Estimating Travel Cost Distributions of Paths in Road Networks using Dual-Input LSTMs

Christopher Hansen Nielsen
Department of Computer Science, Aalborg University
Aalborg, Denmark
chni15@student.aau.dk

Simon Makne Randers
Department of Computer Science, Aalborg University
Aalborg, Denmark
srande15@student.aau.dk

ABSTRACT

Thanks to recent advances in sensor technologies, detailed travel cost information are becoming increasingly available. Such data provide a solid data foundation to capture traffic uncertainty, e.g., in the form of travel cost distributions. We study the problem of estimating travel cost distributions of paths in a road network using floating car data. We consider two different distributions structures, namely histograms and Gaussian Mixture Models. Given a path and a departure time, we aim at estimating the travel cost distribution of the path. To this end, we propose a dual-input long-short term memory (DI-LSTM) model. We introduce two new gates with the purpose of combining two input distributions in every iteration, where one distribution is an edge' distribution, and the other is the distribution of the pre-path until the edge, which is obtained from the previous DI-LSTM unit. Empirical studies on a large trajectory dataset offer insight into the design properties of the DI-LSTM and demonstrate that DI-LSTM out-performs classic LSTM, especially for longer paths. Furthermore, we determine that Gaussian Mixture Models has the potential to be a better suited distribution structure than histograms.

1 INTRODUCTION

The quality of a wide variety of intelligent transportation services, e.g., advanced traffic management and autonomous driving, relies on accurate travel cost information. We are now witnessing increasing needs for high-resolution travel cost information that fully captures traffic uncertainty, which helps improve user experience and service quality. Simply using average costs may not suffice anymore.

For example, considering travel speed as the cost, when an autonomous taxi follows a path to an airport with a tight deadline, having the speed distribution helps the taxi better understand the risk of being late, which cannot be achieved by only using the average speed. Assume that the path has a speed (km/h) distribution of {([20, 30), 0.2), ([30, 40), 0.3), ([40, 50), 0.5)}, meaning that the probability of the traveling speed on the path from 20 to 30 km/h is 0.2, from 30 to 40 km/h is 0.3, and from 40 to 50 km/h is 0.5. If the path is 30 km, we are able to derive a travel time (minutes) distribution: {((36, 45], 0.5), ((45, 60], 0.3), ((60, 90], 0.2)}. If the deadline is 60 minutes, the taxi has a 20% risk of running late. In contrast, when only using average speed, which is 36 km/h, we derive an average travel time of 50 min, which is unable to accurately assess the risk of arriving late. This may result in unpleasant results, e.g., missing a flight or waste of time.

With the increasing digitization of transportation, large amounts of floating car data (FCD) are becoming available. We aim to exploit such data for accurately estimating the cost distribution of a path. However, this is non-trivial due to the data sparseness challenge. A road network often has exponentially many meaningful paths; it is thus almost impossible to have sufficient trajectory data to cover all possible paths to derive their distributions, even with large amounts of trajectories [2]. In contrast, it is more likely to have trajectories on all edges to derive cost distributions for the edges. In case that the trajectory data is also too sparse to cover all edges, studies exist that are able to propagate cost distributions from edges with trajectory data to edges without trajectory data [3]. Thus, we consider a problem setting where we have cost distributions of all edges in a road network. The target is to estimate the cost distribution of an arbitrary path in the road network.

To solve the path cost distribution estimation problem, we present a dual input long short term memory (DI-LSTM) model. Given the cost distributions of the edges in a path, DI-LSTM is able to accurately estimate the cost distribution of the path. We introduce two new gates, a *combine gate* and an *additional input gate*, to thoroughly capture the dependencies among the edges' cost distributions, which improves the accuracy when comparing to classic LSTM. Specifically, the combine gate combines two input distributions—a cost distribution of an edge and a cost distribution of the pre-prath until the edge, which is obtained from the previous DI-LSTM. The additional input gate produces a cost distribution of the new pre-prath, including the current edge, which is then fed into the next DI-LSTM. Each cost distribution is accompanied by the length of the edge/pre-prath it represents.

The main focal points of this work are summarized as follows. First, we formalize the path cost distribution estimation problem. Second, we propose a novel dual-input LSTM to better capture the dependencies of edge cost distributions and thus improve accuracy. Third, we demonstrate that the

proposed model outperforms multiple baselines using a large real-world trajectory dataset.

2 PRELIMINARIES

We introduce important concepts and formulate the problem on *path cost distribution estimation*.

Road Networks: A *road network* is modeled as a graph $G = \langle V, E \rangle$, where V is a set of vertices and a vertex $v_i \in V$ represents a road intersection, and $E \subseteq V \times V$ is a set of edge and an edge $e_k = (v_i, v_j)$ represents a road segment.

Paths and Trajectories: A *path* $P = \langle e_1, e_2, e_3, \ldots, e_n \rangle$ is a sequence of edges, where two consecutive edges in the path e_i and e_{i+1}, where $1 \leq i \leq n-1$, must share a vertex such that it is possible to traverse from e_i to e_{i+1}. A *sub-path* of path P is a sub-sequence of the edges in P. *Pre-path* $P^{(j)}$ of path P is a sub-path of P with the first j edges. For example, given path $P = \langle e_1, e_2, e_3, e_4, e_5 \rangle$, we have pre-path $P^{(3)} = \langle e_1, e_2, e_3 \rangle$.

A *trajectory* is a sequence of *trajectory records* that correspond to a trip on a path. Here, a trajectory record is in the form of $\langle e_i, t_i, d_i \rangle$, indicating that a vehicle traversed edge e_i at timestamp t_i with duration d_i.

Travel Cost Distributions: We consider two distribution structures; histograms and Gaussian Mixturee Models.

A histogram is a collection of $(bucket, probability)$ pairs with each *bucket* representing a speed range $[l, u)$ that is specified by a speed lower bound l and speed upper bound u. The probabilities over different buckets in a histogram sum to 1.0. Two histograms are *homogeneous* if they have the same buckets. In this paper, we use homogeneous histograms to represent the speed distributions of different edges and paths, where each histogram has 22 buckets, and each bucket represents a speed range of 2 m/s, thus covering speeds up to 44 m/s.

To obtain a histogram for an edge, we consider all available trajectory records on the edge. Then, we distribute these trajectory records to different buckets and derive the corresponding probabilities. For instance, assume that in total, 100 trajectory records exist on edge e_i, and 20 out of the 100 trajectory records have speeds 10 m/s to 12 m/s, then the probability of bucket [10, 12) is 0.2.

Similarly, we are able to derive a speed histogram for a path using trajectories that occurred on the path. However, due to the data sparseness challenge, many paths are without any trajectories.

A Gaussian Mixture Model is a probabilistic model designed for representing multiple sub-distributions within the overlying distribution. Given enough components, a Gaussian mixture is capable of modeling an arbitrary number of probability densities [1]. We choose a Gaussian mixture over a single Gaussian distribution, as the underlying data can be multi-modal - meaning there can be multiple peaks.

$$p(y|x) = \sum_{c=1}^{C} \alpha_c(x) D(y|\mu_c(x), \sigma_c(x), ...) \tag{1}$$

The conditional probability for a mixture is defined as seen in equation 1. Here c denotes the index of the respective mixture component, α denotes the mixing parameter, D represents the corresponding distribution, μ represents the mean, and σ is the standard deviation. Using a Gaussian mixture allows us to model a multi-modal problem while representing the overall distribution uniformly. This is essential as we aim to use the Gaussian mixture as input for a neural network and thus need a uniform structure. Similar to how we obtain the histogram distributions, we identify all edges and paths with at least five trajectory records and construct the corresponding Gaussian mixture model using a Mixture Density Network [1].

One advantage of using Gaussian mixtures is that it also allows the consideration of travel time as opposed to travel speed distributions, as the Gaussian mixture retains the same uniform structure. We can still use the same number of mixtures in the Gaussian, whereas if we were using a histogram, we would have to increase the range of the buckets, or increase the number of buckets in order to represent travel time in such a data-structure. This also allows us to consider the travel time problem, rather than only travel speed. We can then consider both travel time and travel speed as a cost.

Mixture Density Network: In order to obtain the Gaussian mixtures we utilize a Mixture Density Network (MDN) [1]. An MDN is a class of Deep Neural Networks, which is designed to model general conditional probability densities. As specified previously, we are interested in generating Gaussian mixtures. As such, we specify the output layers of a standard MDN to specifically predict Gaussian mixtures. We employ a fully connected layer with ReLU as activation to get a hidden representation before we pass it through three different layers to get the α, σ, and μ values.

$$h(x) = FC(x)$$
$$\alpha(x) = \delta(FC(h(x))$$
$$\sigma(x) = ELU(FC(h(x)) + 1$$
$$\mu(x) = FC(h(x)$$

We enforce the standard deviation to be $\sigma(x) > 0$ by using a variation of the ELU activation with an offset of +1. Lastly, as the output layer, the MDN employs a concatenation layer to combine the three parameter-vectors, as seen in equation 2. The MDN utilizes a Negative Log-Likelihood (NLL) loss function.

$$MDN(x) = CONCAT(\alpha(x), \sigma(x), \mu(x)) \tag{2}$$

Since the MDN does not need any truth labels for the training process, we can identify all edges, with at least five observations, that are not part of a path, and use these for training the MDN. This allows us to use the same data as with the histograms for representing the edges. The difference is that this time, the data is being passed through the MDN and converted to Gaussian mixtures instead of being histogram distributions.

Path Cost Distribution Estimation: Given a departure time t, a path $P = \langle e_1, e_2, ..., e_n \rangle$, and the travel cost distribution $D(e_i)$ of each edge $e_i \in P$, path cost distribution estimation aims at estimating the travel cost distribution $D(P)$ of path P.

3 DUAL INPUT LSTM

We propose a dual input long short term memory (DI-LSTM) model to solve the path cost distribution estimation problem. We choose LSTM as the foundation as it is shown to be effective in capturing dependencies in the input sequences. In our setting, a path is a sequence of edges; it is thus essential to capture the dependencies of the cost distributions of the edges in the path to accurately estimate the cost distribution of the path. To better fit this problem setting, we introduce two new gates inside an LSTM unit, which are detailed in the following.

DI-LSTM Units

Figure 1 shows a DI-LSTM unit. The basic structure resembles a classic LSTM unit with gating mechanisms, such as a forget gate, an input gate, and an output gate, and multiple hidden states, such as $c(i)$ and $h(i)$. The gates of an LSTM unit are defined as:

$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i)$$

$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f)$$

$$o_t = \sigma(w_o[h_{t-1}, x_t] + b_o)$$

where i_t, f_t, and o_t represents the input, forget, and output gate respectively - w_x represents the weights of gate x, h_{t-1} represents the output of the previous LSTM unit, x_t represents the current input, and b_x is the bias for gate x. Next, we elaborate the two new gates—a *combine gate* and an *additional input gate*, which are highlighted in Figure 1.

Combine Gate: The combine gate combines two input distributions. The first input is the edge distribution $D(e_i)$. The second input is an aggregated distribution that represents the pre-path $P^{(i-1)}$ until edge e_i, which is the output of the additional input gate from the previous DI-LSTM unit. Both distributions are passed in along with their respective edge/pre-path length.

We introduce two variations of the combine gate—a math gate specifically designed to combine histograms, and a Multi-Layer Perception (MLP) gate designed to consider all distribution types.

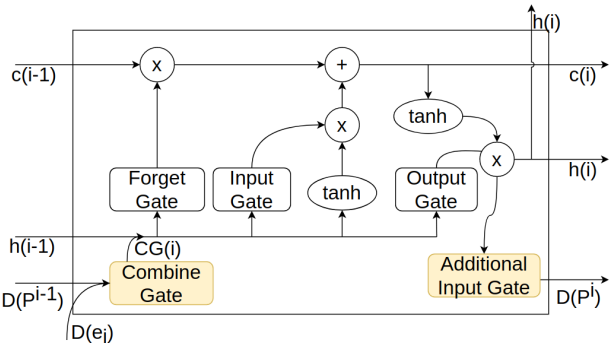


Figure 1: The Dual-Input LSTM unit.

Math gate (MG): The math gate normalizes the two input distributions based on the lengths of edge e_i and the pre-path $P^{(i-1)}$. Specifically, the k-th bucket of the combined distribution, denoted as $CG_{MG}(i)[k]$, has the normalized probability shown in Equation 3.

$$CG_{MG}(i)[k] = \frac{D(e_i)[k] * e_i.len + D(P^{(i-1)})[k] * P^{(i-1)}.len}{e_i.len + P^{(i-1)}.len} \quad (3)$$

MLP gate (MLPG): The MLP gate, see equation 4, employs fully connected layers with ReLU as the activation functions in the internal layers and either an MDN or a softmax (denoted by $\delta(\cdot)$) as the final layer, as it makes sure the output represents a meaningful distribution.

$$CG_{MLP}(i) = \delta\big(FC(D(e_i), D(P^{(i-1)}))\big) \quad (4)$$

Additional Input Gate: After computing the hidden state $h(i)$ inside an LSTM unit, we apply an additional input gate (AIG) to force it into a meaningful distribution, which represents the distribution of pre-path $P^{(i)}$. The AIG is achieved by passing the hidden state $h(i)$ through δ layer, which gives:

$$AIG(i) = D(P^{(i)}) = \delta(FC(h(i))) \quad (5)$$

The Full Model

Figure 2 shows a histogram variation of the full model with a sequence of DI-LSTM units and some fully connected layers in the end. The i-th DI-LSTM unit takes an input edge e_i's distribution $D(e_i)$ and an additional input $D(P^{(i-1)})$ that is obtained from the additional input gate of the previous DI-LSTM unit. Each distribution is accompanied with the length of the edge as a local feature. Meaning the input size in this variation is 23 variables. A special case is the first DI-LSTM unit where we do not have a DI-LSTM unit in front of it. It takes two inputs $D(e_2)$ and $D(P^{(1)})$. Since $P^{(1)}$ only includes

the first edge e_1 in path P, we have $D(P^{(1)}) = D(e_1)$, which can be obtained directly.

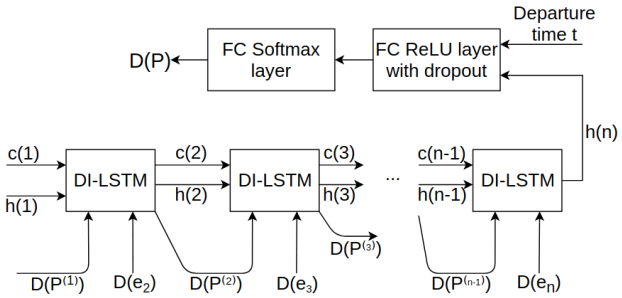


Figure 2: Full DI-LSTM Model.

The output vector $h(n)$ from the last DI-LSTM unit is concatenated with the departure time information and then fed into a fully connected layer with ReLU as the activation function. We consider time of day as the departure time information and is encoded into a 24-dimensional one-hot vector. We also use dropout in this layer to avoid over-fitting. Finally, we apply a fully connected layer with softmax as the activation function to produce the final output vector $D(P)$ that represents the speed distribution of path P.

Loss Function

We design a loss function that measures the discrepancy between a ground truth distribution, denoted as D_1 and an estimated distribution, denoted as D_2. In particular, the loss function Equation 6 combines two metrics that measure the similarity between two distributions— *Kullback-Leibler (KL) Divergence* and *Histogram Intersection*.

$$\text{Loss}(D_1, D_2) = KLD(D_1, D_2) + (1 - INT(D_1, D_2)) \quad (6)$$

where

$$KLD(D_1, D_2) = \sum_{i=1}^{N} D_1[i] \ \ \log(\frac{D_1[i]}{D_2[i]})$$

$$INT(D_1, D_2) = \sum_{i=1}^{N} \min(D_1[i], D_2[i])$$

and N is number of buckets in a histogram. Here, a small KL divergence value indicates that the two distributions are similar, while a small intersection value indicates that they are dissimilar. Since the intersection value is between 0 and 1, we use $(1 - Int(H_1, H_2))$ in the loss function. Thus, minimizing the loss function ensures accurate estimation.

W.r.t the Gaussian mixture models, we integrate over the probability density function (PDF) generated from the Gaussian mixture to achieve a similar bucket structure to that of histograms, an example of this can be found in figure 5a. Allowing us to use the same loss function.

4 EXPERIMENTS

Data. We use a large GPS data set with 1,107,684,238 GPS points, collected on the Danish road network during the period 2012-04-01 to 2014-12-20. After map matching, we select 90,688 paths that are traversed by at least five trajectories, and thus we are able to derive ground truth cost distributions for the paths. We then categorize these paths into four different groups w.r.t. path cardinalities (i.e., the number of edges in the paths): *short*, *medium*, *long*, and *very long*. The average path lengths in kilometers (KM) are summarized in Table 1. For each path cardinality, we randomly choose 10% as testing data and the remaining 90% as training data.

Table 1: Path Categories.

	Short	Medium	Long	Very Long
Cardinality	[2,30]	[31, 70]	[71, 115]	[116, 169]
Length (KM)	2.57	17.02	28.58	41.36

Baselines. We consider five different methods. (1) Length-based normalization (LBN): we normalize the edge distributions w.r.t. their edge lengths and then aggregate the normalized distributions using Equation 3, i.e., the math gate, to obtain $D(P)$. (2) Linear regression (LR): we consider estimating $D(P)$ as a regression problem and use LR as a non-deep learning method. (3) classic long short term memory network (LSTM); (4) DI-LSTM using the math gate (DI-LSTM-MG); (5) DI-LSTM using the MLP gate (DI-LSTM-MLP).

Evaluation Metrics. We employ three popular metrics that are able to quantify the similarity between two distributions. These include *Intersection* and *KL Divergence*, which are already introduced when introducing the loss function in Section 3. The third metric is *Correlation*, defined as follows.

$$\text{COR}(D_1, D_2) = \frac{\sum_{i=1}^{N}(D_1[i] - \bar{D_1})(D_2[i] - \bar{D_2})}{\sqrt{\sum_{i=1}^{N}(D_1[i] - \bar{D_1})^2(D_2[i] - \bar{D_2})^2}}$$

where $\bar{D_k} = \frac{1}{N}\sum_{j=1}^{N} D_k[j]$ and N is the total number of histogram buckets. Note that *lower* KL divergence values and *higher* intersection and correlation values suggest the estimated distributions are more accurate. Similar to the loss function, we integrate over the PDF, allowing us to utilize the same metrics for evaluation when considering the Gaussian mixtures.

Histogram Experiments

In the first set of experiments, we explore the histogram distribution structure. We consider all baselines and models in this set of experiments.

Preliminary Results. We start by testing the two gates, *MLPG* and *MG*, separately. We train the MLPG on 90% of

Table 2: Mean accuracy for histogram representations.

Method	All Paths			Short			Medium			Long			Very Long		
	COR	INT	KLD	COR	INT	KLD	COR	INT	KLD	COR	INT	KLD	COR	INT	KLD
LBN	0.405	0.313	1.449	0.657	0.532	1.014	0.417	0.320	1.394	0.352	0.261	1.626	0.285	0.233	1.660
LR	0.496	0.435	2.546	0.715	0.588	0.717	0.405	0.366	2.725	0.418	0.378	3.208	0.636	0.543	2.093
LSTM	0.810	0.684	0.478	0.782	0.654	**0.528**	0.832	**0.701**	0.423	0.761	0.645	0.587	0.877	0.746	0.337
DI-LSTM-MG	0.805	0.678	0.504	0.757	0.631	0.588	**0.836**	0.700	0.424	0.750	0.640	0.645	0.877	0.735	0.343
DI-LSTM-MLP	**0.821**	**0.692**	**0.450**	**0.783**	**0.657**	**0.528**	0.834	0.698	**0.417**	**0.789**	**0.666**	**0.527**	**0.890**	**0.756**	**0.313**

paths with a cardinality of 2, using the remaining 10% for testing and since the MG cannot be trained, we just test on the same 10% testing data. Table 3 reports the mean performance of the two gates. We see that the *MLPG* performs significantly better than the *MG*, indicating that there exists complex correlations between edge distributions, that a simple equation cannot completely capture.

Table 3: Mean Gate Accuracy.

Gates	COR	INT	KLD
MG	0.427	0.464	1.696
MLP	**0.758**	**0.645**	**0.611**

Results. We report the mean accuracy w.r.t. different path categories in Table 2 using 10-fold cross-validation. Similar to the base gate performance, LBN performs the worst. Simply aggregating edge distributions and normalizing them according to edge lengths is very inaccurate. This suggests that there exist complex relationships among the edge distributions, and they are not proportional to the lengths. LR also performs poorly as LR is unable to capture non-linear relationships among the edge distributions.

The deep learning-based methods, i.e., LSTM, DI-LSTM-MG, and DI-LSTM-MLP, achieve significantly better results compared to LBN and LR, as such deep learning models can capture complex and non-linear relationships. More specifically, the DI-LSTM-MLP achieves better accuracy than DI-LSTM-MG does. This is consistent with the previous observations that complex and non-linear correlations among edge distributions exist, and a simple length based normalization does not work well.

To further this, looking at the performance of the base gates, MLPG and MG, we see that MLPG outperforms MG with a relative difference of 77.59%, 63.94%, and 38.99% in COR, KLD, and INT respectively. Whereas, looking at the full models, the DI-LSTM-MLP outperforms the DI-LSTM-MG with a relative difference of 2.05%, 10.75%, and 1.93% in COR, KLD, and INT respectively. This indicates that while there is large difference in the separated gate performance, it does not appear to be as impactful as expected in the full model. We speculate this is caused by the fact that the MG is deterministic whereas the MLPG is not. Specifically, we expect that the deterministic feature of the MG allows the LSTM unit around it to be trained accordingly, as it quickly learns that the outcome of the MG deterministic, whereas it has a harder time learning the outcome of the MLPG. This is consistent with earlier observations of less optimized versions of the MLPG, where the baseline LSTM would also outperform the DI-LSTM-MLP model.

When considering the different path categories, we notice that DI-LSTM-MLP becomes more superior when estimating distributions for long and very long paths. This is important because the longer a path is, it is less likely that we have trajectories occurred on the path. Thus, the need is greater to estimate the speed distribution for such long and very long paths.

Figure 3 shows a box plot for correlation-based accuracy. The results are consistent with the observations from table 2. Interestingly, we see that the median value for most categories is significantly better than the mean value, suggesting that, while the model produces some poor results, the majority of the time it will produce predictions which are better than the mean result. We see again, that the LBN and LR perform the worst, and also spread over larger ranges. In contrast, the LSTM family has smaller ranges, and DI-LSTM-MLP performs the best, i.e., having the smallest ranges.

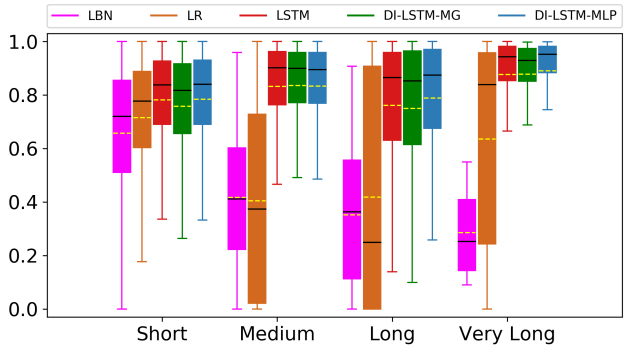


Figure 3: Box-plot of correlation for the histogram experiments, with median (solid line), mean (dashed line), and 10, 25, 75, 90 percentiles.

Table 4: Mean accuracy for Gaussian mixture representations - TG and SG denotes the speed and time Gaussians respectively. Furthermore, GRAN denotes the SG performance evaluated using 22.

Type	Method	All Paths			Short			Medium			Long			Very Long		
		COR	INT	KLD	COR	INT	KLD	COR	INT	KLD	COR	INT	KLD	COR	INT	KLD
TG	LSTM	0.891	0.786	0.37	0.562	0.459	2.885	0.930	0.796	0.083	0.954	0.865	0.086	**0.972**	**0.885**	0.054
	DI-LSTM-MLP	**0.929**	**0.829**	**0.118**	**0.597**	**0.462**	**1.019**	**0.969**	**0.854**	**0.020**	**0.979**	**0.896**	**0.032**	0.969	0.882	**0.047**
SG	LSTM	**0.645**	**0.544**	**0.669**	**0.633**	**0.568**	**0.682**	**0.635**	**0.555**	**0.663**	**0.640**	**0.522**	**0.701**	**0.687**	**0.534**	**0.625**
	DI-LSTM-MLP	0.565	0.480	0.875	0.563	0.516	0.883	0.560	0.495	0.867	0.556	0.453	0.902	0.594	0.458	0.857
GRAN	LSTM	**0.702**	**0.581**	**0.560**	**0.692**	**0.604**	**0.571**	**0.697**	**0.593**	**0.549**	**0.690**	**0.554**	**0.598**	**0.752**	**0.584**	**0.503**
	DI-LSTM-MLP	0.619	0.513	0.7583	0.618	0.548	0.766	0.617	0.528	0.750	0.604	0.484	0.789	0.653	0.494	0.726

Gaussian Mixture Model Experiments

In the previous set of experiments, we considered only the performance of histogram speed distributions. In this section, we explore the differences between travel speed and travel time distributions, as well as looking at the potential difference between Gaussian mixture speed distributions and histogram speed distributions. As reported in table 2, the DI-LSTM-MLP model achieved the best results, and as such, we will employ this model for the next set of experiments.

Results. As explained, the DI-LSTM-MLP is employed for these experiments; however, we also consider the LSTM model as the baseline. Furthermore, we also consider the same path categories. For these experiments, the Gaussian mixtures are predicted with eight mixtures. Eight mixtures are chosen as it gives a similar input size in the models, as the histogram distributions.

Table 4 reports the mean accuracy for the two models w.r.t both speed (SG) and time (TG) Gaussian mixtures. We see that the TG distributions achieve excellent results, both for the LSTM and DI-LSTM-MLP model. Specifically, we see that the DI-LSMT-MLP model continues to show the best results as determined in the previous set of experiments. However, looking at the SG distributions, we see that the LSTM achieves the best performance and that, in general, the speed distributions achieve significantly worse results.

As also observed with the previous experiments, we see that all models perform the worst on almost all metrics in the short path category, with the exception of the intersection metric for the SG data. Interestingly, we see that the TG data achieve significantly worse performance in the short paths compared to all other models and especially if compared to the results it produces for the other path categories. We expect this is due to the unlimited range of the time data, where observations are not locked to a specific range such as the speed data. This results in much more varied data, which is harder for the model to capture. Also, following previous results, we see that the models achieve better results in the longer path categories. This is, however, mostly the case for the TG data, as we see the SG data being more stable across all categories. This is illustrated in figure 4.

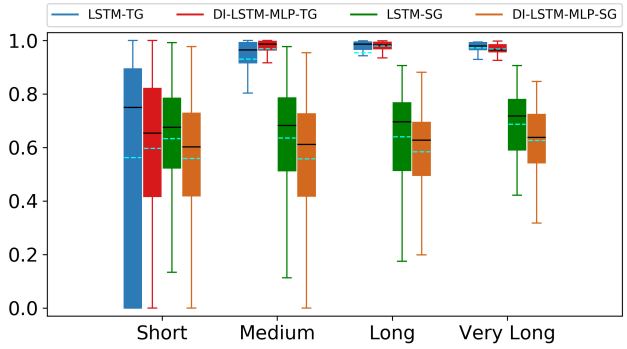
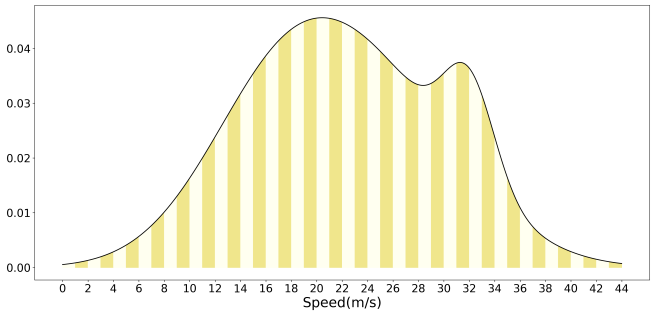


Figure 4: Box-plot of correlation for the Gaussian mixture experiments, with median (solid line), mean (dashed line), and 10, 25, 75, 90 percentiles.

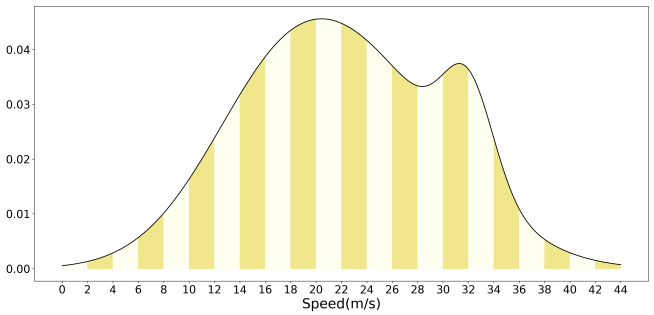
Initially, we expected the performance difference of the LSTM and DI-LSTM-MLP on the SG data, was due to possible error margins in the Gaussian mixtures generated by the MDN. However, considering the high accuracy of the TG data, we now believe the issues of the SG data performance are a result of under-fitting of the models. When considering the SG data, the length of each edge suddenly becomes much more important for the model, as the length of each edge has a much more significant impact on the overall speed of the path. Whereas with TG data, the raw travel time of the edge is sufficient for the model to arrive at the total travel time of the path.

In other words, we expect the two data structures to achieve much closer results if we can better fit and optimize the models, as, in theory, the speed and time should be interchangeable as long as we also have the length.

Comparing Speed Gaussians to Speed Histograms: Another interesting experiment, is that we can compare the results of the SG distribution to the results of using histograms. Since we consider the same evaluation metrics for both structures, one approach is to simply just compare the mean performance values. Here, we see that the histograms achieve significantly better accuracy. We expect again that this is attributed to the under-fitting of the models on the SG data.



(a) A Gaussian mixture integrated into 44 buckets.



(b) A Gaussian mixture integrated into 22 buckets.

Figure 5

Another approach to this problem is to evaluate the Gaussian mixture under the same conditions as the histograms, e.g., in 22 buckets. In figures 5a and 5b, we show how we can integrate a Gaussian mixture into a different number of buckets. The results of evaluating the Gaussians under the same conditions as histograms are reported in the last row of table 4. We observe that compared to the other SG results reported in the same table, the performance has increased significantly for both models. However, if we compare these new results to those in table 2, we see that the histogram models have much higher accuracy across all metrics and path categories. This is again attributed to the general bad performance of the SG models; however, we do see an increase in the performance. This indicates that if the Gaussian mixtures have lower accuracy than the histograms, the Gaussian mixtures could achieve similar or better accuracy under the same conditions.

5 FUTURE WORK AND OUTLOOK

One interesting avenue for future work and potential improvement for the model is the idea of considering more local features of edges in the road network. Currently, we consider the length of an edge as a local feature. However, we could also consider local features such as road type, speed limit, and road quality and see if such features could improve the performance of the model.

Another type of feature we could consider is the edge-to-edge relationship feature. We consider the edge-to-edge relation features to be features that describe the transition of one edge to another. Examples of this could be the angle between them or intersection type, where traffic lights could indicate that the next edge has a longer travel time compared to intersections, where there are no required stops. We expect that the DI-LSTM-MLP model would be able to better capture such edge-to-edge features, as we can consider two edges at a time in the MLPG. Thus allowing us to better define the importance of such features.

In terms of the Gaussian mixture experiments, we still have many things we wish to improve. Due to time limitations, we did not manage to optimize most of the models for the experiments we considered, and we did not have the time to complete more than a 3-fold cross-validation. As explained in the experiments section, we expect the poor performance of the speed Gaussians to be partly a result of too few epochs and, in general, a un-optimized model.

6 RELATED WORK

Most existing studies consider deterministic travel speed estimation and prediction. For example, DCRNN [4] integrates diffusion convolution into RNN to enable travel speed predictions for all edges, but not for paths. Wu et al. [8] introduces Graph WaveNet, a CNN framework that does not rely on a recurrent neural network to capture the temporal dependencies in the data. This method relies on stacked layers of dilated causal convolution to capture temporal sequences of longer range than traditional RNN-based approaches.

We categorize related studies into Table 5, where we consider estimating deterministic vs. stochastic speeds for edges vs. paths. In this paper, we focus on the setting of estimating stochastic speeds for paths, where the most relevant study [5] only works for paths with two edges. In contrast, we are able to estimate speed distributions for any path.

Table 5: Related Work.

	Deterministic Speeds	Stochastic Speeds
Edges	[4, 8]	[3]
Paths	[6, 7]	[5]

7 CONCLUSIONS

We present a novel dual-input LSTM (DI-LSTM) network for estimating travel cost distributions for paths. Experiments show that DI-LSTM has a great potential to achieve accurate estimation when considering histogram representations, outperforming several baseline models, including a standard LSTM model. This is especially true for longer paths. Furthermore, initial experiments with Gaussian mixture model

representations show a greater potential for accurate estimation. Results considering travel time Gaussian mixture models further shows the potential of the DI-LSTM model. However, the results of speed Gaussian mixture models show a worse accuracy. We expect this is a result of un-optimized models, which is more apparent in the results of speed Gaussian mixtures. This indicates a need for further research and experimentation.

REFERENCES

[1] Christopher M Bishop. 1994. Mixture density networks. (1994).

[2] Chenjuan Guo, Bin Yang, Jilin Hu, and Christian S. Jensen. 2018. Learning to Route with Sparse Trajectory Sets. In *ICDE*. 1073–1084.

[3] Jilin Hu, Chenjuan Guo, Bin Yang, and Christian S. Jensen. 2019. Stochastic Weight Completion for Road Networks Using Graph Convolutional Networks. In *ICDE*. 1274–1285.

[4] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2017. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. arXiv:cs.LG/1707.01926

[5] Simon Aagaard Pedersen, Bin Yang, and Christian S. Jensen. 2020. A Hybrid Learning Approach to Stochastic Routing. In *ICDE*. 2010–2013.

[6] Dong Wang, Junbo Zhang, Wei Cao, Jian Li, and Yu Zheng. 2018. When Will You Arrive? Estimating Travel Time Based on Deep Neural Networks. In *AAAI*.

[7] Zheng Wang, Kun Fu, and Jieping Ye. 2018. Learning to Estimate the Travel Time. In *SIGKDD*. 858–866.

[8] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph WaveNet for Deep Spatial-Temporal Graph Modeling. In *IJCAI*. 1907–1913.