Roma. Esso è stato realizzato utilizzando un approccio basato sull'utilizzo di algoritmi di Machine Learning. Si è utilizzato un modello basato sulla realizzazione di un'architettura neurale impiegata come auto encoder variazionale (VAE), un metodo in grado di catturare la struttura statistica dei dati sui quali viene addestrato.

Il VAE è stato dunque addestrato a riprodurre le caratteristiche statistiche dei viaggi contenuti nel *dataset* Octo Telematics relativo al mese di maggio del 2013.

Una volta addestrato, il sistema può essere impiegato per produrre traiettorie di traffico sintetico che ricalchino le caratteristiche statistiche del *dataset* originale. Esso è infatti in grado di produrre un numero qualunque di traiettorie che saranno statisticamente simili a quelle degli utenti facenti parte del dataset di addestramento, ma non saranno uguali. Ciò permette da un lato di salvaguardare la privacy e dall'altro di poter produrre quante traiettorie siano necessarie alle caratteristiche della simulazione da compiere.

I risultati qui presentati mostrano che il VAE realizzato replica con buona accuratezza la distribuzione statistica dei dati di addestramento.

Questo modello sarà utilizzato nelle attività del terzo anno del Piano Triennale per generare la richiesta di traffico alla quale dare una parziale risposta tramite il cosiddetto ride-sharing, con l'obiettivo di ridurre il traffico complessivo.

Ulteriori attività sono state indirizzate all'interfacciamento di quanto qui sviluppato con il software di input e display realizzato in altra parte della Linea di Attività.

1 Introduzione

Nel corso della prima annualità del progetto è stato compiuto uno studio approfondito del *dataset* Octo Telematics a disposizione e ci si è concentrati sulla pianificazione dell'approccio teorico volto alla realizzazione di un modello per la domanda di mobilità nella città di Roma. I dati della Octo Telematics sono relativi alle cosiddette 'scatole nere' montate a bordo di alcuni veicoli da parte di alcune società assicurative a fronte di sconti sulle polizze. Essi sono relativi al mese di maggio 2013, nell'area della Città Metropolitana di Roma e rappresentano le tracce delle traiettorie percorse dai veicoli. Ad ogni passo temporale vengono registrate alcune quantità di interesse del veicolo quali ad esempio la posizione (latitudine e longitudine), lo stato del motore (accensione, spegnimento o in marcia), la velocità, la data e l'ora.

Le attività svolte nel corso del primo anno del PTR 2019-2021 si sono focalizzate principalmente sui dati e la loro analisi, con lo scopo di: eliminare errori e controllarne in generale l'attendibilità; compiere un'analisi dei tempi di sosta concentrandosi principalmente sulle soste piuttosto che sulle traiettorie di movimento; individuare le abitazioni, ovvero le soste notturne di lunga durata; analizzare le soste quotidiane classificandole in diversi tipi di sosta; pianificare l'approccio di modellazione computazionale individuando le reti neurali ricorrenti quali modellatori delle sequenze di attività intraprese nell'arco della giornata dai vari utenti urbani.

Nel rapporto tecnico della precedente annualità si era giudicata una strada idonea quella di utilizzare un'architettura neurale ricorrente che potesse produrre traiettorie di veicoli come sequenze di fermate per attività varie [1], nel corso della presente annualità, ulteriori approfondimenti hanno portato ad una revisione di questa strategia individuando un modello di architettura che possa realizzare la generazione di traiettorie in modo rispondente ai dati, ma slegate da essi. In altre parole si è posto in essere un cosiddetto modello generativo che venga addestrato sui dati a disposizione e che ne riesca a cogliere le caratteristiche statistiche in modo da poter, in seguito, produrre traiettorie completamente 'sintetiche', ovvero slegate dal *dataset* di addestramento, ma che comunque presentino le medesime proprietà generali. Questo modello generativo utilizza le reti ricorrenti come sue costituenti e rappresenta dunque un passo avanti rispetto all'approccio ipotizzato nella precedente annualità.

Nel secondo anno del PTR 2019-2021 le attività si sono dunque focalizzate sulla progettazione, l'implementazione e lo studio approfondito di questa architettura neurale per la realizzazione di un modello di domanda di traffico in grado di cogliere le caratteristiche statistiche del dataset Octo Telematics. Le attività possono essere schematizzate come segue:

- preparazione del *dataset* in termini di fermate e scelta di una collezione di fermate giornaliere limitate;
- studio ed utilizzo di una architettura neurale che realizzi un auto encoder variazionale;
- addestramento ed analisi dell'architettura scelta, dei suoi parametri, dei pesi relativi dei termini di errore;
- studio ed analisi di una possibile architettura neurale alternativa;
- generazione di traiettorie e validazione dell'approccio sia in termini "neurali" che relativi al dominio di applicazione (matrici di flusso veicolare, matrici Origine Destinazione);
- post elaborazione delle traiettorie in output per la presentazione a video;
- interfaccia tra il software del modello generativo ed il software di presentazione all'utente.

2 Descrizione delle attività svolte e risultati

2.1 Preparazione del dataset

La modellazione del traffico inizia con la raccolta dei dati. Le persone nelle loro attività quotidiane generano molti dati durante i loro spostamenti, ad esempio portando ed utilizzando un telefono cellulare. I dati

possono essere raccolti attraverso molti mezzi diversi, ad es. utilizzando i cosiddetti CDR (Call Data Records) ovvero i dati accessori alle chiamate compiute con un cellulare che riportano l'antenna alla quale il cellulare si sia connesso; con l'uso e la pubblicazione di messaggi sui *social network*; attraverso i dati di posizione/navigazione dello *smartphone.* Una ulteriore possibilità è quella di usare i cosiddetti Floating Car Data cioè i dati di posizione ed uso di una vettura dotata di un ricevitore GPS in una cosiddetta scatola nera che alcune assicurazioni fanno montare ai propri clienti in cambio di sconti sulle polizze. Quest'ultima classe di dati è quella a cui si riferisce il presente rapporto tecnico.

Nel corso delle attività del primo anno è stata compiuta un'analisi approfondita dei dati Octo Telematics [2] a disposizione, che contengono i dati GPS memorizzati dalle scatole nere installate a bordo di autovetture private da parte di compagnie assicurative. Essi sono forniti in forma anonima, non è quindi possibile risalire alle generalità del proprietario del mezzo. Nel *dataset* sono presenti 150.633 veicoli che hanno transitato durante il mese di Maggio 2013 all'interno della Città Metropolitana di Roma. Il campione rappresenta dunque circa il 7% di tutto il parco circolante nella zona.

Per ogni veicolo, ad ogni passo temporale, vengono registrate alcune quantità di interesse quali:

- l'identificativo del veicolo,
- la posizione (latitudine, longitudine, qualità del segnale GPS),
- lo stato del motore (accensione, spegnimento o in marcia),
- la velocità e l'orientamento,
- la data e l'ora,
- la distanza percorsa rispetto all'ultima posizione registrata.

La registrazione del dato avviene ogni 2000 metri percorsi quando il veicolo è in marcia oppure ogni 30 secondi, sono inoltre registrate le posizioni in cui avviene un 'cambiamento di stato' ovvero lo spegnimento o l'avviamento del motore anche se asincrone rispetto al passo temporale o spaziale.

A partire da questi dati grezzi sono stati identificati i viaggi effettuati da ogni veicolo: un viaggio è stato definito da una sequenza di dati così organizzati:

1. un'accensione del motore;
2. una sequenza di posizioni con motore acceso;
3. uno spegnimento del motore.

Un viaggio viene considerato come terminato solo se il tempo che intercorre tra lo spegnimento del motore e una sua successiva riaccensione è maggiore di 5 minuti. Le soste di durata inferiore sono ignorate. Il punto in cui il viaggio termina corrisponde a una sosta del veicolo, essa dura fino alla successiva accensione del motore, si veda la Figura 1.

Viaggi per cui non sia presente un punto di inizio o di fine (ad esempio veicoli provenienti o diretti al di fuori dell'area di interesse) non sono stati presi in considerazione.

A valle di questa elaborazione sono stati identificati 11.901.425 viaggi, relativi a 134.074 veicoli sul numero totali di veicoli presenti nel *dataset* pari a 150.633.

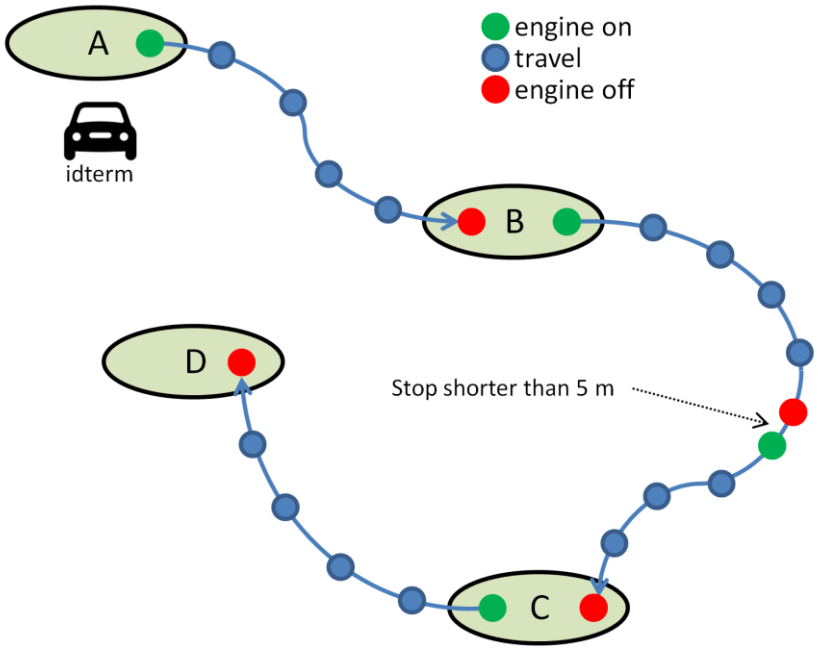

Figura 1. Un esempio di tre viaggi: ogni punto rappresenta un record Octo Telematics

L'approccio allo studio delle traiettorie si è incentrato sull'interpretazione dei viaggi giornalieri come sequenze di attività di utenti. Ovvero un utente, durante la propria giornata, compirà delle azioni in determinati luoghi della città che comporteranno delle soste. Ad esempio si recherà al lavoro dove sosterà per un lungo periodo, tornerà a casa per la notte dove sosterà ancora più a lungo, farà dello shopping o andrà a cena fuori con fermate di durata più corta. Le traiettorie di viaggio, in definitiva, sono una collezione di spostamenti necessari al compimento delle attività umane nei punti di sosta, si veda la Figura 2.

Si è quindi elaborato il *dataset* in modo da realizzare una rappresentazione dei viaggi attraverso le soste. Una sosta è stata descritta dai seguenti sei dati:

- latitudine;
- longitudine;
- durata;
- orario di inizio;
- giorno della settimana (da 0: domenica a 6: sabato);
- lunghezza del viaggio effettuato per raggiungere il luogo in cui si effettua la sosta.

Si è definita come traiettoria una sequenza di soste effettuate da un utente nella stessa giornata.

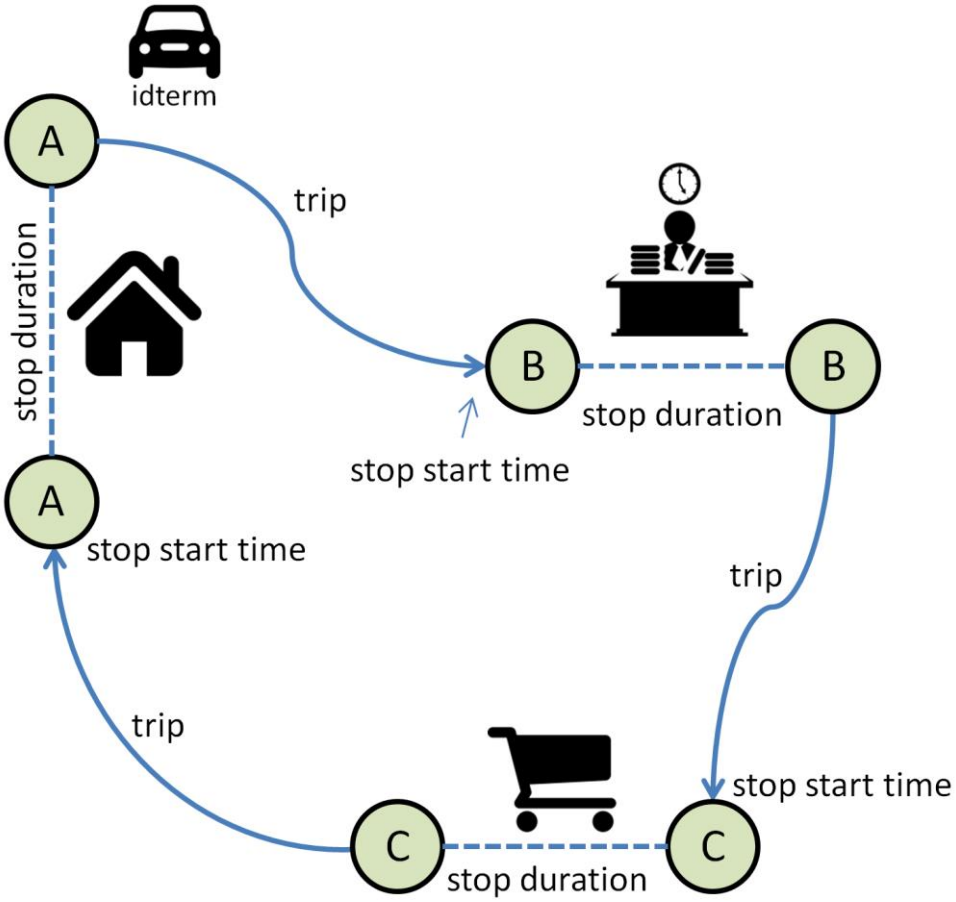

Figura 2. Una serie di traiettorie di viaggio in una giornata. Le attività umane intese come collezione di soste collegate da spostamenti: casa, ufficio, spesa

Facendo riferimento all'analisi svolta nel precedente anno di attività ([1], Figura 2 del Paragrafo 2.2) si è analizzata la distribuzione di numero di soste giornaliere compiute dagli utenti. In Figura 3, per comodità di lettura, è riportata la citata figura.

A valle di questa analisi si è scelto il numero di otto soste come numero massimo di soste nell'arco di una giornata tipo di un utente automobilistico tipico. Con riferimento alla Figura 3, operando una tale scelta si seleziona una percentuale molto elevata di utenti, infatti il valor medio di fermate giornaliere nel *dataset* è di 4.99 soste/giorno e 4.04 soste/giorno rispettivamente nei giorni della settimana lavorativa e in quelli festivi e del weekend.

In prima approssimazione si è deciso di eliminare tutte le traiettorie di tutti gli utenti che abbiano compiuto più di 8 fermate nell'arco della giornata. L'idea di fondo è quella di evitare di prendere in considerazione ad esempio le consegne di merci ed i corrieri, che potrebbero introdurre un *bias* nei dati.

Limitando a 8 il numero massimo di soste in una giornata nel modo citato, si seleziona circa il 10% delle traiettorie del *dataset* ed il 50% dei veicoli.

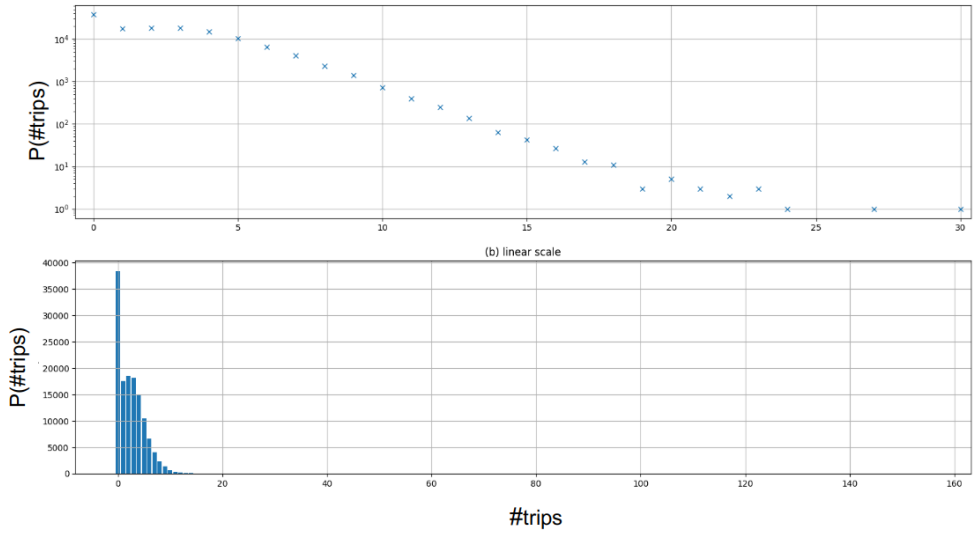

Figura 3. Distribuzione del numero di soste nelle traiettorie del dataset. In alto grafico logaritmico (Figura 2 del Paragrafo 2.2 di [1]

Riassumendo: l'insieme dei dati che saranno analizzati e modellati è rappresentato da tutte quelle traiettorie nel *dataset* originale nelle quali un utente non effettui più di 8 soste nella stessa giornata. Esso è costituito quindi da un insieme di traiettorie, ognuna composta da una sequenza di massimo 8 soste descritte ognuna dai 6 parametri elencati in precedenza. Le dimensioni del *dataset* di addestramento saranno quindi numero di traiettorie x 8 x 6.

Qualora il numero di soste di un dato utente fosse inferiore a 8 i valori relativi alle soste mancanti sono stati posti uguali a zero (il cosiddetto *zero-padding*). Questi dati saranno dati in input ad un'architettura neurale e quindi avranno necessità di una normalizzazione nell'intervallo [0, 1]. Questa è stata compiuta in modo lineare tra il minimo ed il massimo valore per ognuno dei sei campi sull'intero *dataset* con la formula usuale:

$$\hat{x} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

In Figura 4 è mostrato un esempio di sequenza giornaliera di soste dopo la normalizzazione, dove ogni riga rappresenta una delle 8 soste e le colonne sono i sei campi di ciascuna; nel caso presentato sono presenti 3 sole soste e le rimanenti sono azzerate.

I valori di minimo e di massimo di ognuno dei 6 campi vengono salvati per permettere la denormalizzazione dei dati in uscita. In Figura 5 è mostrata la traiettoria di Figura 4 denormalizzata. Per poter considerare correttamente il primo viaggio in un dato giorno la prima fermata di una qualunque traiettoria è rappresentata dall'ultima fermata del medesimo utente compiuta il giorno precedente o nei giorni precedenti. In altre parole la traiettoria giornaliera di un dato utente parte dal punto dove ha lasciato parcheggiata l'auto e quindi nella locazione dell'ultima sosta che, nella maggioranza dei casi, sarà la sera precedente, ma può essere più indietro nel tempo, ad esempio se nel fine settimana l'auto non venga usata. Ciò è evidente nella colonna del giorno della settimana (DOW, Day Of Week) di Figura 5, dove la traiettoria è composta da due soste nel giorno 4 ma il suo inizio è nel giorno 3.

	latitude	longitude	duration	hour	DOW	tripdistance
0	0.53025	0.53616	0.00017	0.84920	0.50000	0.00816
1	0.59362	0.54967	0.01277	0.32801	0.66667	0.00743
2	0.52966	0.53661	0.01939	0.72689	0.66667	0.00480
3	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
4	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
5	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
6	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
7	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

Figura 4. Esempio di input: una sequenza di tre soste normalizzate: le colonne sono rispettivamente latitudine, longitudine, durata della sosta, ora di inizio della sosta, giorno della settimana, distanza del viaggio

	latitude	longitude	duration	hour	DOW	tripdistance
0	41879391.01414	12571554.97133	744.00001	20.36667	3.00000	16104.99992
1	41935725.99238	12592689.98714	32958.00002	7.86667	4.00000	14670.00015
2	41878868.01971	12572257.95546	49888.00067	17.43333	4.00000	9474.00043
3	41408004.00000	11733001.00000	301.00000	0.00000	0.00000	1.00000
4	41408004.00000	11733001.00000	301.00000	0.00000	0.00000	1.00000
5	41408004.00000	11733001.00000	301.00000	0.00000	0.00000	1.00000
6	41408004.00000	11733001.00000	301.00000	0.00000	0.00000	1.00000
7	41408004.00000	11733001.00000	301.00000	0.00000	0.00000	1.00000

Figura 5. La sequenza di Figura 4 denormalizzata

Si noti come i dati che prima erano facilmente identificabili come “zeri” ora hanno dei valori che li rendono più difficili da identificare come tali, questo perché nel processo di normalizzazione lo 0 corrisponde al valore minimo per la variabile. Tuttavia tutti i punti vicino allo “zero” per latitudine e longitudine nel *dataset* sono localizzati geograficamente nel mare Tirreno per cui tutti i punti al di sotto della retta rossa indicata in Figura 6 sono stati considerati punti non validi.

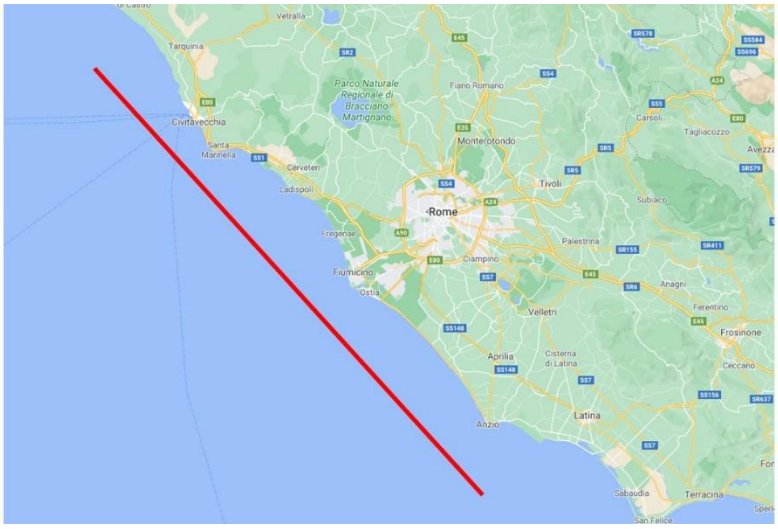

Figura 6. La retta utilizzata per discriminare i punti non validi

2.2 Modelli generativi, auto encoder variazionali e reti LSTM

La mobilità umana può essere vista come un dominio altamente strutturato composto da calendari giornalieri/settimanali per lo più regolari, che mostrano un'elevata prevedibilità tra diverse popolazioni [3]. Ulteriori analisi suggeriscono anche che la mobilità umana mostra regolarità temporali e spaziali [4] [5].
Tutti questi dati sulla mobilità hanno una problematica comune: la questione dell'anonimizzazione, le informazioni di viaggio possono contenere informazioni che non permettono di proteggere il diritto delle persone alla privacy. Oltre a questa questione di base, vi sono ulteriori problemi legati sia all'effettiva disponibilità dei dati sia alla loro rappresentatività, infatti sono spesso dati parziali o poco rappresentativi, poiché sovente essi appartengono a imprese private o ad enti governativi, come ad esempio il *dataset* qui utilizzato.

2.2.1 Modelli generativi

Un possibile modo per superare queste problematiche è l'utilizzo dei cosiddetti modelli generativi, come è stato recentemente proposto in diversi campi di applicazione [6], ma anche nel campo della modellazione della domanda di traffico [7] [8].

Un sistema che modelli la distribuzione del traffico può essere inteso come un modello generativo, ovvero un modello che conosca le regole con le quali generare nuove traiettorie di viaggio con caratteristiche statistiche simili ai dati sperimentali di addestramento.

In estrema sintesi: dato un insieme di osservazioni, si assuma che queste osservazioni siano state generate con una qualche distribuzione statistica sconosciuta P_{data}. Un modello generativo P_{model} è disegnato per approssimare al meglio P_{data}, a questo punto è possibile campionare P_{model} per generare osservazioni che abbiano la stessa distribuzione di P_{data}.

Il modello generativo P_{model} sarà reputato soddisfacente se genererà: a) esempi che sembrano campionati da P_{data} e b) esempi che siano diversi da quelli presenti nel set di osservazioni usato per l'addestramento, ovvero non deve semplicemente riprodurre dati noti, contenuti nell'insieme delle osservazioni originali.

Un modello generativo ha due conseguenze molto positive per lo studio degli spostamenti delle persone in una città:

1. rende anonima la traiettoria, risolvendo il problema della privacy del singolo utente; infatti la traiettoria generata sarà completamente sintetica anche se con le caratteristiche statistiche di quelle 'vere';
2. è possibile creare insiemi di dati molto più abbondanti di quelli usati per la realizzazione del modello generativo.

2.2.2 Auto encoder ed auto encoder variazionali

Ci sono molteplici modi di realizzare un modello generativo [9], per realizzare quello del traffico è stata qui scelta un'architettura neurale nota come auto encoder: un sistema composto dalla giustapposizione di una parte di codifica ed una di decodifica. La prima comprime l'input in uno spazio di variabili latenti e, successivamente, la seconda ricostruisce l'output sulla base di tali informazioni latenti. Questa tipologia di rete è dunque composta da due parti:

- *encoder*: la parte della rete che comprime l'input in uno spazio di variabili latenti e che può essere rappresentato dalla funzione di codifica $h = f(x)$.
- *decoder*: la parte che ricostruisce l'output sulla base delle informazioni precedentemente raccolte. È rappresentato dalla funzione di decodifica $r = g(h)$.

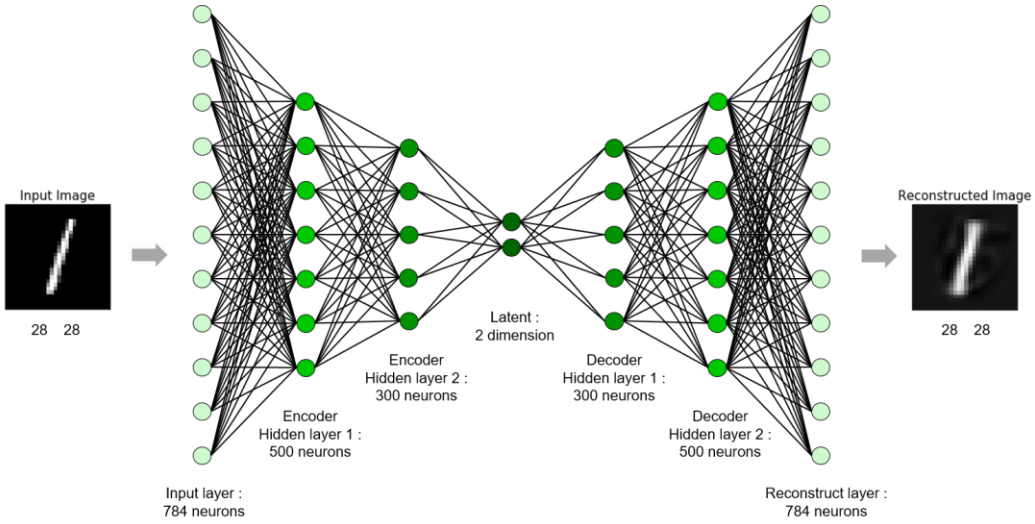

Figura 7. Un esempio di auto encoder per l'elaborazione di numeri scritti a mano

In definitiva la rete calcola la funzione *r = g(f(x))*, il termine 'auto' dell'auto encoder fa riferimento al concetto che la rete viene addestrata a far in modo che l'output *r* sia il più possibile simile all'input *x*. Lo scopo è quello di fare in modo che lo spazio delle variabili latenti *h* possa assumere delle caratteristiche utili, rappresentando l'insieme dei dati con una qualche codifica interna che si possa rivelare vantaggiosa. Ad esempio se lo spazio latente *h* ha dimensioni minori di *x*, quindi anche dell'output *r*, l'auto encoder andrà ad estrarre le caratteristiche più rilevanti dei dati. Caratteristiche che gli permetteranno di riprodurre ugualmente in output l'input, pur basandosi su di un set di dati più ridotto, si veda la Figura 7.

Ad oggi la riduzione del rumore e quella della dimensionalità per la visualizzazione dei dati sono considerati le applicazioni più interessanti degli auto encoder. Ad esempio è possibile condurre delle proiezioni in sottospazi in modo equivalente alla PCA (Principal Component Analysis, analisi a componenti principali).

Per chiarire meglio il concetto si riporta un esempio molto noto in letteratura, l'applicazione di un auto encoder al *dataset* MNIST dei numeri scritti a mano, utilizzato per la lettura automatica dei codici postali negli USA. Le immagini sono 28x28 pixel con 256 toni di grigio, Figura 8 a sinistra. Utilizzando una rete composta da più strati si può facilmente realizzare un sistema che ha come encoder una rete che possieda 28 x 28 neuroni di input e che, strato dopo strato, li riduca ad uno spazio latente *h* con dimensionalità 2; una rete speculare, il decoder, avrà due neuroni di input e li riporterà a 28 x 28 neuroni di output (Figura 7).

Durante l'addestramento si presentano in input alla rete via via tutte le immagini del *dataset* e si impone che in uscita dalla rete si ottengano le medesime immagini di ingresso. Naturalmente all'inizio il sistema 'allucinerà' immagini molto diverse da quanto richiesto, quindi con un errore molto grande, ma proprio sulla base di questo errore, tramite un algoritmo noto come *backpropagation*, è possibile variare i parametri interni della rete per ridurre al minimo l'errore di ricostruzione.

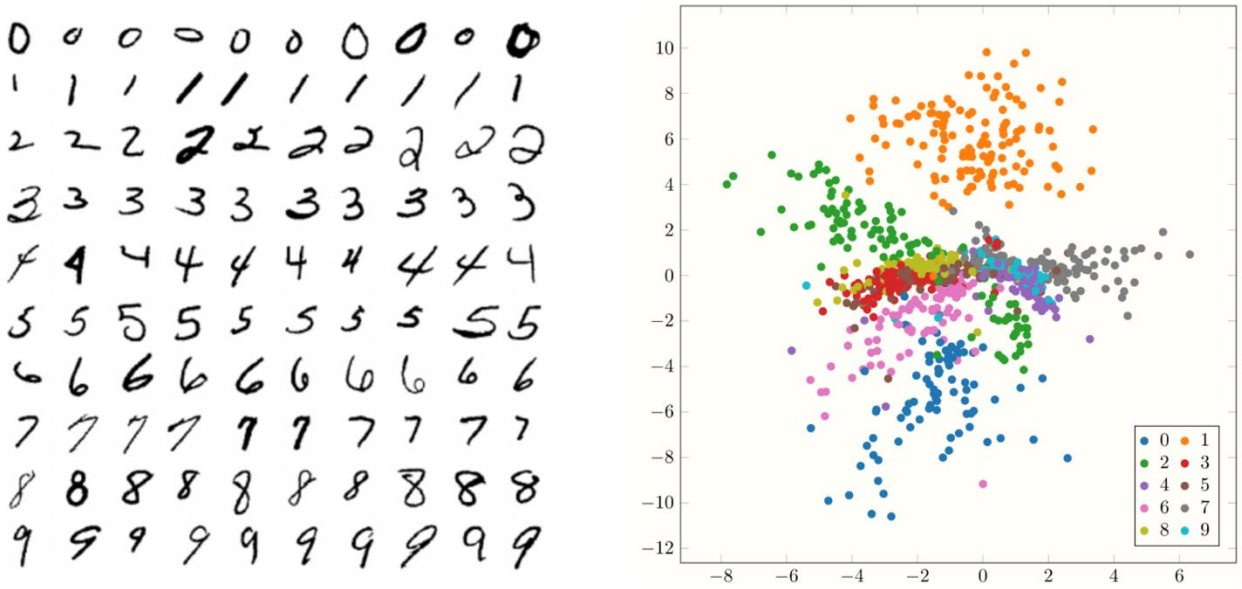

Figura 8. Sinistra: un sottoinsieme del dataset MNIST dei numeri scritti a mano. Destra: rappresentazione grafica dello spazio latente di un auto encoder addestrato con il dataset MNIST

L'auto encoder una volta addestrato sulle immagini è dunque in grado di ricostruirle in output con un errore minimo, pur avendo nello strato più interno (lo strato latente *h*) due soli neuroni. I valori di questi due neuroni possono essere facilmente graficati su di un piano: in Figura 8 a destra sono mostrati quelli relativi ai primi 1000 numeri imparati dalla rete, è evidente come cifre simili siano raccolte in zone vicine, si nota altresì una certa sparsità dei punti. Questi punti rappresentano l'*encoding* realizzato dalla rete con la sua prima metà, usando invece la seconda parte (il *decoder*) è possibile passare dallo spazio latente alla ricostruzione di una cifra.

Quindi dall'auto encoder addestrato si può pensare di estrarre la sola parte di *decoding* (quella che calcola la funzione *r = g(.)*) ed utilizzarla come modello generativo a partire dai valori dello spazio latente.

Va osservato che se si sceglie un punto a caso nello spazio latente e lo si usa come ingresso al decoder non è detto che la cifra in output sia leggibile, inoltre la distribuzione spaziale delle cifre non è uniforme: prendendo punti a caso uniformemente nello spazio latente si produrranno statisticamente meno '9' che '1' essendo le due aree molto diverse.

Per ovviare a questi problemi è stato introdotto il VAE ovvero il Variational Auto Encoder (auto encoder variazionale) [6][10]. L'idea di fondo è sostituire alla mappatura di una cifra in un solo punto dello spazio latente, quella in una distribuzione uniforme multivariata, nel caso qui usato come esempio bivariata essendo due i neuroni nello spazio latente, si veda la Figura 9.

Nel caso variazionale l'encoder, al termine dell'addestramento, codificherà ogni immagine in due vettori che rappresentano il valor medio e la varianza di una distribuzione normale multivariata nello spazio latente. La conseguenza è che sarà ora possibile campionare lo spazio latente distribuito normalmente ed ottenere a valle del decoder le cifre in output con la loro effettiva distribuzione. In altre parole un VAE è in grado di prendere in input dei dati con una data distribuzione statistica e fa in modo di riprodurli in uscita, ma passando attraverso una distribuzione normale nel proprio spazio latente. Ciò serve a poter generare, attraverso un campionamento normale dello spazio latente, dati in output che rispettino la distribuzione dei dati di addestramento, superando i limiti anzidetti dell'auto encoder standard.

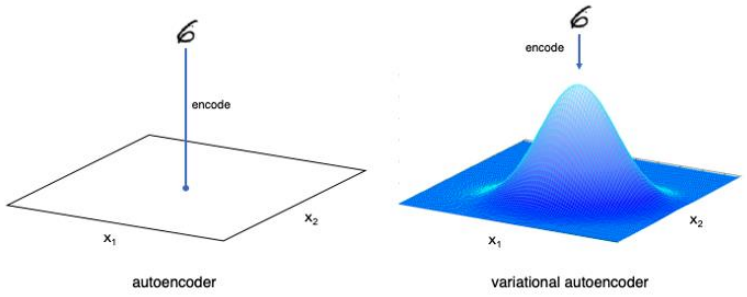

Figura 9. La differenza di fondo tra l'auto encoder standard e quello variazionale

2.2.3 Reti ricorrenti LSTM

Le due parti di *encoding* e *decoding* in un VAE possono essere scelte in molti modi, purché rispettino la specularità tra *encoder* e *decoder*. Nel presente studio è stata utilizzata una rete neurale ricorrente: la LSTM (Long Short Term Memory, memoria a lungo e breve termine) [11], una istanza delle reti neurali ricorrenti.

Le reti neurali ricorrenti (RNN, Recurrent Neural Network) sono una classe di modelli dinamici che, in tempi recenti, sono stati utilizzati per generare sequenze in domini molto diversi tra loro come ad esempio la musica, il testo e dati di filmati. Le RNN hanno ottime prestazioni nella elaborazione di sequenze un passo alla volta, prevedendo cosa accadrà nel successivo. E' possibile generare nuove sequenze da una rete addestrata campionando iterativamente l'output della rete, inserendolo come input per la fase successiva. Queste caratteristiche bene si attagliano al problema in esame, quello di generare sequenze di soste giornaliere.

Le RNN sono in grado di connettere informazioni relative a tempi precedenti con il compito elaborativo presente. Ad esempio le immagini precedenti di un video possono chiarire il contesto dell'immagine presente oppure la comprensione di un testo si basa sulle parole dell'intera frase e non solamente sull'ultima. Nell'architettura delle RNN ciò si può ottenere attraverso la fornitura in input anche dell'output del passo temporale precedente, questo consente alla rete di basare le sue elaborazioni sulla storia passata (un effetto memoria) ovvero su tutti gli elementi di una sequenza e sulla loro posizione reciproca.

Tra le RNN la LSTM (Long Short Term Memory, memoria a lungo e breve termine) è un'architettura progettata per essere migliore nell'archiviazione e nell'accesso alle informazioni rispetto alle RNN standard che spesso presentano il problema di 'dimenticare' dati troppo lontani nel corso della sequenza in elaborazione (problema detto del *vanishing gradient*). La LSTM ha recentemente fornito risultati all'avanguardia in una varietà di attività di elaborazione di sequenze, tra cui il riconoscimento vocale e dei testi [12], immagini [13] e grafia [12].

2.3 Addestramento della rete: studio dell'architettura

Il sistema complessivo qui utilizzato è un VAE composto da due reti LSTM che operano come *encoder* e *decoder*. Questo è stato addestrato a riprodurre le traiettorie giornaliere dei veicoli presenti nel database, prendendo in considerazione le sequenze di soste non più lunghe di 8, espresse in termini di posizione (latitudine e longitudine), durata, giorno della settimana, ora di inizio e lunghezza del viaggio precedente. In termini numerici si tratta di più di 900.000 traiettorie diverse.

Le caratteristiche architetturali di un tale sistema che possono essere regolate sono diverse. Tra le più importanti si possono citare:

- il peso dato alle varie componenti della funzione di errore;
- la dimensione dello spazio latente;
- le dimensioni dello strato di encoding (o di decoding).

È stata eseguita un'analisi dell'influenza di queste caratteristiche dell'architettura sulle prestazioni del sistema, che viene qui brevemente presentata.

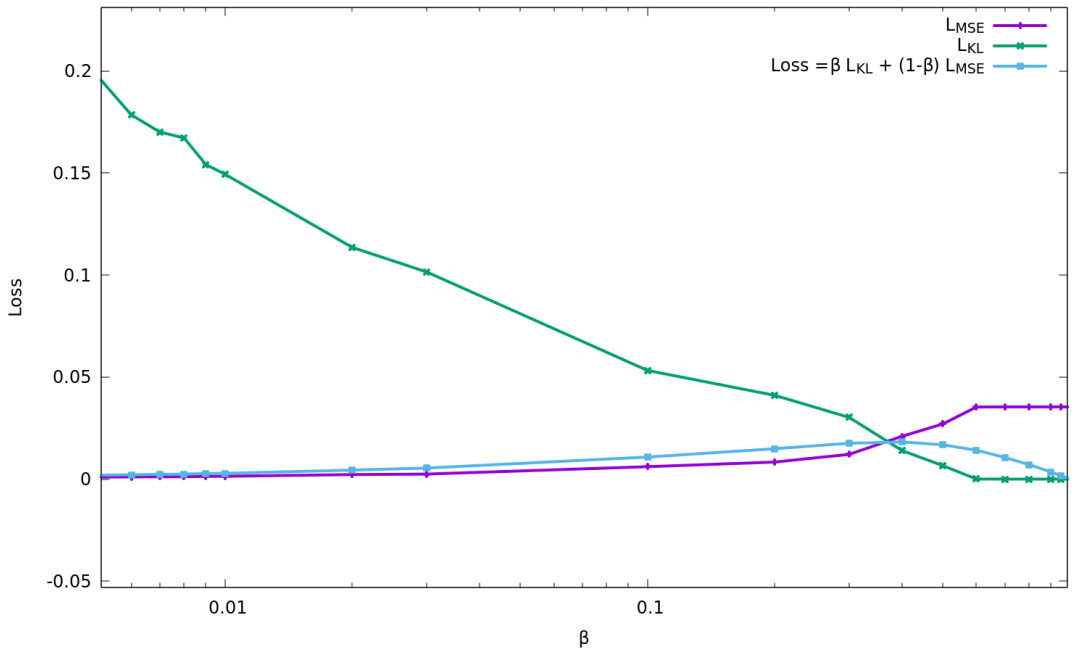

Figura 10. L'errore finale in funzione di β il parametro di compromesso tra errore di ricostruzione (MSE) e termine di regolarizzazione (KL), un valore per β di 0,02÷0,03 assicura un MSE basso con una buona regolarizzazione

La funzione di errore, detta *loss function*, di un VAE è solitamente composta da due termini: un termine di errore di ricostruzione che descrive quanto bene l'output corrisponda all'input e un termine di regolarizzazione che aiuta a fornire una distribuzione normale multivariata nello spazio latente. In questo lavoro i due termini sono bilanciati attraverso la:

$$L_{TOT} = \beta L_{KL} + (1 - \beta) L_{MSE}$$

dove L_{MSE} è l'errore di ricostruzione, tipicamente misurato dall'errore quadratico medio (MSE, mean square error), e L_{KL} è l'errore di regolarizzazione, tipicamente valutato con la divergenza di Kullback-Leibler. Nella Figura 10 è mostrato il comportamento dell'errore finale di addestramento in funzione del termine β. È evidente che è sufficiente un bassissimo contributo del termine di regolarizzazione per ottenere buoni risultati, ovvero β=0.02÷0.03.

Dalla Figura 10 si può vedere come l'errore L_{MSE} cresca all'aumentare di β; è necessario tenere presente che L_{MSE} deve essere tenuto il più basso possibile compatibilmente con una regolarizzazione dello spazio latente. Infatti esso governa la corretta ricostruzione delle traiettorie, aspetto qui di maggior interesse.

Sono stati investigati anche aspetti legati all'architettura del sistema, ad esempio diverse dimensioni dello spazio latente ed in Figura 11 è riassunto sia l'errore di addestramento in funzione delle epoche che quello finale con β fissato a 0,02. È evidente che maggiore è la dimensione dello spazio latente, minore è l'errore. Di seguito, per ulteriori analisi, la dimensione dello spazio latente è stata posta uguale a 20.

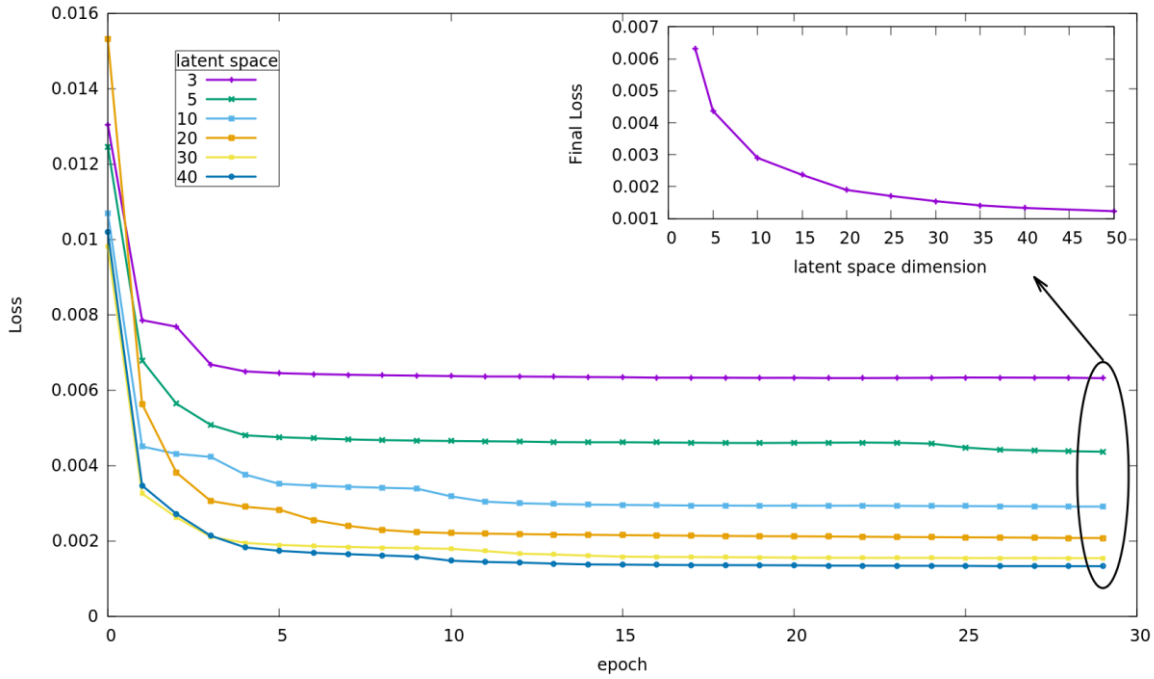

Figura 11. Addestramento con diverse dimensioni dello spazio latente. Grafico grande: l'errore totale in funzione delle epoche di addestramento per diverse dimensioni dello spazio latente. Riquadro: l'errore finale in funzione della dimensione dello spazio latente (con β = 0,02)

Nella Figura 12 è invece mostrato il comportamento della funzione di errore rispetto alla dimensione della LSTM, cioè il numero di neuroni LSTM nelle due parti del codificatore e del decodificatore nell'auto encoder. L'errore finale diventa rapidamente insensibile al numero di neuroni LSTM nell'auto encoder, ed è stato scelto un numero di 32 neuroni per le reti LSTM all'interno dell'auto encoder.

Questa attività di analisi ha portato a scegliere quali parametri per l'architettura un β = 0,02, uno spazio latente di dimensionalità 20 ed un numero di neuroni LSTM pari a 32.

In Figura 13 è invece illustrato il comportamento dell'errore finale in funzione del numero di diversi automobilisti considerati, ovvero 0.1K, 1K, 3K, 10K, 30K; è interessante notare che l'errore rimane approssimativamente costante mentre il set di dati aumenta di numero, sia nei termini di ricostruzione che di regolarizzazione, il grafico è relativo ad elaborazioni con 100 epoche di allenamento, una dimensionalità latente di 20, 32 neuroni LSTM e β = 0,02.

L'errore di ricostruzione complessivo è dell'ordine del 10^{-3}.

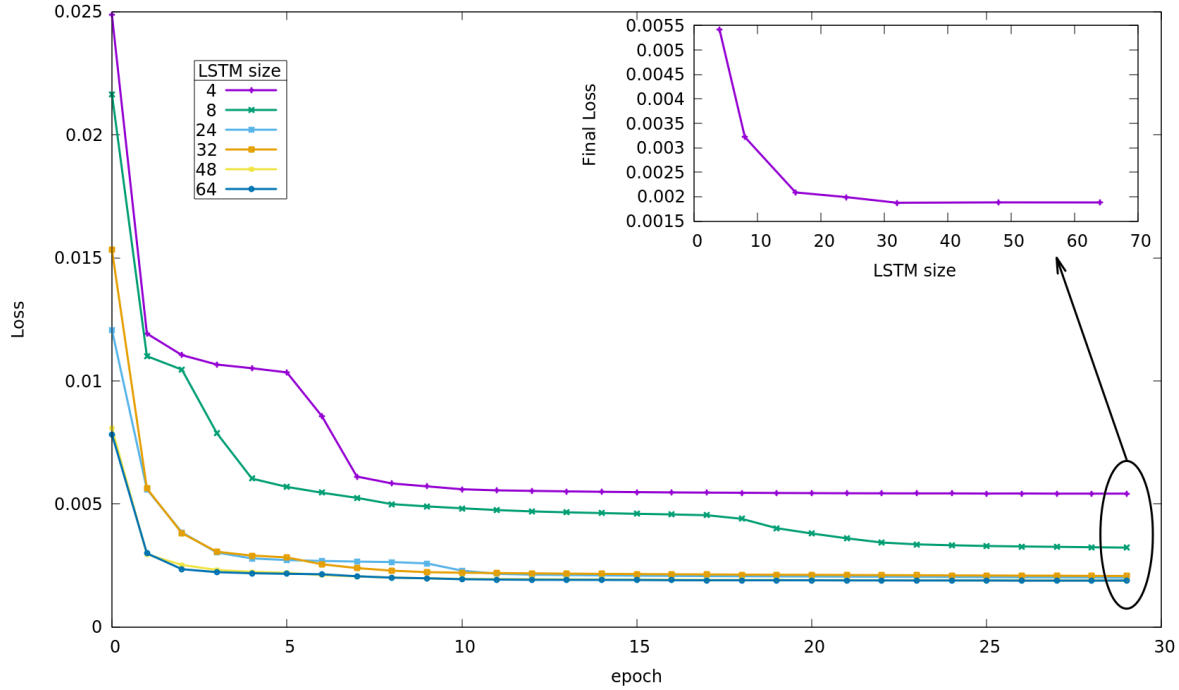

Figura 12. Addestramento con diverse dimensioni LSTM. Grafico grande: l'errore totale in funzione delle epoche di allenamento per diverse dimensioni LSTM. Riquadro: l'errore finale in funzione della dimensione LSTM (con β = 0,02)

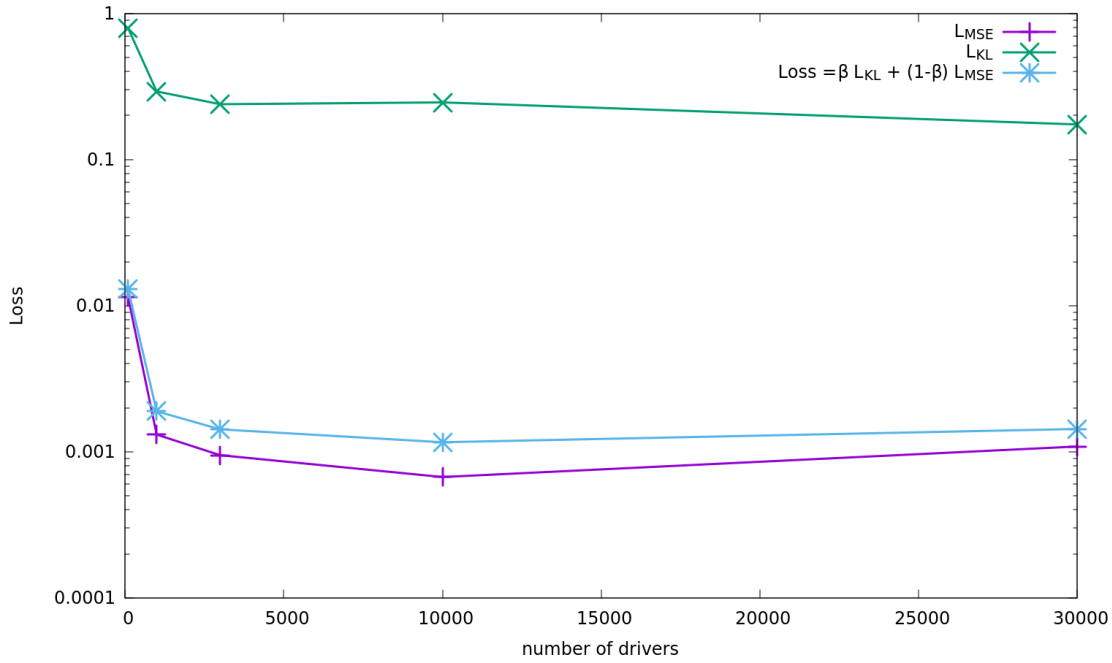

Figura 13. Comportamento dell'errore finale in funzione del numero di driver

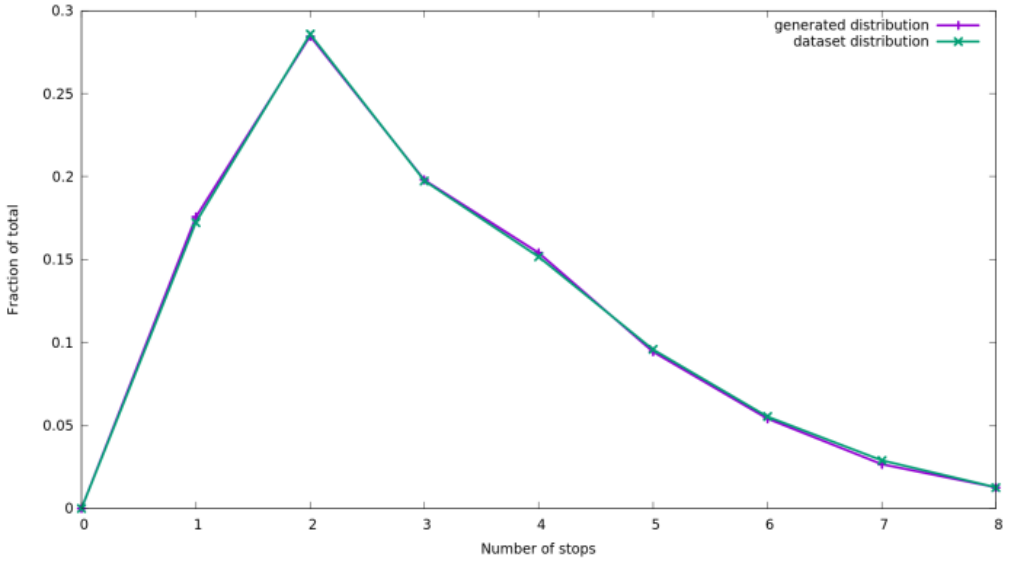

Figura 14. La distribuzione normalizzata del numero di fermate nelle traiettorie: confronto tra quella del dataset di addestramento e quella delle traiettorie generate

Si è poi passato ad una verifica delle capacità della rete nella ricostruzione del *dataset* di addestramento. Nella Figura 14 viene presentata la comparazione tra la distribuzione normalizzata del numero di soste giornaliere nel set di dati di allenamento ed in quello delle traiettorie come ricostruite dalla rete. Il confronto mostra che la distribuzione statistica di questa quantità è stata colta accuratamente dal modello.

Nella Figura 15 è presentato l'istogramma delle fermate usando come variabili l'ora di inizio della fermata stessa e la sua durata.

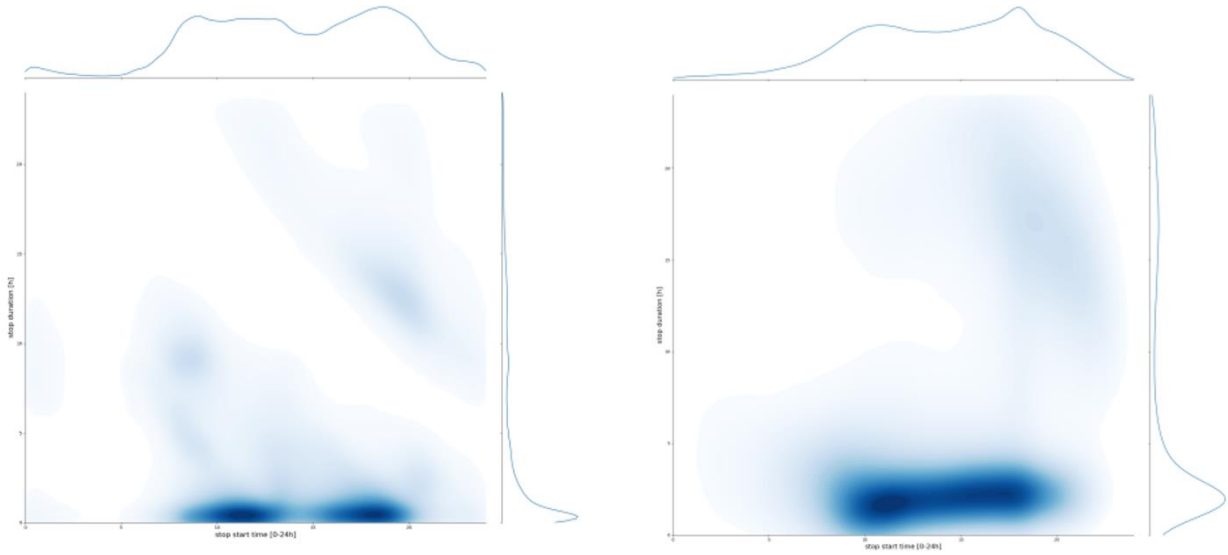

Figura 15. Istogramma delle fermate in funzione dell'ora di inizio della fermata (asse x) e della durata (asse y): a sinistra il set di dati di addestramento, a destra quello delle traiettorie generate

Nella parte sinistra della figura è mostrato l'istogramma come calcolato a partire dal *dataset* di addestramento originale, mentre a destra è graficato quello relativo alle traiettorie ricostruite. Superiormente e sul fianco destro di entrambi i grafici sono mostrate le distribuzioni monovariate.

E' possibile notare che: le soste a breve termine sono nelle stesse fasce orarie e con durate leggermente più lunghe; le soste notturne nelle traiettorie generate sono un po' più lunghe nella durata e più tardive nel loro

inizio, tuttavia è evidente la somiglianza tra i due grafici e la qualità complessiva delle traiettorie generate è simile al comportamento del set di dati originale.

A titolo di esempio in Figura 16 sono comparati i dati di longitudine e di start time tra il *dataset* di addestramento (in blu) e quello come ricostruito dalla rete (in arancione). E' chiaramente visibile come la rete colga la struttura dei dati anche se compie sempre un certo errore.

2.4 Architettura densa

A valle dei risultati ottenuti nei paragrafi precedenti, è stata considerata anche una seconda architettura per la rete VAE. In questa architettura al posto della LSTM è stata utilizzata una rete cosiddetta densa, ovvero i dati di ingresso passano attraverso più strati di neuroni dove tutti gli elementi di uno strato sono connessi con tutti quelli dello strato successivo, una struttura della rete non più ricorrente.

L'origine di questi esperimenti è da ricercarsi nell'osservazione che le reti LSTM sono usualmente utilizzate in contesti dove le sequenze di eventi sono lunghe, come ad esempio nell'elaborazione dei testi o nella sintesi della musica. Nel caso in esame le traiettorie sono limitate a sequenze di 8 eventi di fermata, quindi si è voluto provare se una rete più semplice rispetto alle LSTM potesse offrire risultati validi. In questo approccio l'architettura neurale analizza dunque la sequenza come un pattern unico, ovvero sulla durata del singolo giorno (composto da massimo 8 fermate).

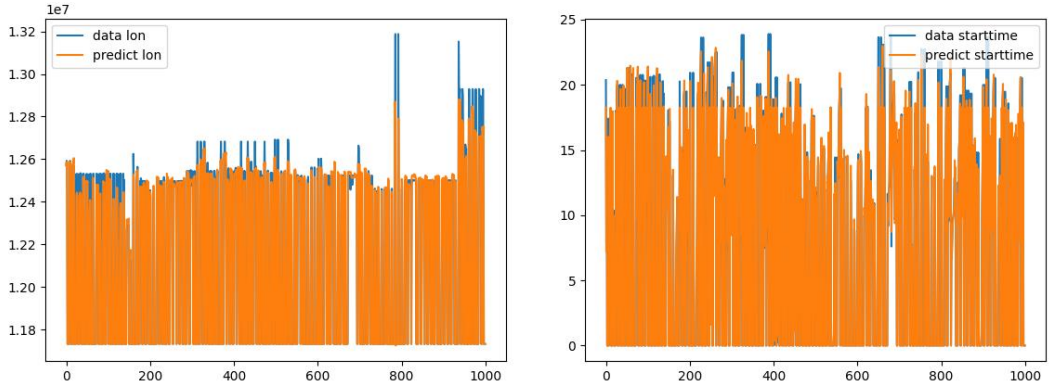

Figura 16. Confronto tra dati di addestramento (in blu) e dati ricostruiti dalla rete (in arancione), longitudine e inizio della sosta (rete LSTM)

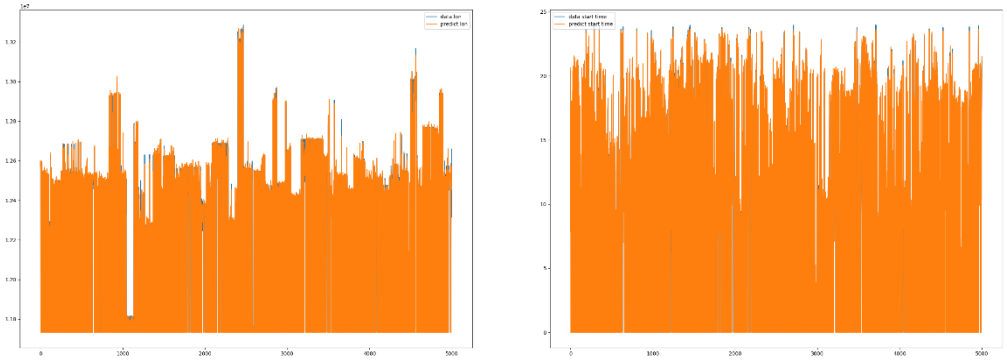

Figura 17. Confronto tra dati di addestramento (in blu) e dati ricostruiti dalla rete (in arancione): longitudine e inizio sosta (rete densa)

Dopo aver compiuto un'analisi dei parametri dell'architettura simile a quella presentata precedentemente, si è implementata una rete con una struttura composta da tre strati completamente connessi di 300, 200 e 100 neuroni nell'*encoder* ed altrettanti in ordine inverso nel *decoder*.

Un esempio dei risultati di addestramento relativi alla rete densa è riportato in Figura 17, dove è chiaramente visibile come l'errore finale sia minore. Comparando la Figura 17 con la Figura 16 si nota come il blu del *dataset* non sia quasi più visibile. Infatti l'errore finale con questa architettura si è rivelato essere di circa un ordine di grandezza minore, scendendo a 10^{-4}.

2.5 Ricostruzione di traiettorie e risultati

Una volta addestrato il VAE può essere utilizzato nella sola parte di *decoder* inserendo valori distribuiti normalmente in ingresso ed ottenendo nuove traiettorie in uscita. Come detto queste traiettorie saranno statisticamente distribuite come quelle di addestramento, ma non saranno esattamente uguali, preservando così la privacy degli utenti che le hanno fornite. In aggiunta è possibile generare traiettorie in qualunque numero, andando così a realizzare *dataset* anche più popolati di quello di origine.

Oltre a questa modalità è anche possibile utilizzare il *dataset* di addestramento, fornendolo in ingresso alla rete, ed ottenere le traiettorie ricostruite in output che, dopo il passaggio attraverso la rete neurale, saranno statisticamente simili ma non uguali a quelle del *dataset*, fornendo così una differente modalità per la modellazione del traffico.

Di queste due procedure, la generazione e la ricostruzione, è stata utilizzata la seconda perché, a valle di un elevato numero di esperimenti, è risultata essere in grado di fornire risultati migliori. Al momento ci si è limitati a un accettazione acritica di tale fatto, limitandosi ad utilizzare i dati ricostruiti dalla rete, ma rimane un fenomeno che necessita di un'approfondita investigazione che esula dalla presente relazione tecnica, ma che è assolutamente opportuna.

Riassumendo, un VAE basato su di un'architettura densa è stato addestrato su circa 900.000 traiettorie giornaliere complessive. Una volta addestrato, il sistema è stato utilizzato per la ricostruzione di nuove traiettorie per auto e conducenti sintetici. I dati in seguito riportati sono rappresentativi della generazione del medesimo numero di traiettorie presenti nel *dataset* di addestramento, in modo da facilitare la comparazione e l'analisi delle prestazioni.

Nei grafici seguenti sono comparate le distribuzioni dei sei descrittori delle singole fermate nei due dataset, quello di addestramento e quello ricostruito dalla rete.

In Figura 18 sono comparate le distribuzioni della latitudine delle soste nel dataset di addestramento, nella parte superiore della figura, e di quella dei dati prodotti dal modello in ricostruzione nella parte inferiore. Le successive figure presentano la stessa comparazione per le quantità: longitudine, ora di inizio della sosta, durata della sosta, lunghezza del viaggio e giorno della settimana.

Per tutte le figure è evidente come il modello sia in grado di cogliere e replicare le distribuzioni statistiche dei dati, smussandole lievemente. E' anche presente un piccolo effetto di bordo: là dove la distribuzione inizia o termina il raccordo con lo zero è dolce; questo è maggiormente evidente nella Figura 21 e nella Figura 22 in prossimità dell'estrema sinistra del grafico.

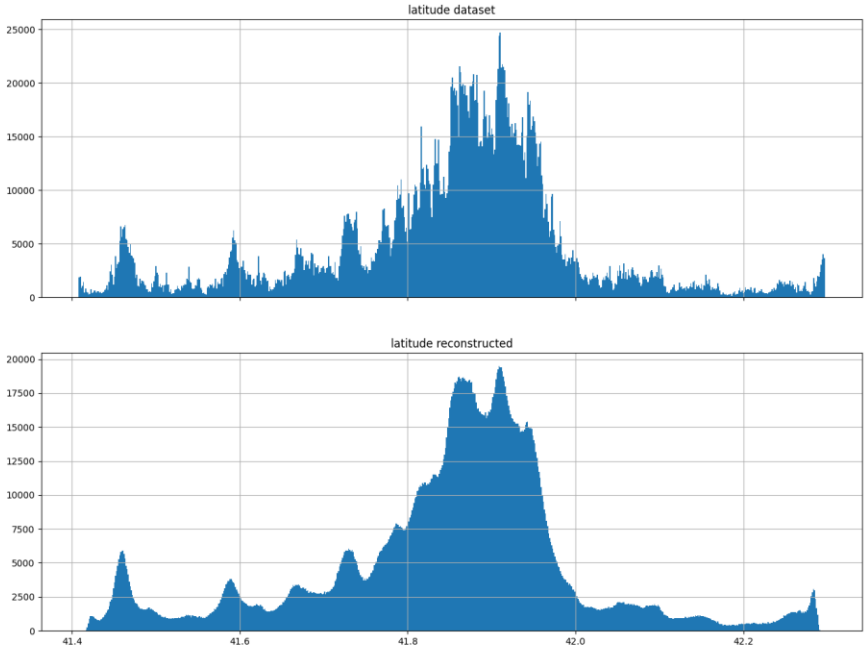

Figura 18. Distribuzione della latitudine: sopra il dataset di addestramento, sotto i dati di uscita del modello

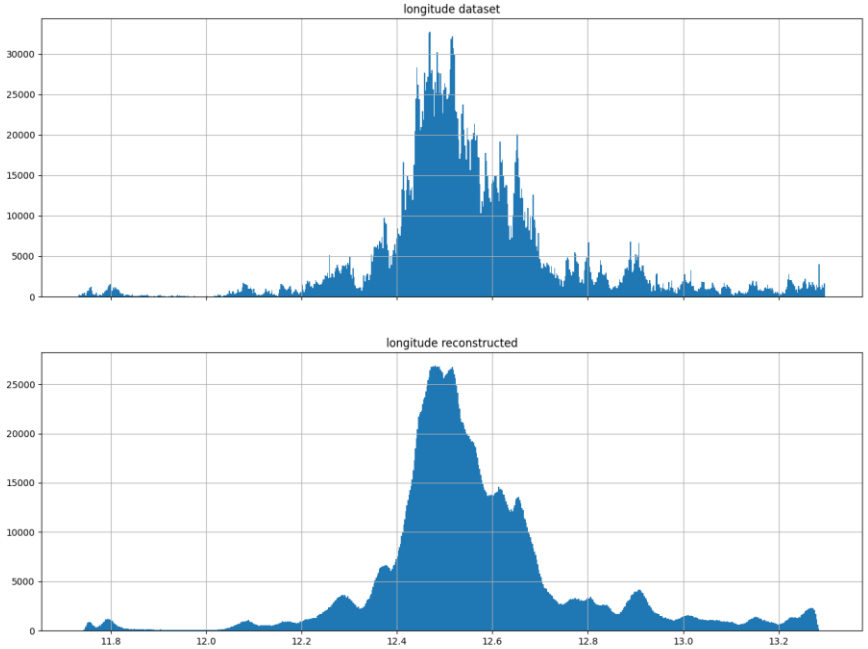

Figura 19. Distribuzione della longitudine: sopra il dataset di addestramento, sotto i dati di uscita del modello

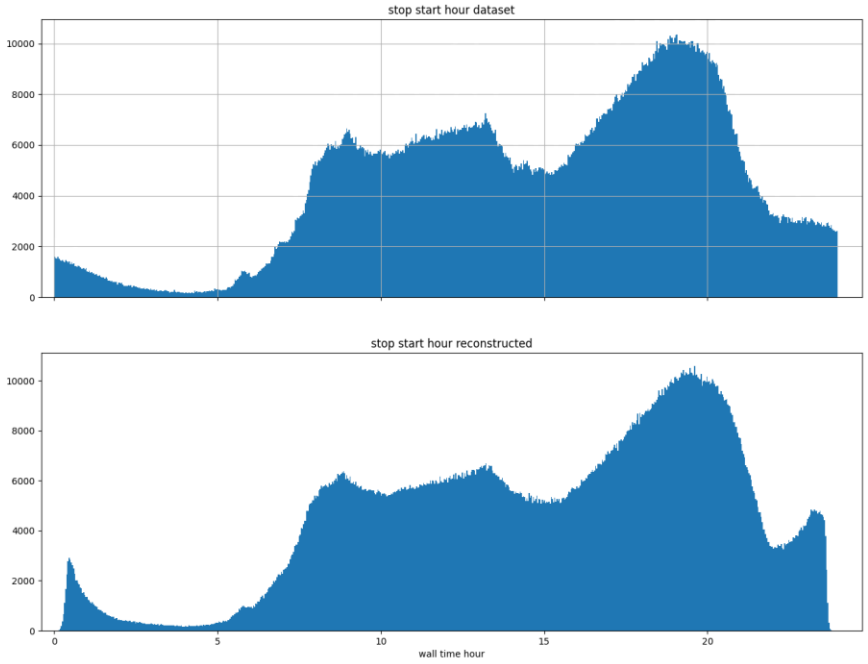

Figura 20. Distribuzione dell'ora di inizio della sosta: sopra il dataset di addestramento, sotto i dati di uscita del modello

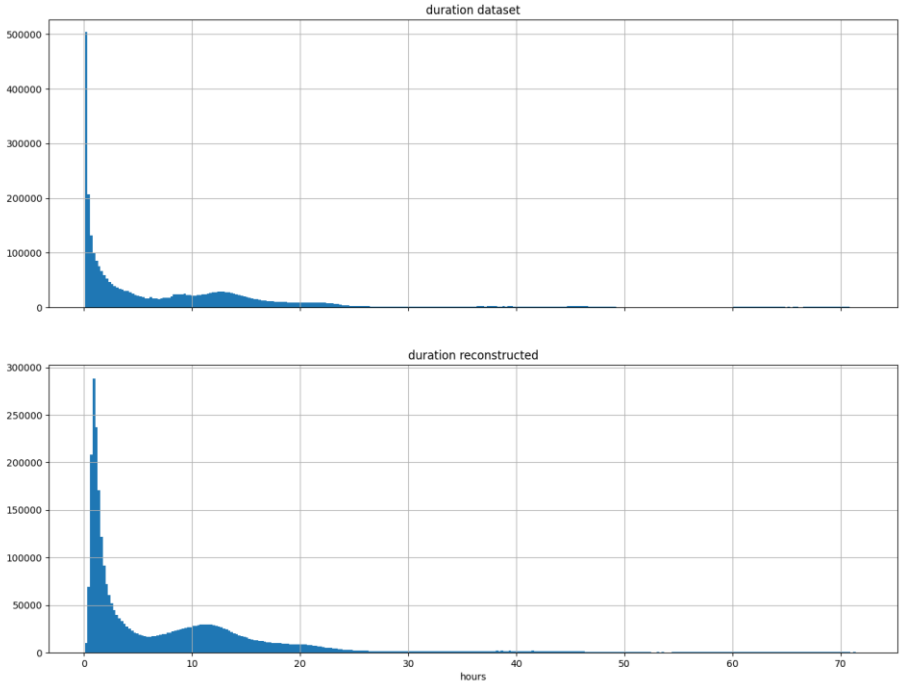

Figura 21. Distribuzione della durata: sopra il dataset di addestramento, sotto i dati di uscita del modello

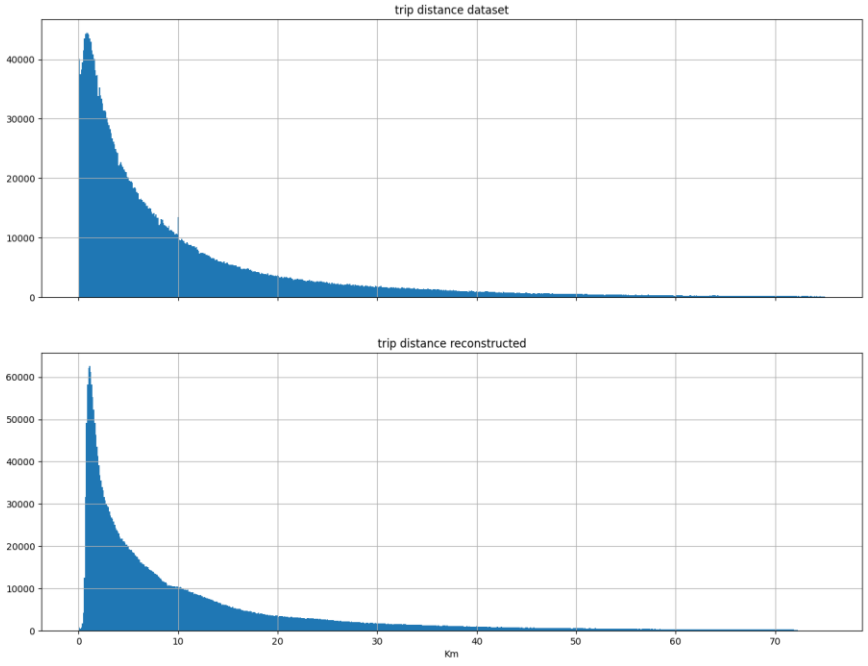

Figura 22. Distribuzione della lunghezza del viaggio: sopra il dataset di addestramento, sotto i dati di uscita del modello

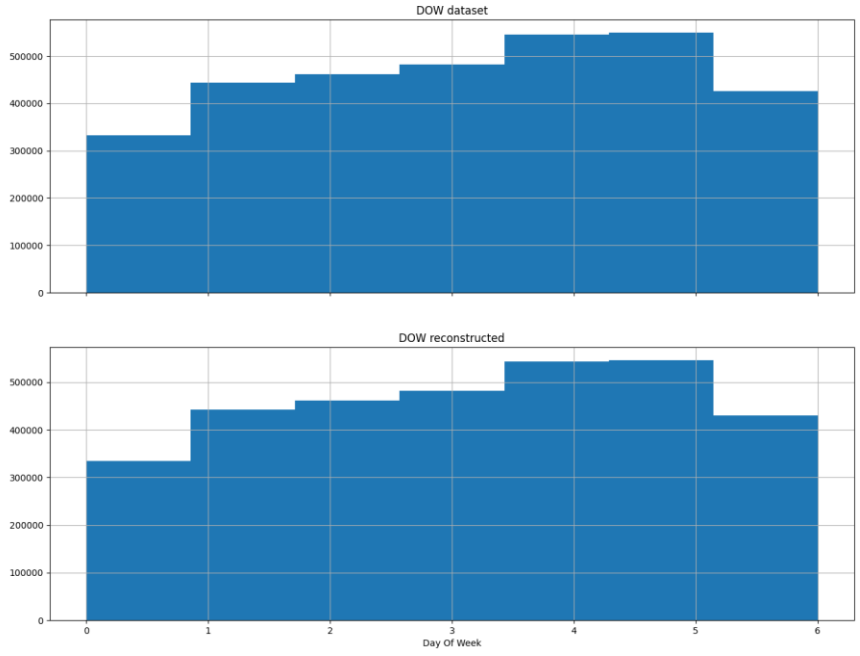

Figura 23. Distribuzione del giorno della settimana: sopra il dataset di addestramento, sotto i dati di uscita del modello

Papapa

In Figura 24 è invece mostrata la distribuzione geografica (in latitudine e longitudine) delle soste presenti nel dataset di addestramento in colore arancione e quelle ricostruite dal modello in blu. Sono state riportate solamente le prime 30.000 fermate su le oltre 3.2 milioni, per permettere una miglior lettura del grafico. Dalla figura si può meglio comprendere come il modello riesca a cogliere la distribuzione spaziale delle soste.

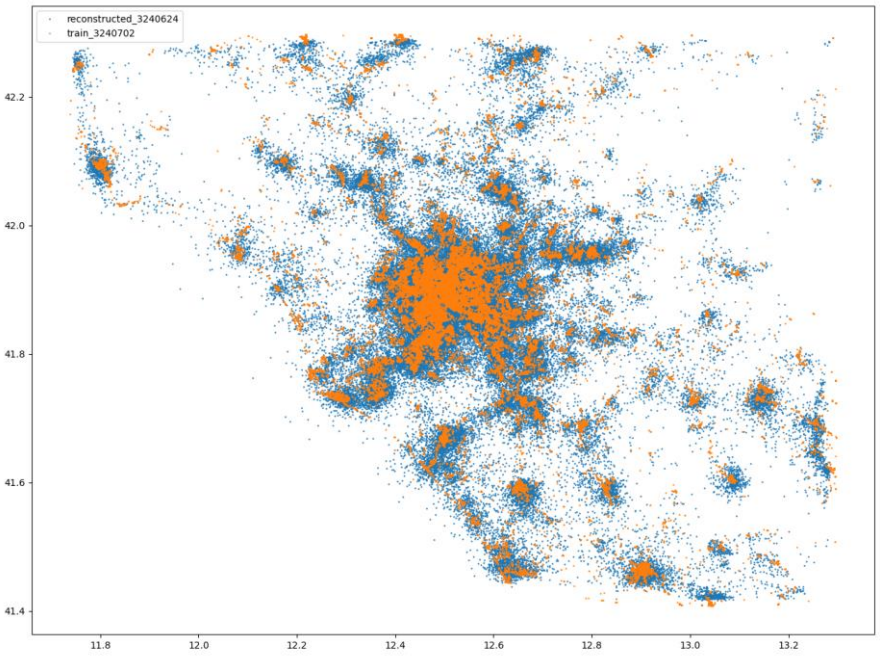

Figura 24. Distribuzione spaziale (latitudine, longitudine) delle prime 30.000 fermate. In arancio il dataset, in blu la ricostruzione del modello

Infine in Figura 25 è mostrato l'istogramma bivariato delle fermate usando come variabili l'ora di inizio della fermata stessa e la sua durata.
Confrontando questa con la Figura 15 si può notare come il modello sia riuscito a cogliere meglio le fermate di breve durata, quelle di colore più intenso.

2.5.1 Dettagli di implementazione

L'implementazione dell'auto encoder variazionale è stata programmata usando il linguaggio Python ed utilizzando la libreria Keras [14], progettata per offrire API coerenti e semplici, ampiamente documentate. Keras, a sua volta, si basa sulla libreria Tensorflow [15] una piattaforma open-source per il machine learning sviluppata da Google.

La fase di addestramento è stata compiuta sia su macchina desktop HP Z800 equipaggiata con scheda grafica NVIDIA per il calcolo ad alte prestazioni, che su Laptop Asus N552VW anch'esso dotato di scheda grafica NVIDIA, ma di minore potenza. L'utilizzo delle schede grafiche porta ad una notevole contrazione dei tempi di calcolo, infatti il produttore delle schede fornisce gratuitamente un'architettura hardware per l'elaborazione parallela (CUDA, Compute Unified Device Architecture [16]) che permette l'utilizzo di Tensorflow con grande efficienza di calcolo.

Le fasi di generazione e ricostruzione possono anche essere implementate su CPU standard, senza utilizzare la GPU (Graphical Processing Unit), essendo computazionalmente meno gravose.

Anche la pre elaborazione e la post elaborazione dei dati sono state implementate in Python sfruttando librerie quali Pandas [17], uno strumento di analisi e manipolazione dei dati open source veloce, potente, e flessibile.

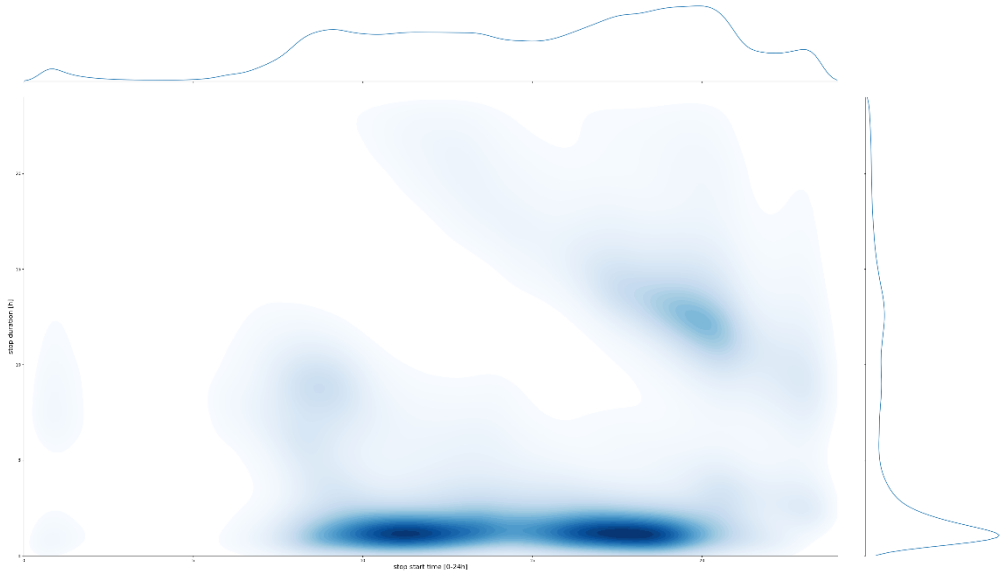

Figura 25. Istogramma inizio sosta vs durata, come ricostruito dal modello

2.6 Post elaborazione dei dati generati

L'uscita del modello è rappresentato da un numero N di traiettorie sintetiche composte da sequenze di 8 soste. Questi dati sono stati post elaborati per generare dati aggregati per area geografica per renderli disponibili a successive elaborazioni ad esempio per il computo di consumi energetici, inquinamento prodotto, etc..

Il primo passo è rappresentato dalla de-normalizzazione mostrata nel paragrafo 2.1, ovvero si sono ricostruite le traiettorie con il loro effettivo numero di stop eliminando la coda a zero eventualmente presente in caso di traiettorie più corte del massimo consentito. Successivamente l'output della rete è stato elaborato in modo da fornire singole traiettorie intese come viaggi da un punto ad un altro, descritte dalle seguenti quantità: *id, from_latitude, from_longitude, start_time, to_latitude, to_longitude, travel_time (in secondi), trip_distance (in metri), day of week, day, parking_time (in secondi),* così come previsto dal software di post elaborazione realizzato in altre Linee di Attività.

Si sono poi aggregati geograficamente i dati seguendo una tassellatura dello spazio in esagoni di diagonale 700 metri, circa 17.000 esagoni. In questo modo è possibile mostrare i dati in termini di flussi entranti e uscenti dai singoli esagoni ad esempio ora per ora. Per ogni esagono e per ogni intervallo di un'ora nella giornata (0-23) sono calcolati i seguenti dati:

1. il numero di veicoli che terminano il proprio viaggio nell'esagono,
2. il numero di veicoli che iniziano il proprio viaggio da quell'esagono,
3. il numero di veicoli in sosta in quell'esagono.

Le prime due proprietà sono banalmente calcolate aggregando i dati sui viaggi descritti in precedenza. L'ultima proprietà richiede di effettuare una somma cumulata delle proprietà 1 e 2 al susseguirsi delle ore della giornata. La differenza tra i due flussi rappresenterà la frazione di auto parcheggiate nell'esagono stesso, una volta che sia noto il valore iniziale di auto ferme. Questo numero è stato derivato dalle stime compiute in [18], basate sul numero effettivo di residenti in ciascun esagono.

A titolo di esempio in Figura 26 è mostrato l'output del modello in termini di flussi veicolari per una data ora del giorno: le ore 8:00. Con ciò si intende che i grafici mostrano le auto entranti ed uscenti tra le ore 8:00 e

le ore 8:59 esagono per esagono, sovrapposti ad una mappa di Roma, limitata al Grande Raccordo Anulare, tratta da Open StreetMap.

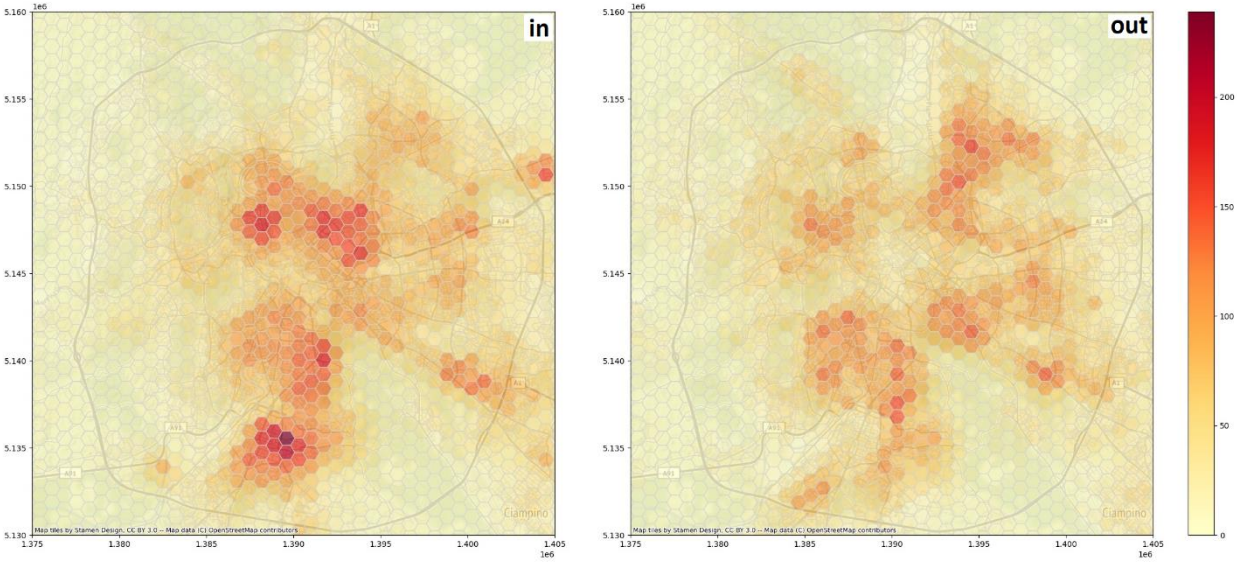

Figura 26. I flussi aggregati per esagoni: a sinistra le auto entranti, a destra quelle uscenti tra le 8:00 e le 8:59

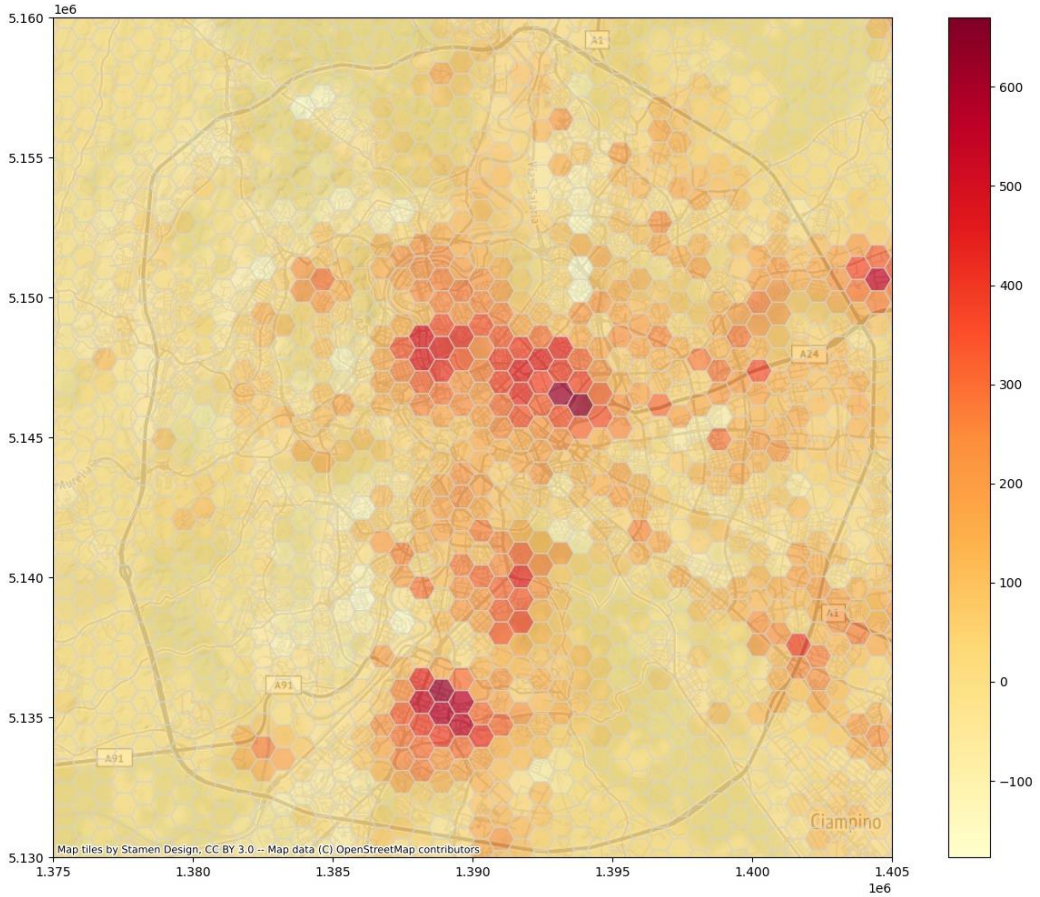

Figura 27. Le auto parcheggiate tra le 9:00 e le 9:59

In Figura 27 sono invece mostrate le auto parcheggiate nell'ora successiva, ovvero tra le 9:00 e le 9:59. La distribuzione delle auto in sosta ricalca fondamentalmente quella delle auto entranti della Figura 26, avvalorando l'ovvia ipotesi che il traffico mattutino sia principalmente diretto verso luoghi di lavoro dove le

auto poi sosteranno per diverse ore. La presenza di valori negativi nella legenda a destra della figura è spiegabile con il problema della determinazione del valore iniziale delle auto in sosta.

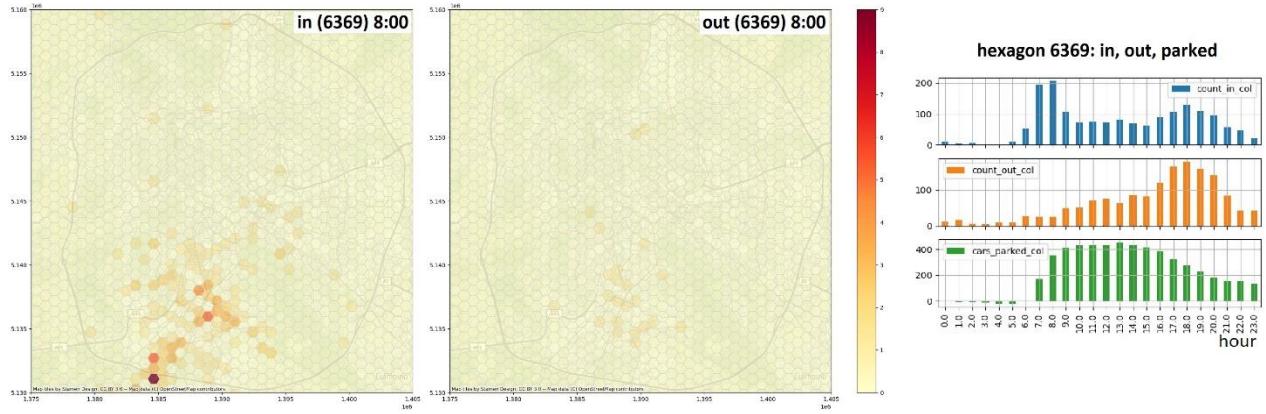

Figura 28. Matrice OD ed istogrammi di flusso dell'esagono 6369 (zona E.U.R.)

Altra interessante elaborazione è presentata in Figura 28, dove è mostrata la matrice origine-destinazione relativa ad un dato esagono. Sulla sinistra sono graficati quegli esagoni da cui si sono originate e in che numero le auto che si sono dirette nell'esagono in esame, al centro gli esagoni verso i quali le auto partite dall'esagono in esame si sono dirette e in che quantità. In aggiunta a destra è anche mostrato l'istogramma complessivo dei flussi della giornata del medesimo esagono (il numero 6369, in zona E.U.R.).

L'esame degli istogrammi sulla destra della figura mostra un risultato coerente con la realtà: molte auto arrivano al mattino con un drastico aumento delle auto parcheggiate fino alle 17-18 quando il deflusso aumenta e le auto parcheggiate diminuiscono.

Va ricordato che il *dataset* originale utilizzato per l'addestramento del modello è comunque basato su di un campione pari a circa il 7% del parco circolante totale. In aggiunta questo campione è rappresentato da autovetture che abbiano montato una scatola nera per motivi assicurativi, quindi è molto probabilmente poco rappresentativo statisticamente dell'universo delle autovetture che insistono sull'area metropolitana di Roma.

I risultati globali sono molto incoraggianti e mostrano la capacità del sistema generativo di cogliere la distribuzione statistica generale delle traiettorie nel set di dati di addestramento.

2.7 Interfaccia con il programma di display

Come precedentemente accennato, il modello qui sviluppato presenta due distinte fasi di utilizzo. Nella prima fase il modello è addestrato sulla base dei dati sperimentali a disposizione, previa una loro opportuna elaborazione. Al termine dell'addestramento il modello ha modificato i propri parametri interni in modo da poter ricostruire nuove traiettorie (sintetiche). La seconda fase è appunto la ricostruzione di nuove traiettorie.

La fase di addestramento è usualmente onerosa dal punto di vista computazionale e quindi richiede tempi abbastanza lunghi. Ad esempio nel caso di addestramenti con 300K utenti, quali quelli descritti, la durata dell'addestramento può facilmente superare le due ore, anche sul citato hardware grafico dedicato. D'altra parte questa è una fase che viene eseguita una sola volta a meno di voler cambiare il *dataset* di addestramento.

La fase di produzione è invece poco onerosa dal punto di vista computazionale, la ricostruzione di una traiettoria si compie in circa 1.2 ms.

I dati prodotti dal modello saranno mostrati a schermo dal programma di input e display sviluppato in un'altra linea di attività di questo progetto, in questa annualità sono quindi stati definiti i requisiti dell'interfaccia tra il modello ed il programma di input e display.

Il modello del traffico genera traiettorie che ricalcano la distribuzione statistica del *dataset* di addestramento, ovvero un mese di viaggi raccolti nell'area della provincia di Roma. La generazione è completamente insensibile al contenuto, ovvero non è possibile scegliere a priori delle caratteristiche delle traiettorie quali area di partenza o di arrivo, giorno, ora, etc. La scelta può essere però compiuta ex post sull'insieme di tutte le traiettorie generate. Ad esempio per poter generare le traiettorie del fine settimana è necessario generare tutte le traiettorie e poi si potranno selezionare quelle con giorno della settimana sabato e domenica. Uguale discorso nel caso di scelta di una particolare zona geografica della città.

Si prevede un'architettura client-server, eventualmente sul medesimo computer, i cui moduli dialoghino tra loro attraverso il protocollo *http* in modalità REST (REpresentational State Transfer). Il client è il programma di visualizzazione mentre il server è il simulatore che genera le traiettorie.

L'interfaccia di gestione del processo di generazione è sul client e deve prevedere la possibilità di indicare:

- l'IP e la porta del server del simulatore;
- il numero di viaggi da generare;
- i giorni della settimana relativi alle traiettorie da generare (ad esempio con una *checkbox*);
- zona della mappa: latitudine longitudine in basso a destra ed in alto a sinistra (totale 4 valori) + un *radio button* per indicare se traffico uscente (origine) o entrante (destinazione), eventualmente anche un *checkbox* per indicare l'intera mappa;
- bottone di start che avvia il processo.

Una volta riempiti i campi il bottone di start invia una richiesta http di tipo POST verso il server all'IP con un file di tipo JSON con i dati anzidetti, ricevendo in risposta un ID della simulazione.

Al termine dell'elaborazione riceve un pacchetto http di fine elaborazione. A questo punto può scaricare dal server i file con i risultati (24 file x N, uno per ogni ora per ognuno degli N giorni richiesti) che saranno nella directory 'output_simdata'. La formattazione dei dati nel file in uscita sarà la seguente:

un elenco di esagoni con numero di veicoli in arrivo, durata media sosta, numero veicoli in partenza, numero auto parcheggiate, ricalcando il formato JSON dei risultati disegnato nel corso del precedente Piano Triennale.

3 Conclusioni

Questo Rapporto Tecnico ha presentato le attività del secondo anno del Piano Triennale 2019-2021 nella Linea di Attività 2.20, Sviluppo di moduli di calcolo e simulazione della mobilità urbana e di protocolli di comunicazione IoT per la gestione real time di flotte elettriche.

Il nocciolo delle attività è stata la realizzazione di un modello di simulazione di richiesta di traffico per la Città Metropolitana di Roma. Questo modello è stato realizzato utilizzando un approccio basato sull'utilizzo di algoritmi di Machine Learning ed in maggior dettaglio tramite l'impiego di un auto encoder variazionale (VAE). Il VAE è stato addestrato a riprodurre le caratteristiche statistiche dei viaggi contenuti nel *dataset* Octo Telematics relativo al mese di maggio del 2013.

Una volta addestrato il modello, esso può essere utilmente impiegato per produrre traiettorie di traffico sintetico che ricalchino le caratteristiche statistiche del *dataset* originale. I risultati qui presentati mostrano che il modello replica con buona accuratezza la distribuzione statistica dei dati di addestramento e ciò permette il superamento del problema della privacy degli utenti e dell'eventuale limitatezza del dataset.

Infatti il modello è in grado di produrre un numero qualunque di traiettorie che saranno statisticamente simili a quelle degli utenti facenti parte del dataset di addestramento, ma non saranno uguali. Ciò permette da un lato di salvaguardare la loro privacy e dall'altro di poter produrre quante traiettorie si desideri.

Questo modello sarà utilizzato nelle attività del terzo anno del Piano Triennale per generare la richiesta di traffico alla quale dare una parziale risposta tramite il cosiddetto ride-sharing, con l'obiettivo di ridurre il traffico complessivo.

Tra gli spunti di ricerca da approfondire vi è quello di comprendere meglio le differenti prestazioni del modello addestrato se utilizzato nella modalità di generazione piuttosto che in quella di ricostruzione, come esposto nel paragrafo 2.5.

Ulteriore auspicabile attività di ricerca è quella di verificare le prestazioni del sistema su altri *dataset* di addestramento comparando i risultati.

4 Riferimenti bibliografici

1. S. Taraglio, S. Chiesa, V. Nanni, "Modello per la simulazione della mobilità urbana veicolare", Report RdS/PTR2019/019, (2019)
2. Octo Telematics, https://www.octotelematics.com/it/home-it/, ultimo accesso 29/04/2021
3. M. C. Gonzalez, C. A. Hidalgo, and A.L. Barabasi: Understanding individual human mo-bility patterns. Nature, 453(7196):779–782, 2008.
4. C. Song, Z. Qu, N. Blumm, and A.L. Barabasi: Limits of predictability in human mobility. Science, 327(5968):1018–1021, 2010.
5. Alessandretti, L., Sapiezynski, P., Lehmann, S. and Baronchelli, A.: Evidence for a Conserved Quantity in Human Mobility. Nature Human Behaviour, 2, pp. 485-491. Doi: 10.1038/s41562-018-0364-x, 2018.
6. Harshvardhan GM, Mahendra Kumar Gourisaria, Manjusha Pandey, Siddharth Swarup Rautaray: A comprehensive survey and analysis of generative models in machine learning. Computer Science Review, Volume 38, 100285, ISSN 1574-0137, https://doi.org/10.1016/j.cosrev.2020.100285, 2020.
7. M. Yin, M. Sheehan, S. Feygin, J. Paiement and A. Pozdnoukhov: A Generative Model of Urban Activities from Cellular Data. IEEE Transactions on Intelligent Transportation Sys-tems, vol. 19, no. 6, pp. 1682-1696, Doi: 10.1109/TITS.2017.2695438, 2018.
8. Lin, Z., Yin, M., Feygin, S., Transportation, M.S., Paiement, J., Cee, A.P.: Deep Generative Models of Urban Mobility. Proceedings of KDD'17, August 13-17, 2017, Halifax, Nova Scotia, Canada, 2017.
9. Foster D., "Generative Deep Learning", O' Reilly, Sebastopol, CA, USA (2019).
10. D.P. Kingma, M. Welling, "Auto-encoding variational Bayes", International Conference on Learning Representations, ICLR 2014, Apr 14 - 16, 2014, Banff, Canada.
11. S. Hochreiter and J. Schmidhuber: Long Short-Term Memory. Neural Computing, Vol. 9, 8, 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735, 1997.
12. Graves, A.: Generating Sequences With Recurrent Neural Networks. arXiv: 1308.0850, 2013.
13. K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra: Draw: A recurrent neural network for image generation. arXiv:1502.04623, 2015.
14. Keras, https://keras.io/, ultimo accesso 29/04/2021
15. Tensorflow, https://www.tensorflow.org/, ultimo accesso 29/04/2021
16. CUDA, https://developer.nvidia.com/cuda-zone, ultimo accesso 29/04/2021
17. Pandas, https://pandas.pydata.org/, ultimo accesso 29/04/2021
18. B. Monechi, I. Biazzo, V. Loreto, F. Tria, "Modelli per la simulazione di scenari di elettrificazione della mobilità veicolare urbana", Report RdS/PAR2016/235, (2016)

5 Elenco delle figure

6 Abbreviazioni ed acronimi

VAE: Variational Auto Encoder, auto encoder variazionale
RNN: Recurrent Neural Network, rete neurale ricorrente
LSTM: Long Short Term Memory, memoria a lungo e breve termine, un tipo di RNN
MSE: Mean Square Error, errore quadratico medio
REST: REpresentional State Transfer, sistema di trasmissione dati su http
JSON: Java Script Object Notation, format per lo scambio dati