Enhancing deep learning-based city-wide traffic prediction pipelines through complexity analysis

Nishant Kumar[1,2*], Henry Martin[2] and Martin Raubal[2]

[1*]Future Resilient Systems, Singapore-ETH Centre, Singapore, Singapore.
[2]Institute of Cartography and Geoinformation, ETH Zurich, Zurich, Switzerland.

*Corresponding author(s). E-mail(s): niskumar@ethz.ch;
Contributing authors: martinhe@ethz.ch; mraubal@ethz.ch;

Abstract

Deep learning models can effectively capture the non-linear spatiotemporal dynamics of city-wide traffic forecasting. Evidence of varying deep learning model performance between different cities, different prediction horizons, different scales, specific city regions, and during particular hours of the day abounds in the literature on deep learning-based traffic prediction, yet a unified metric to quantify the complexity of different prediction tasks does not exist. This paper proposes two metrics - Model Complexity ($\boldsymbol{MC}$) and Intrinsic Complexity ($\boldsymbol{IC}$). While $\boldsymbol{MC}$ quantifies the complexity of deep learning models for city-wide traffic prediction tasks, the $\boldsymbol{IC}$ quantifies the underlying complexity of the prediction task. Both metrics are validated through systematic experimentation using traffic volume data from three cities. Finally, we demonstrate how these metrics can improve the workflows for deep learning-based data-driven traffic prediction pipelines and deployment by reducing the hyperparameter search scope and comparing the effectiveness of different design pathways.

Keywords: deep learning, complexity, traffic prediction, city-wide

1 Introduction

Road network traffic is the net result of the spatiotemporal aggregations of actions of several decision-making agents or groups thereof - each operating for significantly

different motives (Kumar and Raubal, 2021). Moreover, urban traffic is a complex system (Khulbe et al, 2023) - the complexity attributed to the possibility of significantly different outcomes for the system based on individual choices (Allen, 2001). Urban traffic complexity exhibits several spatio-temporal variations. Some of these variations in complexity are intuitive and well-established. For instance, freely flowing traffic is less complex than congested state traffic, which occurs when traffic density is higher than the critical density (Spiliopoulou et al, 2014; Loder et al, 2019; Manibardo et al, 2021). Similarly, traffic on a freeway is known to be significantly easier to model and predict compared to that on urban roads, where the presence of intersections, heterogeneous streams of traffic, and interactions with non-vehicular traffic often result in increased conflicts, and consequently, higher prediction errors (Li et al, 2023; Kumar and Raubal, 2021; Zhang et al, 2021).

As we move away from the intuitive differences in traffic complexity, the variation in the complexity becomes highly nuanced, and quantifying it necessitates considering a wide range of scenarios and mechanisms of interactions, often rendering the challenge to quantify complexity mathematically and computationally intractable (Rand, 2015; Bettencourt, 2015). For instance, an accident can be caused by a human error on the part of a single driver; nevertheless, it can potentially disrupt the freely flowing traffic for all drivers present in upstream traffic of the road segment (Chakraborty et al, 2023). Complexity can be viewed as antithetical to predictability- the more complex a traffic state, the more likely the prediction errors are (Bialek et al, 2001; Boffetta et al, 2002). Traffic complexity can be quantified using a Lagrangian view of traffic by tracking individual agents or an Eulerian view by focussing our attention on the complexity of the aggregated traffic dynamics. The Lagrangian approach becomes computationally challenging as the number of agents increases, and its realisability is limited by the partial observability of the system Batty and Torrens (2005); Shiftan and Ben-Akiva (2011). The Eulerian view of complexity necessitates modelling the aggregated behaviour of all agents and then investigating modelling errors to quantify the underlying complexity of traffic. This paper adopts an Eulerian view, uses deep learning (DL) models to model the traffic under a wide range of scenarios, and investigates the relationship between model complexity and prediction errors.

For data-driven traffic prediction, the relevance of deep learning (DL) models for traffic prediction has increased significantly in the past few years. This increase is mainly due to access to larger datasets since, given sufficient data points, DL models can outperform classical machine learning algorithms (Roßbach, 2018). Several novel DL architectures, increasingly more complex and computationally heavier, have been proposed to model spatiotemporal dynamics of urban traffic (Kumar and Raubal, 2021; Kashyap et al, 2022). Typically, the workflow for developing novel DL architectures for traffic prediction comprises evaluating the model's prediction errors on a held-out subset of data (validation data) to gauge the model's generalisation ability. However, two DL models that exhibit similar prediction errors on the validation set might perform significantly differently when deployed in real-life scenarios and subjected to future test data. This performance drop is often attributed to concept drift - which implies the mapping between the input and output data changes over time (Xiao et al, 2019). To ensure that the DL models retain low generalisation errors

on previously unseen data, the machine learning community has advocated using models that are *just enough* complex (Hu et al, 2021; Bejani and Ghatee, 2021; Hu et al, 2020). The need to research such just-enough-complexity DL models for traffic prediction was highlighted in a recent survey paper, which critically examined the trend to develop ever-larger DL models for traffic prediction, often at the cost of limited scalability, extreme computational requirements, and reduced explainability (Fafoutellis and Vlahogianni, 2023).

Choosing just enough complex models necessitates formulating a model complexity metric and an intrinsic complexity metric for the prediction task itself so that the two can be compared. Moreover, these metrics should be (computationally) cheap to evaluate to ensure practical use in workflows. This paper contributes to both aspects of this challenge by proposing a model complexity (MC) metric for DL models for city-wide traffic prediction tasks and a metric to compute the intrinsic complexity of (IC) such tasks. Both metrics induce only a small overhead relative to the overall training time of a DL model, thereby enabling practical use by the community. For clarification, a city-wide traffic prediction refers to simultaneously predicting traffic at multiple locations throughout the city.

The metrics presented here are targeted at improving the workflows for research into DL-based traffic prediction. Examples of leveraging model complexity for better generalisation abound in the machine learning community - the most relatable being better generalisation through efficient dataset utilisation by using an ensemble of less and more complex models on different subsets of data points, each subset characterised by different difficulty of learning (Maini et al, 2022). Historically, the commonly used techniques to improve model generalisability, such as dropout (Srivastava et al, 2014) and pruning (Han et al, 2015), were inspired by reducing model complexity (Park et al, 2020). Thus, a model complexity for traffic prediction tasks furthers the research into better generalisable DL models for traffic prediction. Similarly, an a priori estimate of the intrinsic task (or sub-task) complexity provides valuable information for researching novel traffic prediction architectures. For instance, while choosing suitable models and hyperparameters, researchers often try a grid search of hyperparameters and model architectures to guide their choices, thus making the modelling process time-consuming. Suppose the complexity of the task is known beforehand. In that case, significant improvements in the modelling pipeline can be made by limiting the search scope to architectures that perform well on problems with a similar complexity rating. Moreover, better quality guarantees for the model prediction can be provided if the generalisation error has been estimated before training the DL model (Jin et al, 2020). While all of the well-known uses of a model complexity metric directly apply to our proposed metric, we have included some demonstration experiments to show selected use cases of our metrics for DL-based traffic prediction tasks.

The rest of the paper is organised as follows. In section 2, we discuss the previous work where we summarise the key empirical observations from the transportation domain and the various existing approaches to DL model complexity. In section 3, we present our methodology, which comprises discussing the concept of Critical Sample Ratio (CSR) and the formulation of our metrics, the Model Complexity (MC), and the Intrinsic complexity (IC) of a traffic prediction tasks. We present the data and

pre-processing steps in the section 4. The section 5 summarises the experimental validation of our metrics and hypotheses. In section 6, we demonstrate selected use cases of our proposed metrics. Finally, in section 7, we conclude the paper and discuss the avenues for future research in this direction.

2 Related Work

In the remainder of this section, we discuss what complexity means for data-driven traffic prediction tasks, followed by empirical evidence from three key dimensions (prediction horizon, number of time steps predicted, and the spatial granularity of prediction), along which the complexities of data-driven traffic prediction are interesting questions for different stakeholders of the transportation system.

2.1 Evidence from DL-based traffic prediction literature

Several recent surveys and benchmarking papers show that the performance of a DL model decreases as the length of the prediction horizon or the number of time steps predicted increases (Kumar and Raubal, 2021; Li et al, 2021), thereby implying that the traffic prediction complexity increases with the prediction horizon. Similarly, the morning peak hour traffic is significantly more complex to model than off-peak hour traffic, possibly due to a relatively larger number of decision-making agents on the road during peak hours (Yu et al, 2017b).

This section summarises the empirical evidence on the differences in DL model performance when predicting traffic at different prediction horizons, different numbers of time steps predicted and the spatial scale of prediction. The prediction horizon refers to the number of time steps between the input and the output. While the prediction horizon distinguishes tasks such as predicting traffic *at* 30 minutes from now and that *at* 1 hour from now, the number of time steps predicted distinguishes tasks such as predicting traffic *for* the next 30 minutes from now and *for* the next 1 hour from now (Fig. 1 shows a detailed implementation-level differentiation of prediction horizon and the number of time steps). Finally, the spatial scale of prediction distinguishes tasks such as predicting city-wide traffic by dividing the city into 25×25 urban tiles and when dividing into 100×100 urban tiles.

Prediction horizon: DL models have been extensively explored for short prediction horizons, ranging from a few minutes to hours. Predicting traffic at varying lengths of horizons into the future has varying implications for traffic planning. For instance, a prediction horizon of a few minutes provides valuable information for real-time traffic management. In contrast, a longer horizon of an hour helps make informed decisions regarding traffic signal timings, re-routing of traffic, and effectively managing incidents.

A significant fraction of recent research on DL-based traffic prediction reports the model performance degradation of DL models with increasing prediction horizon, thereby hinting at the increased underlying complexity of the modelling traffic prediction tasks at longer prediction horizons (Jia et al, 2016; Wu et al, 2018; Li et al, 2023; Kumar and Raubal, 2021; Khajeh Hosseini and Talebpour, 2023). This performance degradation has been observed for a wide range of DL architectures.

As reported in Zheng et al (2022), between a prediction horizon of 5 minutes and 1 hour, while predicting speed using METR-LA dataset, the performance drop measured using Mean Absolute Percentage Error (MAPE) for a Gated-Recurrent Unit (GRU) model (Chung et al, 2014) 7.87% vs. 12.86%, for a CNN-LSTM model (Yu et al, 2017a) is 7.88% vs 12.67%, and that using Temporal Graph Convolution Networks (T-GCN) (Zhao et al, 2019) is 9.77% vs. 13.02%. Very recently, increasingly powerful DL models comprising attention graph convolutional blocks have been proposed, which have demonstrated a significant reduction in the performance drop (2.63 to 2.73 when using AGCN-Seq2Seq-Attn (Zhang et al, 2019).

Number of time steps predicted: Using traffic volume data collected from an expressway near Minneapolis-Saint Paul, Yin and Dai (2022) compared several DL architectures for multi-step prediction of traffic volume and concluded that as the number of steps in a multi-step-prediction task increases, so does the complexity of the task as evidenced by higher prediction errors, with each DL architecture being affected, albeit to varying degrees. When considering 3-time-step and 9-time-step prediction and using iterative prediction, the Mean Absolute Error (MAE) for an LSTM model increased from 364 to 569, and that for an Encoder-Decoder architecture dropped from 361 to 564. Their self-proposed DualMNet architecture (comprising spatial and temporal attention units), which is capable of predicting multiple time steps, reduced this performance drop (MAE varied from 341 to 549). Apart from traffic volume, they used two other spatiotemporal datasets, air quality (Beijing 2.5) and indoor-environment (SML, 2010) datasets from the UC-ML repository (Asuncion and Newman, 2007) and reported performance drop for both datasets when shifting from predicting multi-step to iterative forecasting.

Spatial scale of prediction: During our literature review, we found only a single paper which has extensively compared the model performance at different scales, referred to as *coarser* and *finer*. This sparsity of literature may be attributed to the significant computational challenges that arise when handling different dimensions of input and output tensors for different scales. In Wang et al (2022a), 14 sensors were grouped into five groups based on spatial proximity, and this scale is referred to as *coarser* scale; when sensors are considered separately, it is referred to as the *finer* scale. From Figure 1 in their paper, we inferred the sensors in each group to be approximately within squares of 1 sq. km and the minimum distance between sensors to be 200 metres. They proposed a novel architecture called MC-STGCN composed of Graph-Convolutional units) that aggregates information from both scales in the hidden layers and predicts traffic flow at various sensor locations. While comparing their architecture with existing architectures, they report that most models (such as LSTM(Hochreiter and Schmidhuber, 1996), T-GCN (Zhao et al, 2019), and DCRNN (Li et al, 2017)), including their proposed model, experience a significant drop in performance when shifting from coarse to fine scales. Specifically, for the finer scale, the validation error measured using Root Mean Squared Error (RMSE) was more than double that of the coarser scale.

Summary: Based on the empirical evidence from the literature, the following three trends in complexity are clear:

1. Longer prediction horizons result in increased complexity of traffic prediction.

2. A higher number of time steps for prediction results in increased complexity of traffic prediction.
3. Predicting traffic at a finer spatial scale is more complex than predicting at a coarser scale. If the city is divided into tiles, a finer spatial scale implies a smaller size of tiles.

2.2 Deep learning complexity

The meaning of the model complexity of DL models differs significantly from traditional machine learning models. The set of trainable parameters, which is often leveraged to derive metrics for model complexity in classical machine learning, is not correlated with the model complexity of DL models (Zhou, 2021). This contrasts the established Bias-Variance tradeoffs for classical machine learning (Belkin et al, 2019). Classical metrics for model complexity (such as the VC-dimension (Vapnik et al, 1994) and Rademacher complexity (Sammut and Webb, 2010)) have limited practical importance for DL models. This limited practical use can be attributed to the over-parametrization of DL models, implying a significantly large number of parameters, often much higher than the number of data points that a DL model is trained on. The over-parametrization results in the true realised complexity (effective complexity) varying significantly from the theoretical estimates of realisable complexity obtained using model parameters (expressive complexity). The effective complexity depends on both- the model and the data and, as expected, is realised only *after* the model has been trained (Hu et al, 2021).

Effective complexity refers to the complexity of the decision boundary of the trained model. A smoother decision boundary implies a less complex model (Hu et al, 2021). Two main approaches are found in the literature on effective complexity for DL models. The first approach, conceptualised in Montufar et al (2014), relies on approximating the decision boundary with piece-wise linear approximations and counting the number of such linear approximations needed. This approach has been advanced by Raghu et al (2017), who count the number of activations of an activation function (thresholded ReLU), and by Gamba et al (2022), who measure the deviations of a line segment between two input data tensors, after its transformation by the DL model. The second approach is based on quantifying the sensitivity of the model output to small perturbations in the input data tensors. For a classification task, the sensitivity can be understood as analogous to the distance of a data point from the classification boundary. In Hahn et al (2021), post-training model sensitivity was proposed as a complexity measure for the prediction task. The measure was used to determine which data points were relatively more difficult to learn. Novak et al (2018) conducted an empirical study using FFNNs and proposed the Jacobian norm of the input-output matrix of the trained model as a measure of the complexity of the decision boundary near an input data tensor. They concluded that a high negative correlation exists between the DL model's sensitivity and its generalisation to the unseen test data. Alternatively, a DL model, which generalises better, has data points further away from the decision boundary than a less generalisable DL model (cf. Johnson and Khoshgoftaar (2019) for a comprehensive survey on the sensitivity of DL models).

The second approach based on sensitivity was used in Arpit et al (2017), and the Critical Sample Ratio (CSR) was proposed as a metric to measure DL model complexity. Their findings have found applications in several domains and resulted in more robust models; *e.g.*, in Wang et al (2022b), for prediction in the presence of noisy labels in an NLP setting, in Papangelou et al (2019), for determining adversarial patient samples in clinical trials, and in Borg et al (2022), for detecting obscenity in videos). This paper introduces a new model complexity metric (MC), expanding on CSR for multivariate time series regression, tailored for computationally efficient analysis of traffic data.

3 Methodology

The methodology comprises the formulation of our two metrics - Model Complexity (MC), which is defined for a DL model and Intrinsic Complexity (IC) which is defined for the task itself. The key notations are summarised before discussing the background of the CSR and our proposed metrics (MC and IC).

3.1 Summary of key notations

For a multi-step prediction, the representation of the number of time steps predicted and the number of time steps after which prediction starts varies across different papers. Hence, to avoid any ambiguity, we summarise the key notations used in this paper, including a clear definition of a task and the dimensions of input and output tensors for various tasks:

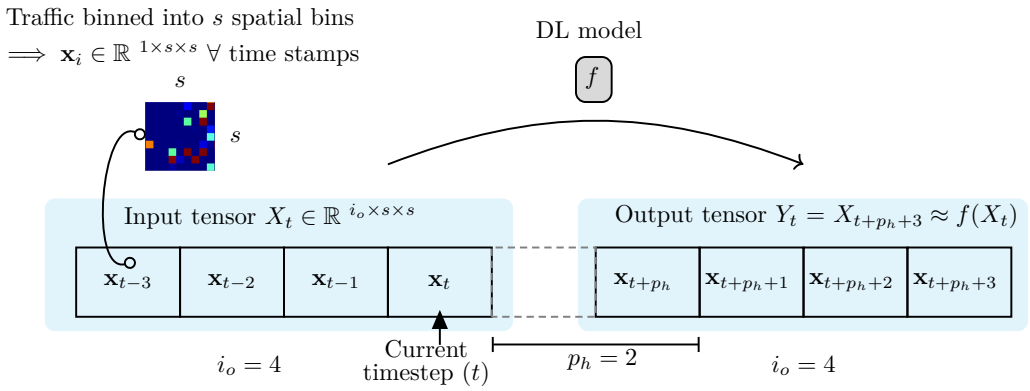


Fig. 1: Illustration showing how the heatmaps of traffic data at several time steps $\mathbf{x_{t_k}}$ are concatenated to create the input and output tensors. The DL model f learns to approximate Y_t with $f(X_t)$. The number of time steps in input and output i_o is 4 and the prediction horizon (p_h) is 2

When predicting a macroscopic traffic variable (e.g., speed, volume, or density) at several time horizons simultaneously at multiple points in the city, each input data

tensor X is a 3-D tensor. The tensors are represented using capitalised letters following the recommendations in Chiang et al (2021). We define a city-wide traffic prediction task ($Task(i_0, p_h, s, city)$) with the parameters explained below:

- **Input-Output length** (i_o): refers to the number of time steps in input tensor and the number of time steps in the output tensor. The experiments in this paper only consider tasks where the number of time steps used as input and output are the same. For instance, tasks such as 'predict next 10 time steps using last 5 time steps' are not considered.
- **Prediction Horizon** (p_h): refers to the number of time steps from the current time (t) when the prediction begins; hence the timestep $t+p_h$ represents the start of the prediction period, while $t + p_h + i_o - 1$ marks the end of the prediction period, where i_o is the total number of steps predicted.
- **Scale** (s): refers to the spatial scale of prediction - implying that the traffic data (speed, density, or volume) in the *city* is binned into $s \times s$ (urban) pixels to create the dataset. A lower value of s implies coarse spatial scale and vice-versa.
- **City**: refers to the city name.

Our nomenclature allows us to capture a wide range of tasks such as single-step traffic prediction ($i_0 = 1$), multi-step traffic prediction ($i_0 > 1$), and short-term prediction versus long-term prediction (by varying p_h). Moreover, given a task, each tensor X that has a timestamp (say t), and if we consider this task to be modelled by a DL model f, the prediction and the ground truth can be written as $f(X_t)$ and (X_{t+p_h}) respectively. For simplified notation, depending on the context, when our discussion requires comparing tasks where only one of the parameters varies and others remain unchanged, we omit the other parameters to reduce clutter.

All input and output tensors for $Task(i_0, p_h, s, city)$ have the dimensions $(i_o \times s \times s)$. A simple representation of the tensor dimensions and the meaning of different parameters of a task are shown in Fig. 1 for an example task, $Task(i_0 = 4, p_h = 2,\ s,\ city)$.

A single raw traffic dataset $\mathcal{D}_{\text{raw}}$ can result in a wide range of tasks - with each task needing a preprocessed dataset $\mathcal{D}$. The task-specific dataset $\mathcal{D}$ comprises the training and the validation data tensors and are denoted by $\mathcal{D}_{train}$ and $\mathcal{D}_{val}$ respectively. Additionally, $\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{val}$. There is no mixing of datasets across tasks; hence, we have omitted the subscript for task $\mathcal{D}$ as it is always clear from the text which dataset is being referred to.

Finally, the validation loss is represented by *Val MSE* and refers to the validation mean squared error defined by:

$$MSE = \frac{1}{|\mathcal{D}_{val}|} \sum_{X,Y \in \mathcal{D}_{val}} \|f(X) - Y\|_2 \tag{1}$$

3.2 Background: Critical Sample Ratio

Critical Sample Ratio (CSR) was defined for classification tasks as the fraction of data points that are assigned labels that differ from their neighbour's labels. Formally, the effective complexity of a DL model is given by the ratio $\frac{\#\text{CS}}{\#\text{data points}}$, where a data tensor X is a Critical Sample (CS) point if:

$$f(X) \neq f(\hat{X}) \qquad \text{for any } \hat{X} \quad \text{s.t.} \quad \|X - \hat{X}\|_\infty \leq r_X \tag{2}$$

where $f(X)$ is a DL model that takes an input X and outputs a class label $Y = f(X)$ and r_X is the radius that defines the neighbourhood of data tensor X.

For the single-sensor, single-horizon output ($p_h = 1$), the CSR can be trivially extended by quantising the traffic variable into discrete bins. However, the extension is not straightforward for multivariate (more than one sensor) and multiple time steps ($i_o > 1$) cases. First, the extension involves defining thresholds in the output space to define equality. Second, an efficient neighbour-search strategy is warranted in light of huge tensor sizes. Since traffic data exhibits strong daily and weekly periodicity, this paper defines the neighbourhood of tensors in terms of thresholds in the temporal dimension and thus accomplishes an efficient neighbour-search strategy. The key methodological details are summarised in the next two subsections.

Suppose we set $r_X = 0$ in Equation 2, no data point is a critical sample, whereas if $r_X = \infty$, every data point becomes a critical sample. Arpit et al (2017) used a fixed value for r_X (0.3) for the image dataset (pixel-values in range 0-255) and found that r did not affect the qualitative nature of their results. However, due to the significant variations in the dimensionality of input tensors across our tasks, the expected distances between random pairs of data tensors can change significantly across tasks, as shown in Fig. 2). Hence, a suitable task-specific threshold for r_X must be determined. Given the periodicity of traffic data, the nearest tensors from X are highly likely to be found within small temporal *bands* of daily periodicities. Thus, if we define r_X as the maximum of L_∞ distances across all neighbours of X that have a timestamp within a temporal *band*, r_X can vary across tasks, yet for each task, defines a neighbourhood that, with a high probability, comprises a tensor-set that is nearest to X amongst all the tensors in the dataset. We detail the parametrisation of r_X using the temporal *band* below and introduce *sample complexity* for regression tasks, which is analogous to the *critical samples* for classification tasks.

3.3 Sample Complexity and Model Complexity

First, we quantify how complex the decision boundary is near a data tensor. Thereafter, the sample complexity is aggregated across all data tensors to quantify the DL model's complexity. The sample complexity of a data tensor can be captured by quantifying how much the DL model has transformed the input space in the neighbourhood of the tensor in question. As shown in Fig. 2, the neighbours to a data tensor X, based on the L_∞ norm, are likely to be found within short intervals around multiples of 24 hours.

Let us define the maximum L_∞ distance from X to its neighbours within these intervals as r_X. Let us assume that the traffic data is collected every p minutes, and each data tensor has a timestamp t. Then, using a look forward and backward policy of h hours for d days, the neighbours of X are the data tensors whose timestamps lie within the set $\mathbb{U}_t$, given by:

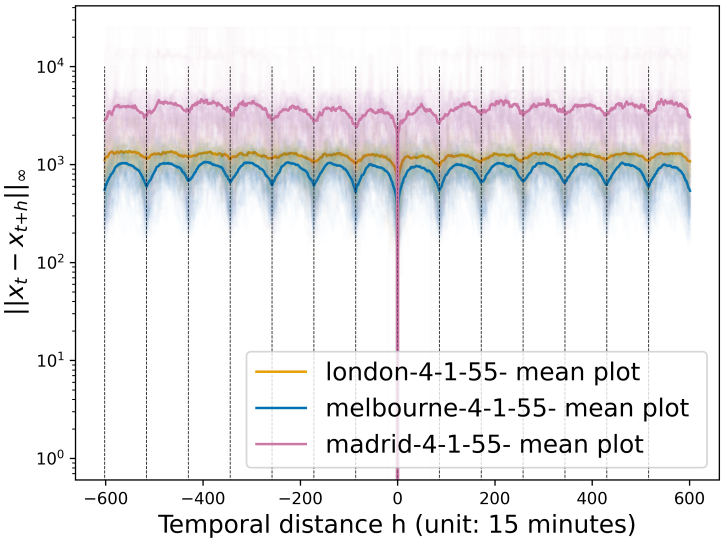


Fig. 2: Solid plot: Mean plot of L_∞ distances of 100 randomly chosen source data tensors x from their temporal neighbours (± 7 days). The light plots are individual plots that constitute the mean plot. The dashed vertical lines show markers for a temporal distance of 24 hours from the timestamp of X. The nearest neighbours of X are highly likely to be present near the dashed vertical lines. The label london-4-1-55 implies $Task(i_o = 4, p_h = 1, s = 55, city = London)$

$$\mathbb{U}_t = \left\{ t' \mid t' \neq t,\ t' = \ t \ \pm \underbrace{n_d \cdot 24 \cdot \frac{60}{p}}_{\text{daily periodicity offset}} \pm \underbrace{n_h}_{h \text{ hours look forward \& backward}} \right\} \tag{3}$$

where

$$n_d \in \{0, 1, 2, ..., d\} \ \text{ and } n_h \in \{0, 1, 2, ..., h \cdot \frac{60}{p}\} \tag{4}$$

The corresponding set of the temporal neighbours of any tensor X is given by:

$$\mathbb{T}_X = \{X_t \mid t \in \mathbb{U}_t\};\ |\mathbb{T}_X| = |\mathbb{U}_t| = (2 \cdot d + 1) \cdot (2 \cdot h \cdot \frac{60}{p} + 1) - 1 \tag{5}$$

Finally,

$$r_X = \max(\{||X_i - X||_\infty\}\ |X_i \in \mathbb{T}_X) \tag{6}$$

Any statistics that measure the degree of disagreement between the sets $\mathbb{T}_X$ and $\mathbb{T}_Y$ can capture the complexity of the decision boundary of f in the vicinity of data tensor X. We are interested in capturing the extent to which a DL model f makes

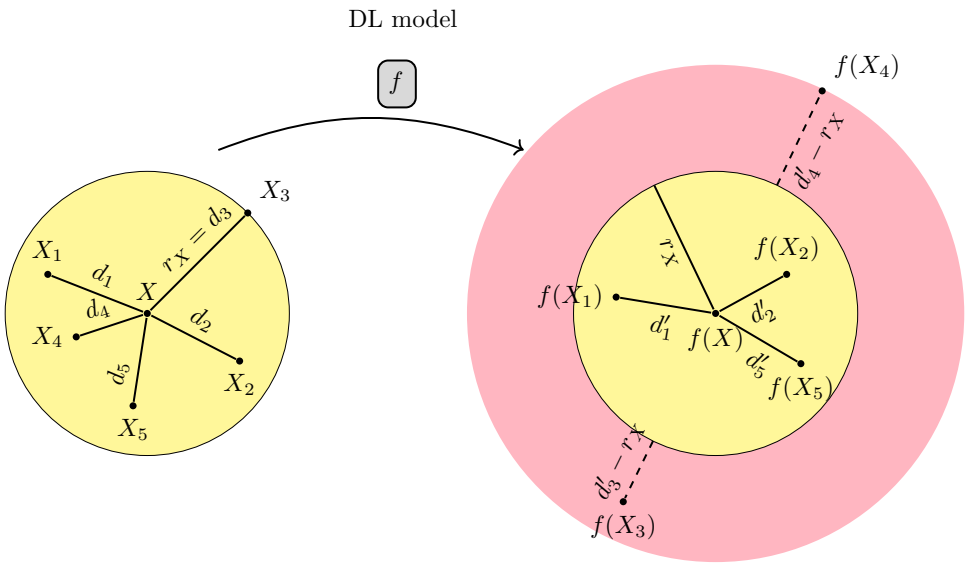


Fig. 3: An abstract representation of a DL model f showing how it transforms the neighbourhood of tensor X and results in some neighbours (X_4 and X_3) being mapped outside the circle in the input space. Here, the sample complexity of X, given by $SC(X|f)$ is $d'_3 + d'_4 - 2 \cdot r_X$

predictions for nearby data tensors that are significantly different from the prediction for the data tensor X. We propose to measure the degree of this mismatch using sample complexity (SC) of a data tensor X given a model f, defined as:

$$SC(X|f, \mathcal{D}) = \sum_{X_j \in \mathbb{T}_X} \left(d_{f(X_j)} - r_X\right) \cdot 1_{d_{f(X_j)} > r_X} \tag{7}$$

where $d_{f(X_j)} = \|f(X_j) - f(X)\|_\infty$.

Finally, the effective model complexity of f is defined as the mean of sample complexity across all training data tensors. We refer to the effective model complexity as Model Complexity(f), shortened to MC(f) and defined by:

$$MC(f|\mathcal{D}) = \frac{1}{N} \sum_{k=1}^{N} SC(X_k|f, \mathcal{D}) \tag{8}$$

where $N = |\mathcal{D}|$ is the total number of data tensors in the data set $|\mathcal{D}|$, which is composed of the training and validation data, $\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{val}$. For illustration, the sum of protrusions into the red region in Fig. 3 visualises the SC values for the neighbourhood of the data tensor X. When the DL model (f) and the dataset being referred to $\mathcal{D}$ are clear from the context, we omit the subscripts f and $\mathcal{D}$ from the representation of MC in order to avoid clutter.

In theory, N should be computed using all data points; however, in practice, smaller values of $n(< N)$, implying a randomly chosen subset of $\mathcal{D}$ can be used to estimate MC provided the mean of SC is found to converge when using $n < N$ (cf. Appendix A section 7 for a detailed analysis of the computational overhead of evaluating MC).

The metric MC quantifies the degree to which a DL model transforms the input space. Let us consider a DL model that performs just as good as the persistence forecast ($f(X) = X$; also referred to as the 'recent forecast' or the 'naïve model' in some literature (Aldwyish et al, 2021). It does not transform the input space at all and represents the most fundamental level of complexity. For such a model, Equation 7 yields a sample complexity of zero for all data tensors. Consequently, the model complexity of the naïve model, as per Equation 7 is zero.

3.4 Intrinsic Complexity

Let us consider that a perfect model (f_{PM}) exists that achieves zero error on validation data, such that $f_{PM}(X_i) = Y_i$; $\forall X_i \in \mathcal{D}$, $\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{val}$. All our modelling attempts for this task target approximating the f_{PM} as closely as possible using only the training data. In some machine learning literature, the f_{PM} is also referred to as the oracle (Thompson et al, 2020). If f_{PM} exists, it implies that we can compute the intrinsic sample complexity (ISC) of any data tensor X by replacing the prediction ($f(X_j)$ in Equation 7) with $f_{PM}(X_j)$, which is same as the known ground truth value Y_j, as shown below:

$$ISC(X|\mathcal{D}) = \sum_{X_j \in \mathbb{T}x} \left(d_{Y_j} - r_X\right) \cdot 1_{d_{Y_j} > r_X} \tag{9}$$

where $d_{Y_j} = \|Y_j - Y\|_\infty$ and $Y_k = f_{PM}(X_k)\ \forall\ X_k, Y_k \in \mathcal{D}$.

Similarly, if we measure the complexity of f_{PM} using Equation 8, we can capture the intrinsic complexity of the underlying traffic prediction task. Formally,

$$\text{Intrinsic Complexity (Task)} = MC(f_{PM}|\mathcal{D}) \tag{10}$$

$$IC(Task) = \frac{1}{N} \sum_{k=1}^{N} ISC(X_k|D) \tag{11}$$

Both ISC and IC can be computed before training a DL model since they do not depend on the model. This concludes our formulation of a task-specific (IC) and a data-point-specific (ISC) complexity measure for traffic prediction tasks. SC and MC, as formulated in Equation 8 and Equation 7 respectively, depend on the DL model f and on the $Task$ (implicitly through f). However, unlike SC and MC, the IC and the ISC are invariant to the model f being used and are an intrinsic property of the prediction task defined by the available traffic data. To ensure that ISC and IC have practical usefulness, these should be computed only using $\mathcal{D}_{train}$. Additionally, in theory, N should be computed using all data points, but in practice, smaller values of $n(< N)$, implying a randomly chosen subset of $\mathcal{D}$ can be used to estimate IC provided the mean of ISC is found to converge when using $n < N$.

3.4.1 Intrinsic Complexity of temporal bins

The Intrinsic Sample Complexity (ISC) defined in Equation 9, can be aggregated in different ways to compute the complexity of the different spatial and temporal bins of the data.

Let $\mathcal{D}^{t+\delta t} \subseteq \mathcal{D}$ such that for any $X_i \in \mathcal{D}^{t+\delta t}$, implying that the timestamp of X_i lies between t and $t + \delta t$. Thus, the intrinsic complexity of this temporal bin ($IC_{Temp}(t + \delta t)$) is:

$$IC_{Temp}(t + \delta t \,|\, Task) = \sum_{X_i \in \mathcal{D}^{t+\delta t}} ISC(X_i|\mathcal{D}) \tag{12}$$

3.4.2 Intrinsic Complexity of spatial bins

The complexity of each spatial bin can be computed by decomposing the Task into sub-tasks, with each sub-task being to predict the traffic on a specific pixel or a patch (which is a group of nearby pixels).

We can divide the $Task$ into sub-tasks of predicting y at each pixel for pixel-level complexity. If the subtask for any pixel at i, j is $Task_{pixel}(i, j)$, the IC of the pixel is given by:

$$IC_{Pixel}(i, j \,|\, Task_{Pixel}(i, j)) = \sum_{k=1}^{N} ISC(x'|\mathcal{D}^{Pixel}) \tag{13}$$

where $\mathcal{D}^{Pixel}$ is a new dataset as a result of transforming each element x in $\mathcal{D}$ to x' in $\mathcal{D}^{Pixel}$ using the operation $x' = x \odot M_{i,j}$, where $M_{i,j} \in \mathbb{R}^{i_o \times s \times s}$ is a 3D mask with $M_{i,j}(r, p, q)$ being a single voxel of $M_{i,j}$ such that

$$M_{i,j}(r, p, q) = \begin{cases} 1 & \text{if } p = i \text{ and } q = j \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

and $r \in \{1, 2, .., i_o\}$, and $p, q \in \{1, 2, .., s\}$

Similarly, the complexity of a spatial patch of size $p_x \times q_y$ with its left-upper corner at pixel location i, j in the city can be computed by defining a subtask $Task_{Patch}(i, j, p_x, q_y)$ and the IC for the patch is given by:

$$IC_{Patch}(i, j, p_x, q_y \,|\, Task_{Patch}(i, j, p_x, q_y)) = \sum_{i=1}^{N} ISC(x'|\mathcal{D}^{Patch}) \tag{15}$$

where $\mathcal{D}^{Patch}$ is a new dataset as a result of transforming each element x in $\mathcal{D}$ to x'' in $\mathcal{D}^{Patch}$ using the operation $x'' = x \odot M_{i,j,p_x,q_y}$ where $M_{i,j,p_x,q_y} \in \mathbb{R}^{i_o \times s \times s}$ is a 3D mask with $M_{i,j,p_x,q_y}(r, p, q)$ being a single voxel of M_{i,j,p_x,q_y} such that

$$M_{i,j,p_x,p_y}(r, p, q) = \begin{cases} 1 & \text{if } i \leq p \leq i + p_x \textit{ and} \\ & j \leq q \leq j + q_y \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

and $r \in \{1, 2, .., i_o\}$, and $p, q \in \{1, 2, .., s\}$

3.5 Relationship between MC, IC, and Validation-errors

Training a DL model f for traffic prediction to match the perfect model f_{PM} with zero validation error implies that $MC(f)$ equals the intrinsic complexity $IC(Task)$. However, traffic's unpredictable nature and data noise make this ideal unattainable, highlighting the untapped potential indicated by the difference $|IC - MC|$. We hypothesize a strong correlation between this difference and validation errors when considering different tasks with varying complexities. Moreover, suppose we consider multiple tasks with significantly different complexities and freeze the DL model architecture and hyperparameters. In this case, the model's capacity to achieve significantly different model complexities is restricted. Hence, a much smaller variation in the MC values is expected compared to the variation in IC across various tasks. Consequently, the unrealised model capacity term ($|IC - MC|$) will be dominated by IC. Therefore, given a fixed model architecture and hyperparameters, we hypothesise a high correlation exists between the validation errors and the intrinsic complexities IC across a range of tasks of varying complexities.

We summarize our hypotheses as follows, with empirical validation discussed later:
1. For a single task:

$$\text{Validation error} \propto |IC - MC|$$

2. Across tasks with varying complexities, using the same model architecture and hyperparameters:

$$\text{Validation error} \propto IC$$

4 Data, Pre-processing steps, and Models

This section presents the details regarding the datasets, the main preprocessing steps, and the basic architecture of DL models used across our experiments. Experiment-specific aspects (*e.g.,* learning rate and exact model architecture) are discussed along with each experiment.

4.1 Raw data

The traffic flow data used for our experiments was released by The Institute of Advanced Research in Artificial Intelligence (IARAI), which organised the Traffic4Cast 2022 competition (Neun et al, 2022). This data was collected using vehicle loop detectors (VLDs) across three cities and was provided for seven continuous months, with an average value of traffic volume every 15 minutes. Each city has data from approximately 4000 VLDs, and approximately seven months of data is available. Each city's seven months of traffic data are split roughly in the middle (3.5 months) to create train and validation sets, as detailed in Table 1. The data set can be downloaded by following the instructions at the official repository for the competition [1].

[1] https://github.com/iarai/NeurIPS2022-traffic4cast

4.2 Preprocessing

The traffic data (traffic volume for our case) is converted to a heatmap using a commonly used spatial binning technique for data-driven city-wide traffic prediction - specifically, as detailed in Fafoutellis and Vlahogianni (2023, p.23). The minimum and maximum latitude-longitude values across all the VLDs in the city are computed and used to define the spatial extent of the city under consideration. The city is then divided into $s \times s$ urban tiles (pixels). Each VLD v^k is then assigned to a pixel location (i, j) based on the pixel into which the coordinates of v^k lie. For any time step t, the input and output tensors are computed by averaging over all VLDs that are mapped to the same spatial bin. For a task defined by $Task(i_o, p_h, s, city)$, the dimensions of input and output 3D tensors are given by: $X, Y \in \mathcal{D} \subset \mathbb{R}^{i_o \times s \times s}$. A heatmap visualising the traffic volume data for Madrid is shown in Fig. 4.

Depending on the distribution of VLDs, the dimensions of the area of study for each city vary, and consequently, the physical meaning of scale also varies according to the city. For instance, for London at scale 25, the area of each pixel is 1.97×1.20 sq. km, whereas for Madrid, where the VLDs are concentrated in a smaller area, the area of each pixel at scale 25 is 0.82×0.87 sq. km. (cf. Appendix C Table 9 for a detailed summary of pixel sizes at different scales for each city and Appendix Fig. 16 for the spatial distribution of VLDs with respect to the road network for each city).

The dimensions of the input and output tensors, and hence, the tensors in the processed dataset $\mathcal{D}$, are determined by the task.

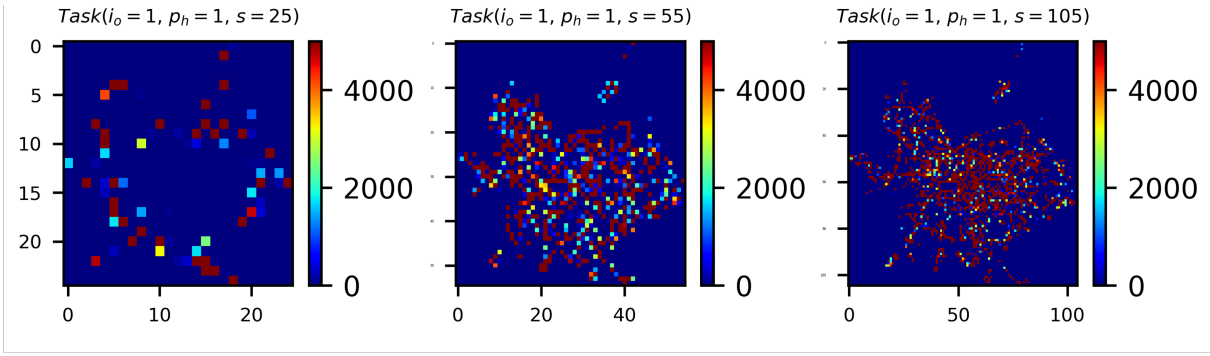


Fig. 4: Aggregated values of traffic volume after binning the VLDs into pixels at different scales for Madrid; scale $s \in \{25, 55, 105\}$

4.3 DL architectures and baseline

Convolutional Neural Networks (CNNs) models are known to work well with image-like data, where the dynamics are spatially correlated, and LSTM units have been shown to effectively capture the long and short-term dependencies in time series data (Polson and Sokolov, 2017; Kumar and Raubal, 2021). All experiments use multilayer convolutional-LSTM DL models with batch normalization (Bjorck et al, 2018).

Table 1: Train-Validation split for each city showing different seasons. The number of training data points for a task $Task(i_o, p_h, s)$ is given by the number of days in training data (#days train) multiplied by $(96 - (2 \cdot i_o + p_h + 1)$ due to missing data points at the edges of each day

City	Train Start	Val Start	#days train	#days val
London	01.07.2019 ☀	02.11.2019 ✲	105	110
Melbourne	01.06.2020 ✲	15.09.2020 ☀	105	109
Madrid	01.06.2021 ☀	15.09.2021 ✲	105	108

We change the number of filters (denoted by FIL) and the number of layers (denoted by DEP) to vary the models' size and, consequently, their realisable complexity. Across all experiments, the filter size is set to 3x3, and the padding (for convolution) used is such that the output and input for a convolution retain the same dimensions. To avoid any ambiguity, the model f-DEP-4-FIL-128 implies [Input Layer] → [4x (ConvLSTM2D (128 filters) → Batch Normalization)] → [Conv3D] → [Output Layer]. (cf. Appendix B Fig. 13b and Fig. 13a for representative diagrams of models corresponding to two arbitrarily selected values of DEP and FIL)

The Adam optimiser was used with an initial learning rate of 0.001 for all experiments unless otherwise mentioned. The loss function used to train the models is the mean squared error, and all models use the ReLU activation (Li and Yuan, 2017) function in the last (output) layer.

The persistence forecast (defined as $f(X) = X$) is used as the baseline. The persistence forecast, also referred to as the naïve model in some literature, has a straightforward implementation and is known to perform better than other simple baselines such as the historical average for traffic prediction tasks (Aldwyish et al, 2021). Any DL model which, after training for 30 epochs, did not perform better than the persistence forecast on validation data was not included for drawing the inferences in this paper. Moreover, across all experiments, MC, IC, IC_{Pixel}, and IC_{Temp} are always computed using $\mathcal{D}_{train}$, whereas validation error (often shortened to Val MSE) is always calculated using the validation data.

All models were implemented in tensorflow (Abadi et al, 2015) and trained on an NVIDIA V100 GPU with 16GB of VRAM.

5 Validation Experiments

In this section, we first present a set of three experiments to validate our proposed metrics MC and IC to ensure that the metrics align with the established findings in the literature on effective model complexity (for MC) and the complexity of traffic prediction (for IC). After that, we present an experiment to validate our hypotheses regarding the relationship between MC, IC, and the validation error.

Table 2: IC values for tasks with varying i_o, rounded to the nearest integer

Task \ City	London	Madrid	Melbourne
$IC(Task(i_o = 1, p_h = 1, s = 55))$	830	2245	466
$IC(Task(i_o = 2, p_h = 1, s = 55))$	896	2710	475
$IC(Task(i_o = 3, p_h = 1, s = 55))$	1007	3862	545
$IC(Task(i_o = 4, p_h = 1, s = 55))$	1601	5268	801
$IC(Task(i_o = 5, p_h = 1, s = 55))$	2000	6897	956
$IC(Task(i_o = 6, p_h = 1, s = 55))$	2056	79166	1106
$IC(Task(i_o = 7, p_h = 1, s = 55))$	2119	10435	1140
$IC(Task(i_o = 8, p_h = 1, s = 55))$	2478	10975	1156

5.1 Validation: Consistency of Model Complexity with literary evidence

The literature on DL model complexity has established that the effective model complexity of a DL model evolves during training, evolving from a less complex model to a highly complex model (cf. subsection 2.2). Suppose we train a DL model and evaluate its validation error (Val MSE) and model complexity (MC) after every epoch during the training; the MC, an effective model complexity metric, should increase as the validation loss decreases. Moreover, suppose we continue training even after the validation loss stops falling. In that case, the model will likely overfit and achieve a greater complexity with lower training loss but a higher validation loss.

Experiment: To validate that our proposed metric MC demonstrates this behaviour, we choose a task: $Task(i_o = 4, p_h = 1, s = 55, city = London)$ and train a model f-DEP-4-FIL-128 which comprises four hidden convolutional layers and each layer having 64 filters of size 3×3. Batch-normalisation layers are present between the convolutional layers.

Results and Discussion: The evolution of MC across epochs during the training process is shown in the Fig. 5. For the first three epochs of the training, the small values of MC (≈100) correspond to high validation and training losses (> 1000). As training progresses, the losses decrease (≈ 750), along with a higher MC value (≈ 250). If we continue training the model, the validation loss stops decreasing at around epoch five, but the training loss continues to decrease (overfitting regime). In the overfitting regime, the MC continues to increase, thus validating that the behaviour of MC aligns well with the evidence from the literature.

5.2 Validation: Consistency of Intrinsic Complexity with literary evidence

The literature review subsection on the empirical evidence from data-driven traffic prediction (subsection 2.1) revealed that the task complexity increases with the number of time steps in the prediction (multi-step is more complex) and increases with the prediction horizon (longer horizons are more complex), and increases with the decreasing size of tiles (microscopic prediction is more complex).

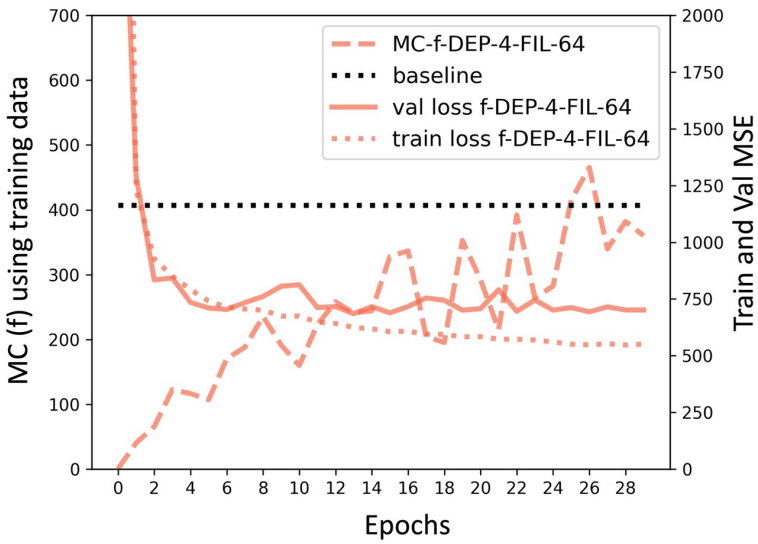


Fig. 5: Evolution of model complexity, training loss, and validation loss during training for task: $Task(i_o = 4, p_h = 1, s = 55, city = London)$. The baseline label refers to the validation loss achieved using the naïve model, given by $(f(X) = X)$ and having a complexity value of 0

Table 3: IC values for tasks with varying p_h rounded to the nearest integer

City Task	London	Madrid	Melbourne
$IC(Task(i_o = 4, p_h = 1, s = 55))$	1601	5268	801
$IC(Task(i_o = 4, p_h = 2, s = 55))$	1769	7379	984
$IC(Task(i_o = 4, p_h = 3, s = 55))$	2098	8190	1040
$IC(Task(i_o = 4, p_h = 4, s = 55))$	2380	9097	1185
$IC(Task(i_o = 4, p_h = 5, s = 55))$	2249	9554	1145
$IC(Task(i_o = 4, p_h = 6, s = 55))$	2378	10014	1264
$IC(Task(i_o = 4, p_h = 7, s = 55))$	2522	10380	1215
$IC(Task(i_o = 4, p_h = 8, s = 55))$	2597	9947	1232

Table 4: IC values for tasks with varying scales, rounded to the nearest integer

City Task	London	Madrid	Melbourne
$IC(Task(i_o = 4, p_h = 1, s = 25))$	1165	1334	244
$IC(Task(i_o = 4, p_h = 1, s = 35))$	1386	3160	467
$IC(Task(i_o = 4, p_h = 1, s = 45))$	1480	3159	678
$IC(Task(i_o = 4, p_h = 1, s = 55))$	1601	5268	801
$IC(Task(i_o = 4, p_h = 1, s = 65))$	1582	5825	859
$IC(Task(i_o = 4, p_h = 1, s = 75))$	1775	7181	9311
$IC(Task(i_o = 4, p_h = 1, s = 85))$	1667	7916	972
$IC(Task(i_o = 4, p_h = 1, s = 95))$	1721	12011	1049
$IC(Task(i_o = 4, p_h = 1, s = 105))$	1767	11531	1005

Experiment: In this experiment, we create several tasks by varying the three parameters in our Task definition: i_o, the input-output length, p_h, the prediction horizon, and s, the scale of prediction. For each city, we vary the three key parameters of our $Task$ definition (i_o, p_h, s) to create new tasks of varying complexities. The ranges of parameters are shown below:

- $i_o \in \{1, 2, 3, \mathbf{4}^*, 5, 6, 7, 8\}$
- $p_h \in \{\mathbf{1}^*, 2, 3, 4, 5, 6, 7, 8\}$
- $s \in \{25, 35, 45, \mathbf{55}^*, 65, 75, 85, 95, 105\}$

To limit the number of experiments, when varying one dimension, the other two dimensions assume default values, which are depicted with * in the list above. Thus, the total number of tasks for one city is $8 + 8 + 9 = 25$ and the total number of tasks across all cities $= 75$. For each task, we compute the Intrinsic complexity ($IC(Task)$) using Equation 11 and summarise the values in three tables.

Results and Discussion: The tables Table 2, Table 3, and Table 4 show the calculated IC values for each dimensions. The trends in IC values agree with the known findings from the literature and validate IC as a reliable measure of task complexity for city-wide traffic prediction. Moreover, we find that the tasks from Madrid consistently have higher IC values than those from London and Melbourne. This trend is expected since the data from each city is collected from different years. The data from Melbourne was collected from June 2020 to December 2020, which was the year when there were severe COVID-related restrictions on travel. Very few vehicles on the road imply fewer interactions between different drivers, resulting in traffic conditions which are likely to be homogeneous and free-flowing, thereby having lesser complexity.

Additionally, the differences in the intrinsic complexities (δIC) across tasks in the same dimension can be used to enhance the robustness of predictions. For instance, we observe from Table 4 that the IC for $Task(i_o = 4, p_h = 1, s = 25, city = Madrid)$ is 1334 while that for $Task(i_o = 4, p_h = 1, s = 105, city = Madrid)$ is 11531. This difference suggests that the task with $s = 25$ is simpler and less prone to errors when using a DL model. Consequently, predictions for the simpler task ($s = 25$) can serve as a sanity check for the model trained on the more complex task ($s = 105$) since each urban pixel in $s = 105$ can be mapped to a pixel in $s = 25$.

Table 5: Pearson Correlation Coefficients between Validation MSE and complexity metrics

Metric	Coefficient	p-value
Val MSE vs $\lvert IC - MC \rvert$	0.9497	1.06e-3
Val MSE vs IC	0.9494	1.07e-3
Val MSE vs MC	0.6869	8.82e-2

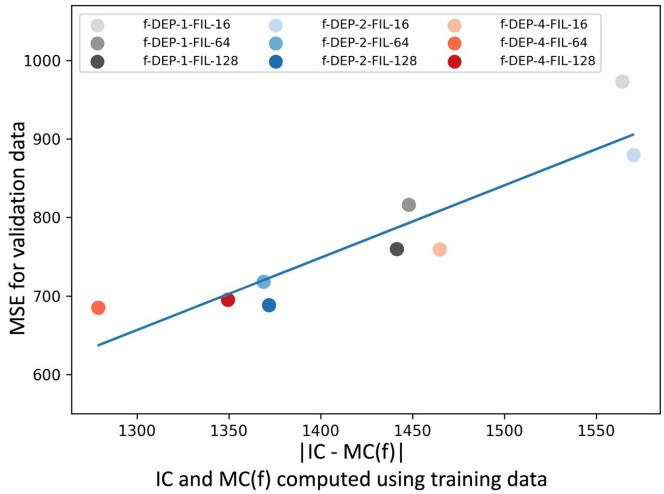


Fig. 6: When training several models for a single task, we find a high correlation between the validation error and the unrealised modelling potential, denoted by $|IC - MC|$. Pearson Correlation Coefficient: 0.910 p-value: 6e-04

5.3 Validation of proportionality hypotheses

We perform two experiments to test the hypotheses introduced in subsection 3.5 regarding the relationship between IC, MC, and the validation error. First, we train several models for a single task and then train a single model for multiple tasks of varying intrinsic complexities.

Experiment (Validation error $\propto$ |IC − MC|): For a single task $Task(i_o = 4, p_h = 1, s = 55, city = London)$, we train a total of 9 DL models, each of which is a Convolutional LSTM model defined by f-DEP-FIL where DEP $\in \{1, 2, 4\}$ and FIL $\in \{16, 64, 128\}$. All models have a BatchNormalisation layer after each Convolutional LSTM layer, as shown in the Fig. 13. All models are trained for 30 epochs, and the validation MSE and Model Complexity (MC) are recorded after each epoch. Additionally, we know from the previous experiment (Table 2) that the IC for this Task is 1601; hence we compute $|IC - MC|$ for each epoch. We then plot the lowest validation error for each model against the MC for the corresponding epoch.

The plot shown in Fig. 6 show that the validation MSE increases as the $|IC - MC|$ increases, and the relationship appears almost linear. This observation validates our hypothesis that for a single task, validation error is proportional to the difference between the Intrinsic Complexity and the Model Complexity.

Experiment (Validation error $\propto$ IC): We choose 8 tasks of varying complexities by varying their p_h and train the same DL model (f-DEP-3-FIL- 64) for all of them. The 8 tasks are $Task(i_o = 4, p_h \in (1, 2, ..., 8), s = 55, city = London)$. The model is trained for 30 epochs for each task, and the validation MSE and Model Complexity (MC) are recorded after each epoch. Additionally, we use the IC values from (Table 3) to compute the $|IC - MC|$ for each epoch. Finally, we choose the best epoch

(lowest validation error) for each Task and plot the validation MSE against the MC, IC, and $|IC - MC|$ as shown in Fig. 7.

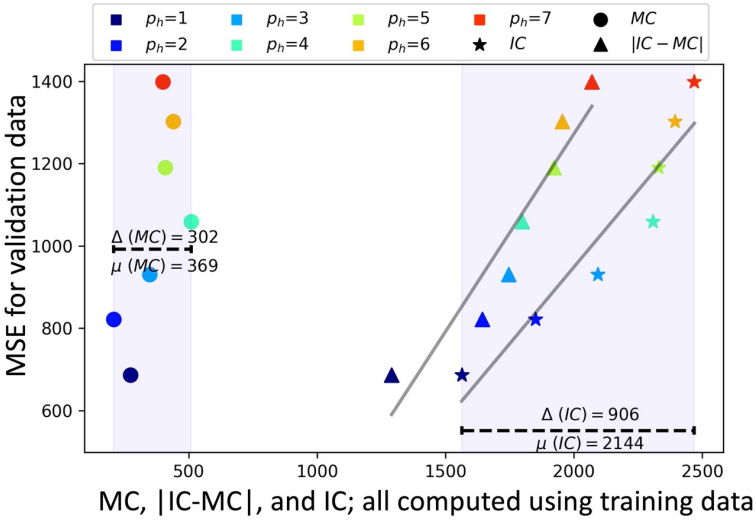


Fig. 7: When training a single model for multiple tasks, we find a high correlation between the validation error and the Intrinsic Complexity (IC). Here, $\Delta(MC)$ and $\Delta(IC)$ represent the ranges of MC and IC values across 7 tasks. $\mu(MC)$ and $\mu(IC)$ represent the mean of MC and IC values across those tasks. Clearly, $\mu(MC) << \mu(IC)$ and $\Delta(MC) << \Delta IC$; implying that the expression $|IC - MC|$ is influenced mostly by IC

Similar to the previous experiment, we find that the validation MSE increases with increasing $|IC - MC|$. Additionally, it is interesting to note in Fig. 7 that the total variation in the values of MC for a single model across all eight tasks is much smaller compared to IC. As a result, the term $|IC - MC|$ is mainly influenced by the IC, thus validating our hypothesis that given a model, the validation MSE increases with an increase in the intrinsic complexity of the Task and shows a high correlation with IC, as shown in Table 5). Alternatively, the IC, a quantity computed *before* training the model, can be used to estimate the validation error, which is a quantity that can be computed only *after* training the DL model.

6 Use Cases of proposed metrics

In this section, we demonstrate selected use cases for the proposed metrics. The use cases discussed here supplement the well-known uses of model complexity metrics as discussed in section 2.

6.1 Improving model selection using Model Complexity

Typically, when attempting to propose a model architecture for a given task, if the validation loss is not within desirable limits, it is often imperative to test the performance of a bigger model by changing the hyperparameters to increase the model capacity. The change of hyperparameters often results in changes in the number of trainable parameters of the DL model. When training different models on the same dataset, the MC might exhibit some relationship with the number of trainable parameters, but the exact nature of such a relationship is unclear. Hence, using the number of parameters as the model complexity metric could be misleading. Instead, one should use an effective model complexity metric, like the proposed MC, to evaluate the impact of changing hyperparameters. (cf. Appendix B Fig. 14 for a bar graph showing an example comparison between MC and the number of trainable parameters)

Experiment: We choose the same task as Experiment 1: $Task(i_o = 4, p_h = 1, s = 55, city = London)$ and train a total of 9 different Conv-LSTM models with the depths $\in \{1, 2, 4\}$ and the number of filters $\in \{16, 64, 128\}$. Each model is trained for 30 epochs. After every epoch, we compute the MC of the model and evaluate the validation and training losses. We plot each model's validation and training loss for the best five epochs versus the MC values for those epochs.

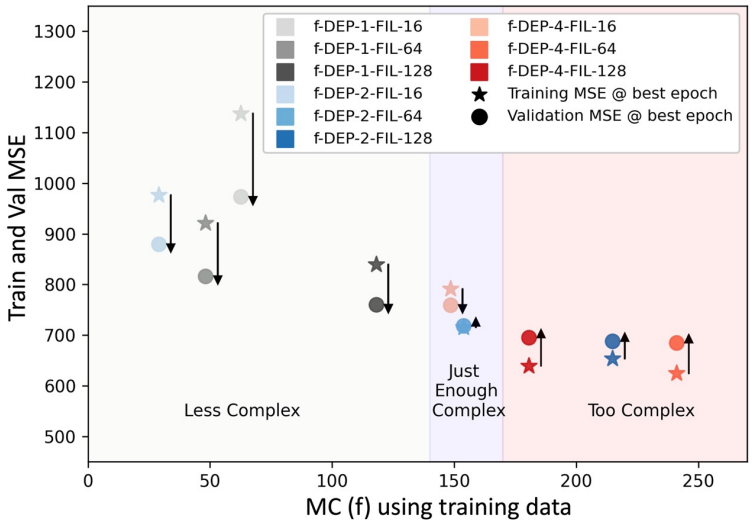


Fig. 8: Model complexity (MC) for 12 different models. the MC and training MSE are plotted only for the best epoch- corresponding to the epoch that achieved the lowest validation MSE

Results and Discussion: The scatter plot in Fig. 8 demonstrates how MC can guide the choice of hyperparameters such that a model which is *just enough* complex after training is selected. For the example in the figure, while it is clear that the model f-DEP-2-FIL-128 has the smallest difference between the validation and training losses

and should be selected, the complexity value ($MC(f)$ on the X-axis) provides an additional layer of validation to the model selection process.

Additionally, we know from the literature on DL effective model complexity that given two models with similar validation loss, selecting the model with lower effective complexity will result in lower errors on unseen data (Hu et al, 2021). Hence, out of all the models shown in Fig. 8, even though the model f-DEP-4-FIL-128 appears to have the same validation errors as f-DEP-2-FIL-64, the latter is a better choice since it has lower model complexity.

6.2 Explaining the effect of regularisation techniques using Model Complexity

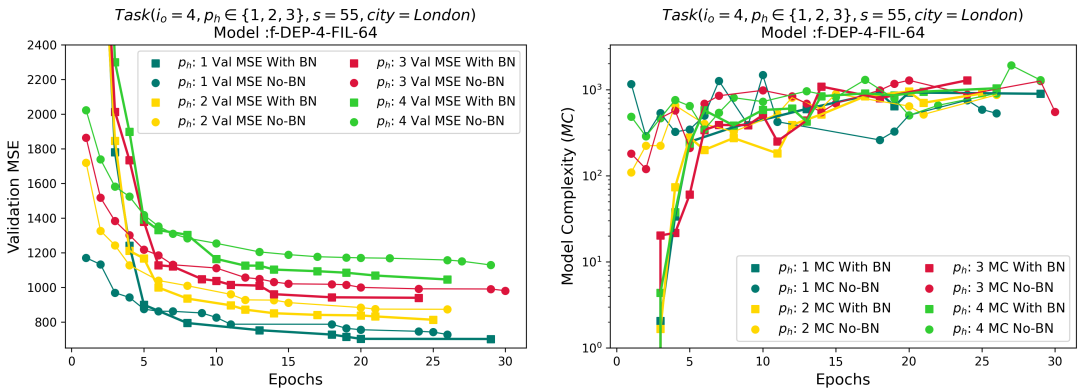
Regularisation techniques have been shown to improve the generalisation errors of DL models. Here, we demonstrate how MC can explain such generalisation using batch normalisation (Ioffe and Szegedy, 2015). While batch normalisation is not a regularisation technique in the traditional sense, it has been shown to have a regularising effect by reducing the changes in the distribution of activation during training (referred to as internal covariate shift). Moreover, using batch normalisation has been shown to significantly reduce the number of epochs required for training (Bjorck et al, 2018). Batch normalisation is known to increase the stability of the optimisation algorithm, specifically by making the gradients more predictable and 'well-behaved', thereby convergence to a more stable, often lower minima (Santurkar et al, 2018, p.9).

Experiment: We choose four tasks: $Task(i_o = 4, p_h \in \{1, 2, 3, 4\}, s = 55, city = London)$ and for each task, train two otherwise identical models (f-DEP-4-FIL-128), with, and without batch normalisation layers. After each training epoch, the validation MSE and the corresponding MC are evaluated and plotted in Fig. 9.

Table 6: Mean and Standard Deviation for MC across all epochs, which performed better than the baseline for the model f-DEP-4-FIL-64. All values have been rounded off to the nearest integer

p_h for Task	With Batch Normalisation		Without Batch Normalisation	
	Mean (μ)	Standard Deviation (σ)	Mean (μ)	Standard Deviation (σ)
1	596	313	621	374
2	616	396	583	418
3	707	361	651	328
4	754	336	878	561

Results and Discussion: As shown in Fig. 9a, the models with batch normalisation result in better performance, exhibited by a faster decrease in the validation loss (MSE). This better performance can be explained by tracking the evolution of MC across epochs, as shown in Fig. 9b. On the one hand, for the models without batch normalisation, the MC achieve high values in just a few epochs (1-2) yet continues to increase afterwards. On the other hand, the MC evolves slowly for models with batch normalisation layers, suggesting that batch normalisation constraints the model complexity within 'just enough' complexity during training, thereby preventing



(a) (b)

Fig. 9: Evolution of model complexity, training loss, and validation loss during training. Both MC and Val MSE plots have been smoothed by plotting the values only for those epochs which achieved the lowest validation loss until then. (cf. Appendix B Fig. 15 for unfiltered plot)

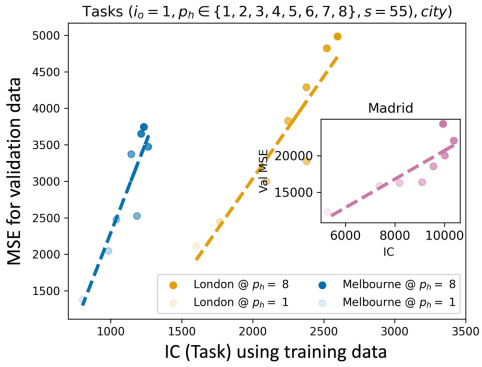
overfitting. The constraining of MC is supported by the mean and standard deviation of MC across various training epochs as shown in Table 6. Only the epochs where the DL model performed better (lower validation MSE) than the baseline were considered for computing the mean. This experiment demonstrates that using MC to measure the model complexity during training could help analyse the impact of novel optimisation algorithms and regularisation techniques in future.

6.3 Using Intrinsic Complexity to estimate validation errors before training a model

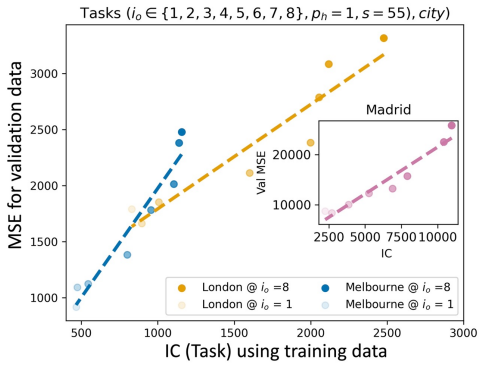
In light of our hypothesis that the validation error is proportional to the intrinsic complexity when using the same model for tasks of varying complexities, the IC can estimate the validation errors apriori.

Experiment: In this experiment, we consider a total of 75 tasks of varying complexities along three task dimensions (i_o, p_h, and s). The tasks are the same as summarised in subsection 5.2. We then train 75 DL Conv-LSTM models with fixed hyperparameters. We experimented with different numbers of filters, depth, and the batch normalisation layers to ensure that a single set of hyperparameters beats the baseline across all tasks and finally chose f_{def} which is represented by [Input Layer] → [ConvLSTM2D (128 filters) → Batch Normalisation] → [ConvLSTM2D (64 filters) → Batch Normalisation] → [2x ConvLSTM2D (64 filters)] → [Conv3D] → [Output Layer]. (cf. Appendix B Fig. 13c for a pictorial representation of the architecture)

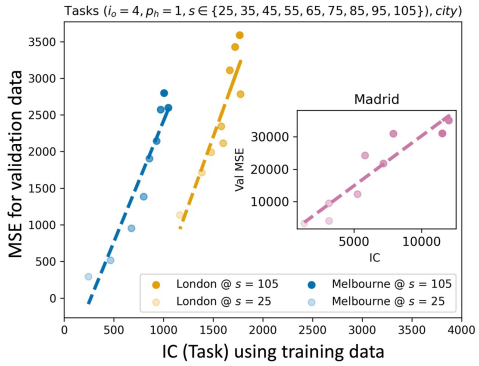
Finally, we plot the validation errors with respect to $IC(Task)$ and fit a linear regression model for each dimension. All models are trained for 30 epochs, and the training is stopped when the validation loss does not decrease for five consecutive



(a) Validation MSE vs IC for varying prediction horizon



(b) Validation MSE vs IC for varying input-output length



(c) Validation MSE vs IC for varying scales

Fig. 10: Linear relationship between IC and Validation error; Lightness indicates the value along the experimental axis. *e.g.* The light yellow marker is London at prediction horizon (p_h) = 1, whereas the dark yellow triangle is London at p_h=8. Pixels which were zero throughout the day are excluded while computing the validation errors since the data is spatially sparse, as shown in the heatmaps

epochs (early stopping = 5). The batch size was 64 up to scale 75 and 8 thereafter. The Adam optimiser was used with an initial learning rate of 0.001.

Results and discussion: Fig. 10a, Fig. 10b, and Fig. 10c show that the metric $IC(Task)$ exhibits a linear relationship with the validation mean-squared error across all task dimensions (i_o, p_h, and s) and all three cities. The linearity implies that the IC can be used to compute the estimates of validation error before training a DL model for the given task. Consequently, errors from previous work could be used to estimate the error for a new task by computing the IC of the new task and comparing it to that of previous work, providing valuable insights for model selection

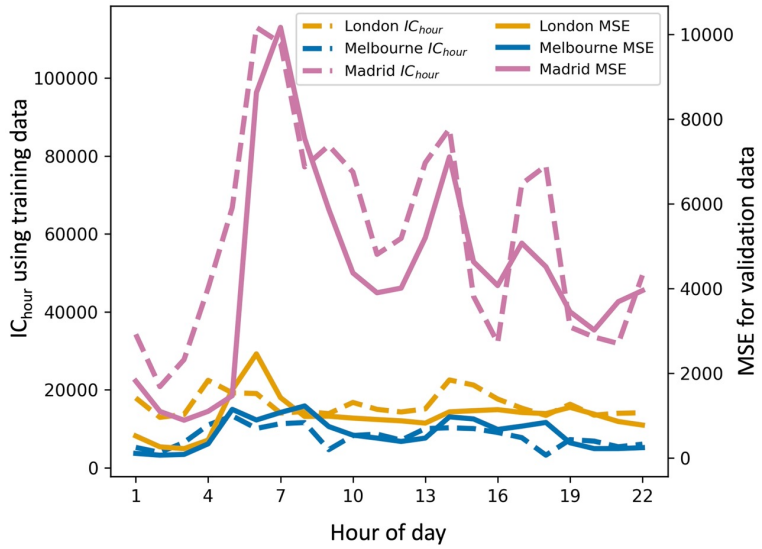


Fig. 11: Hourly distribution of errors and IC_{Temp} for $Task(i_o = 4, p_h = 1, s = 55, city)$

6.4 Using IC for Pre-training estimates of temporal and spatial errors

Equation 12 and Equation 13 are used to compute IC_{Temp} per hour and IC_{Pixel} per pixel respectively for selected tasks. Both ICs are computed using only the $\mathcal{D}_{train}$.

6.4.1 $\mathbf{IC_{Temp}}$ and temporal variation of errors

For each city, we select the task, $Task(i_o = 4, p_h = 1, s = 55)$ and train f_{def}. The IC_{Temp} is computed using the training data for each hour of the day, and the mean hourly validation error is computed using validation data.

Results and discussion: Fig. 11 shows the plot of mean hourly validation MSE against the hourly IC_{Temp} for each task. As evidenced by the plots, the IC_{Temp} exhibits varying degrees of correlation with the validation MSE across three cities and two tasks. Table 7 shows the computed correlation values. A high correlation is observed for Madrid, but lower values are observed for Melbourne and London. We hypothesise that this discrepancy may be attributed to the differences in changes in driving patterns between the data points sampled from $\mathcal{D}_{train}$ and $\mathcal{D}_{val}$ since they belong to different seasons Table 1. The training data for London and Madrid are from the summer months, while the validation data are from the winter months. The seasons are reversed for Melbourne, where the training data is from winter, and the validation data is from summer. Moreover, the data for Melbourne is from 2020, which marked the peak of lockdowns and, consequently, the

Table 7: Pearson's correlation between Val MSE and $IC_{Temp}(Task(i_o = 4, p_h = 1, s = 55, city)$

City	Pearson's Corr	p-value
London	0.48	0.024
Melbourne	0.25	0.270
Madrid	0.67	0.001

Table 8: Pearson's Correlation between each IC_{Pixel} and spatial variation of Val MSE for $Task(i_o = 4, p_h = 1, s = 85, city)$

Error	City	Pearson's Corr	p-value
Val MSE (non-zero)	London	0.595	4.772e-137
Val MSE (non-zero)	Madrid	0.737	6.313e-242
Val MSE (non-zero)	Melbourne	0.834	0
Val MSE	London	0.615	0
Val MSE	Madrid	0.721	0
Val MSE	Melbourne	0.849	0

6.4.2 IC$_{Pixel}$ and spatial variation of errors

For each city, we select as task $Task(i_o = 4, p_h = 1, s = 85)$ and compute IC_{Pixel} for each pixel in the city using $\mathcal{D}_{train}$. We then train f_{def} for each city.

Results and discussion: The IC_{Pixel} exhibits a high correlation with the spatial distribution of errors as summarised in Table 8. Since depending on the scale, the sparsity of non-zero pixel varies, as shown in Fig. 4, we list the correlation for all pixels and those where the ground truth values were more than 0 (referred to as Val MSE (non-zero)). A high correlation for non-zero pixel locations establishes IC_{Pixel} as a reasonable metric to quantify pre-training estimates of the spatial variation of the complexity of a city-wide traffic prediction task across a city.

By dividing the city into pixels of high and low complexity, or temporal bins of high and low complexity, the metric IC opens new avenues for attacking the challenge of city-wide traffic prediction, thanks to the rich literature on efficient dataset utilisation (*e.g.* additive NNs (Park et al, 2020) and ensembles of models of low and high complexity (Maini et al, 2022)).

6.5 Using IC to evaluate design pathways

Pivotal design choices related to model hyperparameters and design pathways emerge in DL-based traffic prediction. Several hyperparameter-related choices can readily benefit from the use of IC. For example, given a task that has higher IC compared to another task that has been studied in the literature (and is shown to need h hidden layers for a specific DL architecture $n_{hidden} = h$), one can limit the search of the right number of hidden layers to $n_{hidden} \geq h$, thereby saving model training time, which would otherwise have been lost experimenting with $n_{hidden} \leq hid$. While several such

design pathways can be evaluated using IC values tabulated in Table 4, Table 3, and Table 4, this subsection focuses on the more nuanced choices, such as opting for single versus multiple time-step predictions or predicting next-step (5-minute) versus next-hour prediction.

In some instances, the deployment requirements guide these decisions, whereas the tractability of tasks guides the rest. For example, it may be observed that for the fixed number of time steps (say $i_o = 4$), a DL model yields significantly lower validation errors while predicting traffic after 5 minutes ($p_h = 1$) than while predicting traffic after 15 minutes ($p_h = 3$). In such instances, researchers might prefer to generate short-term predictions and use a rolling horizon prediction to generate the prediction at 15 minutes; finally, out of the two options, the model which achieves a smaller validation error can be selected. However, this approach necessitates training multiple models to identify the optimal prediction strategy.

The metric IC offers a strategic advantage in this context. It facilitates the assessment of simpler pathways, such as model composition, by providing a computationally efficient means to gauge their feasibility. Once the most effective pathway is ascertained through IC, focused efforts can be directed towards actual model training.

Let us consider the task $T1 : Task(i_o = 4, p_h = 3, s = 55, city = London)$. The modelling requirements for $T1$ could potentially be met by composing $T1$ using three easier tasks with shorter prediction horizons, *viz.* $T2 : Task(i_o = 4, p_h = 1, s = 55, city = London)$. Given that $T2$ is inherently less complex than $T1$, one might investigate if a rolling horizon is a better choice to achieve $T1$ using $T2$. Specifically, if f_{T1} was trained for task $T1$ and f_{T2} was trained for task $err_{f_{T2}}$ and the validation errors were err_{fT1} and $err_{f_{T2}}$ respectively, one might check if $err_{f_{T1'}} < err_{f_{T1}}$, where $f_{T1'}(x) = f_{T2}(f_{T2}(f_{T2}(x)))$. However, while such a check is possible only after training the model and evaluating the errors, IC helps us evaluate these pathways before training any DL model. From Table 4, we find that $IC(T1) = 1601$ and $IC(T2) = 2098$, implying that $IC(T2) << 3 \times IC(T1)$, hence it is highly likely that the rolling horizon pathway is not beneficial. One is better off by targeting the efforts to train a single DL model for $T1$.

Similarly, when choosing between multi-step prediction (represented by $i_o > 1$ in our task nomenclature) and iterative multi-step prediction, wherein a rolling horizon (*e.g.* $f(f(X))$) prediction is used to achieve multi-step prediction, the IC can help guide the choice. For instance, in Yin and Dai (2022), the authors experimented with several datasets and DL models. They concluded that multi-step prediction be attempted first instead of iterative prediction because the error accumulation in iterative prediction offsets the complexity gain. However, their conclusion was possible only after much experimentation, including training large DL models. Using IC, we can evaluate such pathways before training the model. For instance, if the given task is $IC(Task(i_o = 2, p_h = 1, s = 55, city = London))$, and we wish to evaluate if using a multi-step iterative prediction using $IC(Task(i_o = 1, p_h = 1, s = 55, city = London))$ might be a preferred pathway, we see from Table 2, that $IC(Task(i_o = 2)) = 896$, whereas $IC(Task(i_o = 1)) = 829$. Since $829 \times 2 >> 896$, the multi-step task is a preferred choice for the given task $IC(Task(i_o = 2, p_h = 1, s = 55, city = London))$.

7 Conclusion

In this paper, we presented two metrics - a model complexity (MC) metric for DL models that predict city-wide traffic and an intrinsic complexity (IC) metric for the inherent complexity of the given traffic prediction task. While MC helps us explain various aspects of DL model training and generalisation, the IC is model-invariant and can, therefore, be computed *before* any DL model is trained. We validated the presented metrics using systematic experiments to ensure that the metrics align with the known evidence from the DL literature. Moreover, we found that IC exhibits a linear relationship with the validation error and hence can be used to estimate the validation error before training a DL model, thereby reducing the element of surprise and significantly saving computational resources. Furthermore, we extended the IC into three derived metrics - IC_{Temp}, IC_{Pixel}, and IC_{Patch} - that can be used to compute the intrinsic complexity of the prediction task for any time-of-day, any urban tile (pixel), and any spatial patch (group of nearby urban tiles) in the city. These derived metrics are also highly correlated with the temporal and spatial distribution of errors.

Implications to DL-based traffic prediction: An apriori estimate of the spatial and temporal variation of prediction errors helps segment the city into high and low-complexity regions. Such a segmentation opens new avenues for improving prediction using ensembles of high and low-complexity models. Similarly, as discussed in subsection 5.2, the robustness of city-wide traffic prediction can be enhanced by using a coarser spatial scale (low s) DL model to serve as a sanity check for finer spatial scales (high s) using an inter-scale pixel matching.

Implications to rescue routing: While the focus of this paper was to guide the workflow and make informed model choices regarding DL models in city-wide traffic prediction, a direct practical use of our metrics lies in routing emergency vehicles. Given our modelling limitations, there might still be scenarios where the traffic in specific parts of the city is too complex and cannot be predicted reliably. An apriori estimate of high prediction errors can be used to determine the most reliable travel time estimates for emergency vehicles by routing emergency vehicles away from high-complexity urban tiles (pixels) in the city - thus reducing the uncertainty in travel time predictions. The travel time uncertainty is a major challenge and an active research area in the rescue-routing community (Anuar et al, 2021).

Future works: The focus of this study was the DL models, which attempt to model traffic dynamics without any additional data sources. Specifically, this study did not include the benefits of data fusion from multiple sources or the possibility of using models inherently designed to perform well for different scales since they skew the inherent complexity of the traffic dynamics. For instance, in the case of a planned event, the social media data could be used to estimate the traffic demand several hours in advance, thereby significantly reducing the complexity of the prediction as one could estimate using solely the traffic data. Practically similar performance for different horizons, ranging from 30 minutes to 2 hours, has been reported in context-based studies where additional point-of-interest (POI) data is used along with traffic data and a DL model capable of fusing different layers of data is proposed (Zhang

et al, 2023). Similarly, several multi-scale models have been proposed for traffic volume prediction, created either using techniques such as skip-connections between the convolutional hidden layers at different scales, such as the multi-scale ConvLSTM-Resnet (He et al, 2022) and the U-Net (Eichenberger et al, 2022) or using multi-scale spatio-temporal graph-convolutions (Wang and Jing, 2022; Wang et al, 2022a). Given the architectural designs that allow for information sharing between hidden layers at different scales, such models might be capable of producing similar validation errors across different scales in the future, thereby reducing the $|IC - MC|$ gap, hence decreasing the sensitivity of the proposed metrics. However, the metrics introduced in this paper (MC and IC) are foundational, as they are defined by averaging the sample complexity values across all data tensors. In the future, one might propose more nuanced statistics to capture the distribution of sample complexity values to increase the sensitivity of these metrics and retain their usefulness as more capable DL models become available.

As a trade-off between computation time and the extensive nature of our experiments, we trained a total of 75 DL models, 25 for each city. This results in a sparse coverage of the task space, and further experimentation could lead to additional insights about the proposed metrics. In future work, focusing on a single task dimension and sampling the tasks more densely would be interesting. Moreover, focusing on a single task dimension would allow us to use other DL architectures (such as Graph Neural Networks). Finally, in this paper, using the scale dimension, we investigated the effect of spatial aggregation on the complexity of the task, but the impact of temporal aggregation remains unexplored.

Author contributions

N.K.: Nishant Kumar, H.M.: Henry Martin, M.R.: Martin Raubal.

N.K., M.R., and H.M. conceptualised the study. N.K. and H.M. developed the methodology. N.K. wrote the codes, generated the results and figures, and prepared the first draft of the manuscript. H.M. reviewed the analytical formulations. H.M. and M.R. reviewed the manuscript. M.R. supervised the study and was responsible for funding acquisition. All authors agree with the final version.

Funding

This work is an outcome of the Future Resilient Systems project at the Singapore-ETH Centre (SEC) supported by the National Research Foundation, Prime Minister's Office, Singapore, under its Campus for Research Excellence and Technological Enterprise (CREATE) programme.

Acknowledgements

We thank HERE Technologies for providing the Traffic4cast dataset. The authors thank Kiran Bacsa and Wei Liu from Future Resilient Systems, Singapore, for their insightful discussions and valuable feedback.

Availability of Data and Materials

The data can be downloaded through the competition repository [2]. The source code of our experiments is available at our lab's github link [3]. Detailed instructions on how to evaluate the proposed metrics for new datasets are included in our repository.

Statements and Declarations

Conflict of Interest The authors declare no known competing interests.

Ethical Approval Not applicable

References

Abadi M, Agarwal A, Barham P, et al (2015) TensorFlow: Large-scale machine learning on heterogeneous systems. URL https://www.tensorflow.org/, software available from tensorflow.org

Aldwyish A, Tanin E, Xie H, et al (2021) Effective traffic forecasting with multi-resolution learning. In: 17th International Symposium on Spatial and Temporal Databases, pp 44–53

Allen PM (2001) A complex systems approach to learning in adaptive networks. International Journal of Innovation Management 5(02):149–180

Anuar WK, Lee LS, Pickl S, et al (2021) Vehicle routing optimisation in humanitarian operations: A survey on modelling and optimisation approaches. Applied Sciences 11(2):667

Arpit D, Jastrzębski S, Ballas N, et al (2017) A closer look at memorization in deep networks. In: International conference on machine learning, PMLR, pp 233–242

Asuncion A, Newman D (2007) UCI machine learning repository

Batty M, Torrens PM (2005) Modelling and prediction in a complex world. Futures 37(7):745–766

Bejani MM, Ghatee M (2021) A systematic review on overfitting control in shallow and deep neural networks. Artificial Intelligence Review 54(8):6391–6438

Belkin M, Hsu D, Ma S, et al (2019) Reconciling modern machine-learning practice and the classical bias–variance trade-off. Proceedings of the National Academy of Sciences 116(32):15849–15854

Bettencourt LM (2015) Cities as complex systems. Modeling complex systems for public policies pp 217–236

[2] https://github.com/iarai/NeurIPS2022-traffic4cast
[3] https://github.com/mie-lab/Complexity-Aware-Traffic-Prediction

Bialek W, Nemenman I, Tishby N (2001) Predictability, complexity, and learning. Neural computation 13(11):2409–2463

Bjorck N, Gomes CP, Selman B, et al (2018) Understanding batch normalization. Advances in neural information processing systems 31

Boffetta G, Cencini M, Falcioni M, et al (2002) Predictability: a way to characterize complexity. Physics reports 356(6):367–474

Borg M, Tabone A, Bonnici A, et al (2022) Detecting and ranking pornographic content in videos. Forensic Science International: Digital Investigation 42:301436

Chakraborty M, Gates TJ, Sinha S (2023) Causal analysis and classification of traffic crash injury severity using machine learning algorithms. Data science for transportation 5(2):12

Chiang D, Rush AM, Barak B (2021) Named tensor notation. arXiv preprint arXiv:210213196

Chung J, Gulcehre C, Cho K, et al (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:14123555

Eichenberger C, Neun M, Martin H, et al (2022) Traffic4cast at neurips 2021-temporal and spatial few-shot transfer learning in gridded geo-spatial processes. In: NeurIPS 2021 Competitions and Demonstrations Track, PMLR, pp 97–112

Fafoutellis P, Vlahogianni EI (2023) Unlocking the full potential of deep learning in traffic forecasting through road network representations: A critical review. Data Science for Transportation 5(3):23

Gamba M, Chmielewski-Anders A, Sullivan J, et al (2022) Are all linear regions created equal? In: Camps-Valls G, Ruiz FJR, Valera I (eds) Proceedings of The 25th International Conference on Artificial Intelligence and Statistics, Proceedings of Machine Learning Research, vol 151. PMLR, pp 6573–6590, URL https://proceedings.mlr.press/v151/gamba22a.html

Hahn M, Jurafsky D, Futrell R (2021) Sensitivity as a complexity measure for sequence classification tasks. Transactions of the Association for Computational Linguistics 9:891–908. https://doi.org/"10.1162/tacl_a_00403", URL "https://doi.org/10.1162/tacl_a_00403", "https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00403/1957707/tacl_a_00403.pdf"

Han S, Pool J, Tran J, et al (2015) Learning both weights and connections for efficient neural network. Advances in neural information processing systems 28

He R, Liu Y, Xiao Y, et al (2022) Deep spatio-temporal 3d densenet with multi-scale convlstm-resnet network for citywide traffic flow forecasting. Knowledge-Based

Systems 250:109054

Hochreiter S, Schmidhuber J (1996) Lstm can solve hard long time lag problems. Advances in neural information processing systems 9

Hu X, Liu W, Bian J, et al (2020) Measuring model complexity of neural networks with curve activation functions. In: Proceedings of the 26th ACM SIGKDD International Conference on knowledge discovery & data mining, pp 1521–1531

Hu X, Chu L, Pei J, et al (2021) Model complexity of deep learning: A survey. Knowledge and Information Systems 63(10):2585–2619

Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning, pmlr, pp 448–456

Jia Y, Wu J, Du Y (2016) Traffic speed prediction using deep learning method. In: 2016 IEEE 19th international conference on intelligent transportation systems (ITSC), IEEE, pp 1217–1222

Jin G, Yi X, Zhang L, et al (2020) How does weight correlation affect generalisation ability of deep neural networks? Advances in Neural Information Processing Systems 33:21346–21356

Johnson JM, Khoshgoftaar TM (2019) Survey on deep learning with class imbalance. Journal of Big Data 6(1):1–54

Kaplan J (2022) Notes on contemporary machine learning for physicists. Lecture Notes, Department of Physics and Astronomy, Johns Hopkins University, URL https://sites.krieger.jhu.edu/jared-kaplan/files/2019/04/ContemporaryMLforPhysicists.pdf, available online

Kashyap AA, Raviraj S, Devarakonda A, et al (2022) Traffic flow prediction models–a review of deep learning techniques. Cogent Engineering 9(1):2010510

Khajeh Hosseini M, Talebpour A (2023) Probabilistic traffic state prediction based on vehicle trajectory data. Data Science for Transportation 5(3):22

Khulbe D, Kang C, Ukkusuri S, et al (2023) A probabilistic simulation framework to assess the impacts of ridesharing and congestion charging in new york city. Data Science for Transportation 5(2):8

Kumar N, Raubal M (2021) Applications of deep learning in congestion detection, prediction and alleviation: A survey. Transportation Research Part C: Emerging Technologies 133:103432

Li F, Feng J, Yan H, et al (2021) Dynamic graph convolutional recurrent network for traffic prediction: Benchmark and solution. ACM Transactions on Knowledge

Discovery from Data (TKDD)

Li F, Feng J, Yan H, et al (2023) Dynamic graph convolutional recurrent network for traffic prediction: Benchmark and solution. ACM Transactions on Knowledge Discovery from Data 17(1):1–21

Li Y, Yuan Y (2017) Convergence analysis of two-layer neural networks with relu activation. Advances in neural information processing systems 30

Li Y, Yu R, Shahabi C, et al (2017) Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. arXiv preprint arXiv:170701926

Loder A, Ambühl L, Menendez M, et al (2019) Understanding traffic capacity of urban networks. Scientific reports 9(1):1–10

Maini P, Garg S, Lipton Z, et al (2022) Characterizing datapoints via second-split forgetting. Advances in Neural Information Processing Systems 35:30044–30057

Manibardo EL, Laña I, Del Ser J (2021) Deep learning for road traffic forecasting: Does it make a difference? IEEE Transactions on Intelligent Transportation Systems

Montufar GF, Pascanu R, Cho K, et al (2014) On the number of linear regions of deep neural networks. Advances in neural information processing systems 27

Neun M, Eichenberger C, Martin H, et al (2022) Traffic4cast at neurips 2022–predict dynamics along graph edges from sparse node data: Whole city traffic and eta from stationary vehicle detectors. In: NeurIPS 2022 Competition Track, PMLR, pp 251–278

Novak R, Bahri Y, Abolafia DA, et al (2018) Sensitivity and generalization in neural networks: an empirical study. arXiv preprint arXiv:180208760

Papangelou K, Sechidis K, Weatherall J, et al (2019) Toward an understanding of adversarial examples in clinical trials. In: Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2018, Dublin, Ireland, September 10–14, 2018, Proceedings, Part I 18, Springer, pp 35–51

Park M, Lee S, Hwang S, et al (2020) Additive ensemble neural networks. IEEE Access 8:113192–113199

Polson NG, Sokolov VO (2017) Deep learning for short-term traffic flow prediction. Transportation Research Part C: Emerging Technologies 79:1–17

Raghu M, Poole B, Kleinberg J, et al (2017) On the expressive power of deep neural networks. In: international conference on machine learning, PMLR, pp 2847–2854

Rand W (2015) Complex systems: concepts, literature, possibilities and limitations. Modeling complex systems for public policies Brasília: IPEA pp 37–54

Roßbach P (2018) Neural networks vs. random forests–does it always have to be deep learning. Germany: Frankfurt School of Finance and Management

Sammut C, Webb GI (eds) (2010) Rademacher Complexity, Springer US, Boston, MA, pp 823–823. https://doi.org/10.1007/978-0-387-30164-8_690, URL https://doi.org/10.1007/978-0-387-30164-8_690

Santurkar S, Tsipras D, Ilyas A, et al (2018) How does batch normalization help optimization? In: Bengio S, Wallach H, Larochelle H, et al (eds) Advances in Neural Information Processing Systems. Curran Associates, Inc.

Shiftan Y, Ben-Akiva M (2011) A practical policy-sensitive, activity-based, travel-demand model. The Annals of Regional Science 47(3):517–541

Spiliopoulou A, Kontorinaki M, Papageorgiou M, et al (2014) Macroscopic traffic flow model validation at congested freeway off-ramp areas. Transportation Research Part C: Emerging Technologies 41:18–29

Srivastava N, Hinton G, Krizhevsky A, et al (2014) Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research 15(1):1929–1958

Thompson NC, Greenewald K, Lee K, et al (2020) The computational limits of deep learning. arXiv preprint arXiv:200705558

Vapnik V, Levin E, Cun YL (1994) Measuring the vc-dimension of a learning machine. Neural Computation 6(5):851–876. https://doi.org/10.1162/neco.1994.6.5.851

Wang S, Zhang M, Miao H, et al (2022a) Multivariate correlation-aware spatio-temporal graph convolutional networks for multi-scale traffic prediction. ACM Transactions on Intelligent Systems and Technology (TIST) 13(3):1–22

Wang Y, Jing C (2022) Spatiotemporal graph convolutional network for multi-scale traffic forecasting. ISPRS International Journal of Geo-Information 11(2). https://doi.org/10.3390/ijgi11020102, URL https://www.mdpi.com/2220-9964/11/2/102

Wang Y, Baldwin T, Verspoor K (2022b) Noisy label regularisation for textual regression. In: Proceedings of the 29th International Conference on Computational Linguistics, pp 4228–4240

Wu Y, Tan H, Qin L, et al (2018) A hybrid deep learning based traffic flow prediction method and its understanding. Transportation Research Part C: Emerging Technologies 90:166–180

Xiao J, Xiao Z, Wang D, et al (2019) Short-term traffic volume prediction by ensemble learning in concept drifting environments. Knowledge-Based Systems 164:213–225

Yin C, Dai Q (2022) A deep multivariate time series multistep forecasting network. Applied Intelligence 52(8):8956–8974

Yu H, Wu Z, Wang S, et al (2017a) Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks. Sensors 17(7):1501

Yu R, Li Y, Shahabi C, et al (2017b) Deep learning: A generic approach for extreme condition traffic forecasting. In: Proceedings of the 2017 SIAM international Conference on Data Mining, SIAM, pp 777–785

Zhang T, Song W, Fu M, et al (2021) Vehicle motion prediction at intersections based on the turning intention and prior trajectories model. IEEE/CAA Journal of Automatica Sinica 8(10):1657–1666

Zhang Y, Zhao T, Gao S, et al (2023) Incorporating multimodal context information into traffic speed forecasting through graph deep learning. International Journal of Geographical Information Science 37(9):1909–1935

Zhang Z, Li M, Lin X, et al (2019) Multistep speed prediction on traffic networks: A deep learning approach considering spatio-temporal dependencies. Transportation research part C: emerging technologies 105:297–322

Zhao L, Song Y, Zhang C, et al (2019) T-gcn: A temporal graph convolutional network for traffic prediction. IEEE transactions on intelligent transportation systems 21(9):3848–3858

Zheng G, Chai WK, Katos V (2022) A dynamic spatial–temporal deep learning framework for traffic speed prediction on large-scale road networks. Expert Systems with Applications 195:116585

Zhou ZH (2021) Why over-parameterization of deep neural networks does not overfit? Science China Information Sciences 64(1):1–3

Appendix A: Computational overhead of evaluating MC and IC

Both MC and IC are computationally cheap to evaluate, thereby adding to their practical usefulness. Moreover, evaluating IC is more efficient compared to MC since there is no forward pass through the model. Hence, no GPU is required to compute IC. This section discusses the computational overhead and hardware requirements to compute the metrics efficiently.

Computing MC

In subsection 3.3, we discussed that for evaluating the $SC(X|f)$ for an input data tensor x, we need to search for the neighbours of X. If h hours forward and backward search policy for d preceding and succeeding days are used, and the traffic data is sampled every p minutes, the total number of neighbours for each data point is given by Equation 5 as $|\mathbb{T}_X| = (2 \cdot d + 1) \cdot (2 \cdot h \cdot \frac{60}{p} + 1) - 1$. One training pass (also called one epoch time) comprises one forward pass and one backward pass through all $N = |\mathcal{D}_{train}|$ data points. Let us denote the time taken for one training pass for all N data points as E and the ratio of the training pass to the forward pass as α. We have:

$$E = t_{forward} + t_{backward} \tag{17}$$

and

$$\alpha = \frac{E}{t_{forward}} \tag{18}$$

Hence, the computational cost of evaluating MC is given by $|\mathbb{T}_X| \cdot t_{forward}$. When training a DL model, the most readily available time is the training epoch time (E). Hence, we rewrite $t_{forward}$ in terms of E. Finally, expand $|\mathbb{T}_X|$, we get:

$$\text{Computational cost of evaluating MC} = ((2 \cdot d + 1) \cdot (2 \cdot h \cdot \frac{60}{p} + 1) - 1) \cdot \frac{E}{\alpha} \tag{19}$$

If $h = 1$, $d = 3$, and $p = 15$, and, if all data points are used (*i.e.* $N = |\mathcal{D}_{train}|$), the expression in Equation 19 evaluates to $55 \cdot \frac{E}{\alpha}$, implying 55 forward passes through the model. The forward pass is usually several times faster than the backward pass ($\alpha > 1$). For fully connected feed-forward architectures, α can be evaluated analytically to be close to 3. However, the evaluation is analytically intractable for other architectures, such as the Conv-LSTM architecture used in our experiments, and hence, it needs to be evaluated empirically (Kaplan, 2022). Given our Conv-LSTM architecture and hardware (Nvidia-V100 16 GB GPU and architectures as in Appendix B), α varied from $724ms/70ms \approx 10$ for f-DEP-4-FILT-128 to $141ms/37ms \approx 4$ for f-DEP-2-FILT-64. Hence, assuming a conservative estimate of $\alpha = 5$ yields the computational overhead of computing MC as $55/5 \approx 10 \cdot E$.

Moreover, according to Equation 8, the evaluation of MC essentially implies estimating the mean of SC values for all data tensors. A small subset of SC values might be sufficient to estimate the mean within the required margin of error without making

any assumptions on the distribution of SC. At the implementation level, depending on the downstream applications of MC, which determines the tolerated margins of errors, a smaller value of $n < N$ might be sufficient to estimate the MC using a subset of the data. Moreover, if the distribution of SC values is known from previous experiments, theoretical estimates on sufficient sample size n can be obtained.

The size of the chosen subset $\mathcal{D}_{subset} \subset \mathcal{D}_{train}$, given by $|\mathcal{D}_{subset}| = n$ determines the MC computation time, which can be written in terms of the time required for a single training pass (epoch time):

$$\text{Computational cost of estimating MC} = ((2 \cdot d + 1) \cdot (2 \cdot h \cdot \frac{60}{p} + 1) - 1) \cdot \frac{E}{\alpha} \cdot \frac{n}{N} \quad (20)$$

We experimented with smaller values of n for one of the tasks ($Task(i_o = 4, p_h = 4, s = 55, city = London)$). We iterated over n from 100 to the maximum possible N for this task($N = |\mathcal{D}_{train}| = 8714$) in steps of 500, and, at each iteration, the MC and IC were computed. These MC and IC values were compared with the true values (computed using $n = N$), and the absolute percentage difference is shown in Fig. 12. For MC, the estimated mean for a sample size of ≈ 2000 corresponds to 10% deviation from the true MC.

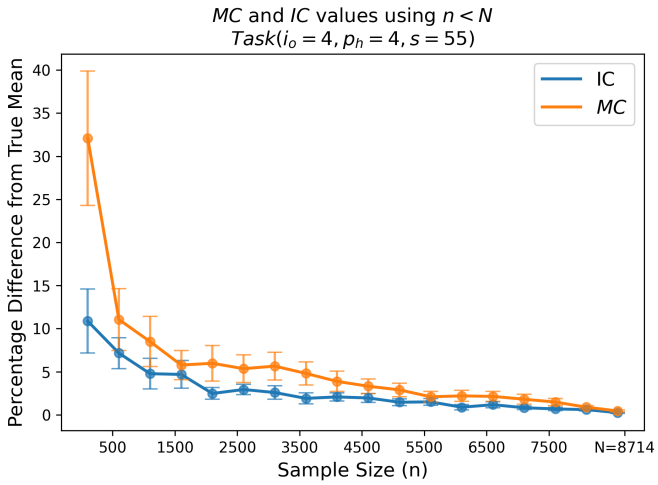


Fig. 12: Errors in estimating MC and IC when using a subset $\mathcal{D}_{subset} \subset \mathcal{D}_{train}$. Here, $n = |\mathcal{D}_{subset}|$ and $N = |\mathcal{D}_{train}|$. This plot depicts the deviation of the sample mean ($n < N$) from the true mean ($n = N$) as a function of sample size (n).

Computing IC

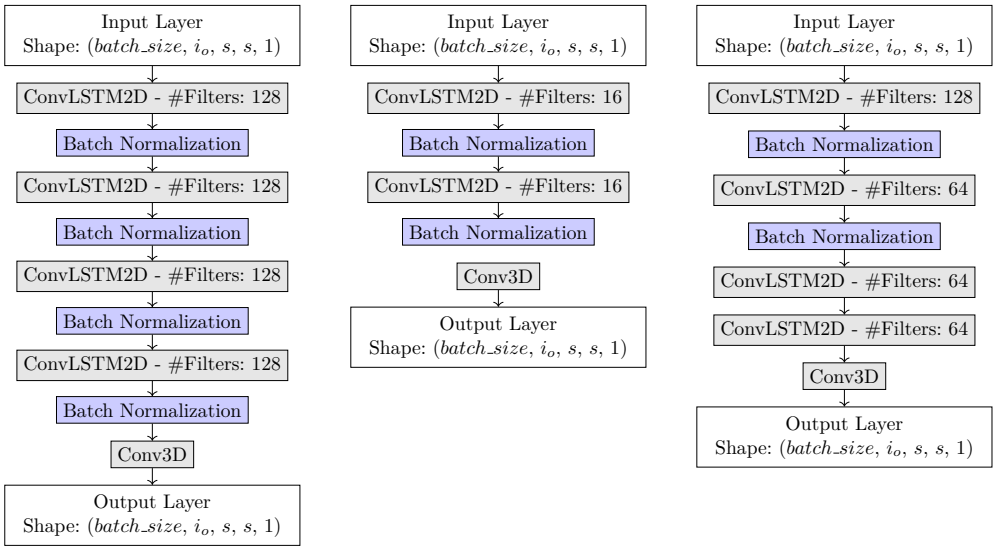
As shown in Equation 11, the evaluation of IC depends only on the L_∞ distances between the tensors in D_{train} and not on the DL model being used. Hence, unlike

MC, the evaluation of IC (and consequently - IC_{temp}, IC_{pixel}, and IC_{patch}) - does *not* require a forward pass through the model, implying that evaluating IC does not need a GPU. The lack of need for a GPU implies that a bigger batch size can be used and is limited only by the machine's RAM, thereby enabling faster evaluation of IC. Additionally, not needing a forward pass implies that the term $\frac{E}{\alpha}$ is replaced with the time taken to read a data tensor, which is much smaller than $\frac{E}{\alpha}$. Given our hardware setup, the time taken to read the input data tensors using a dataloader was approximately half of $\frac{E}{\alpha}$.

Moreover, similar to MC, in practice, the IC can be evaluated for a small number of data points since the mean converges with a small number of samples as shown in Fig. 12. For IC, the estimated mean for a sample size of ≈ 1000 corresponds to 5% deviation from the true IC.

Appendix B: Additional figures for model architectures details and unfiltered plots

All DL models used for our experiments are Conv-LSTM models comprising Convolutional layers placed sequentially. We used a shorthand notation to denote the models by mentioning the number of Convolutional filters (FIL) and the depth of the model (DEP). Here, in Fig. 13 we present the detailed architectures for the interested reader. The special case of the architecture f_{def} is also shown.



(a) Model with depth 4 and filters 128
f-DEP-4-FIL-128

(b) Model with depth 2 and filters 16
f-DEP-2-FIL-16

(c) Model with depth 4 and decreasing filter sizes used for f_{def} use case experiment across multiple tasks

Fig. 13: Exact architectures for different models used in various experiments. **Fig. (a)** and **Fig. (b)** represent two of the 9 combinations of f-DEP-FIL where DEP $\in \{1, 2, 4\}$ and FIL $\in \{16, 64, 128\}$. **Fig. (c)** represents f_{def}: the exact architecture used for the IC validation experiment in subsection 6.3

The Fig. 15 shows the unfiltered plot when comparing f-DEP-4-FIL-64 with and without Batch Normalisation Layers. In the main manuscript, we filtered this figure by showing only the epochs which performed better than all previous epochs. Finally, the Fig. 14 shows how the number of parameters differs significantly from the true model complexity (MC), even when trained on the same dataset.

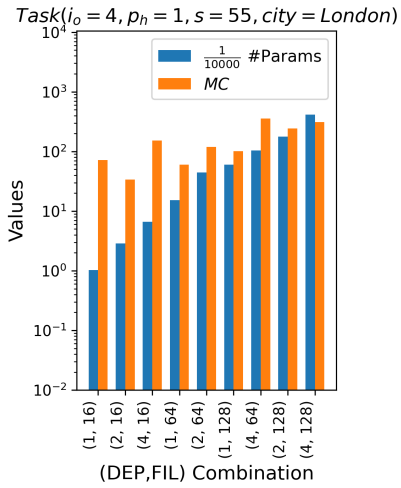


Fig. 14: Comparison of the number of trainable parameters to the model complexity (MC) when different models are trained on the same dataset. The models are sorted in ascending order based on their number of parameters

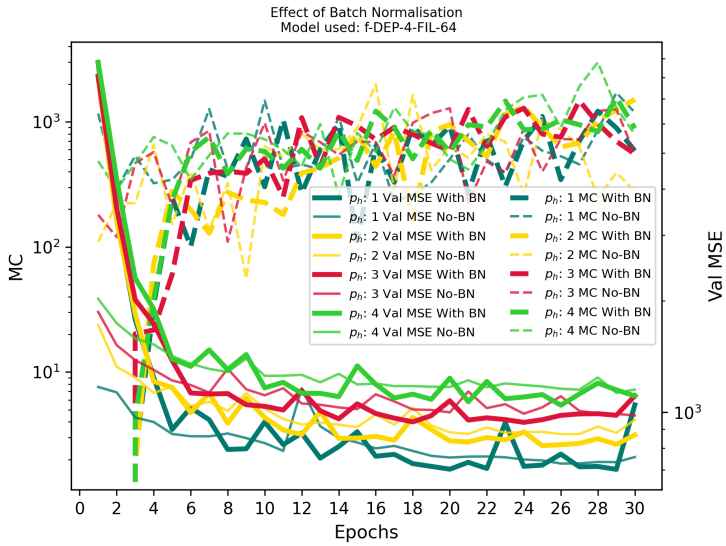
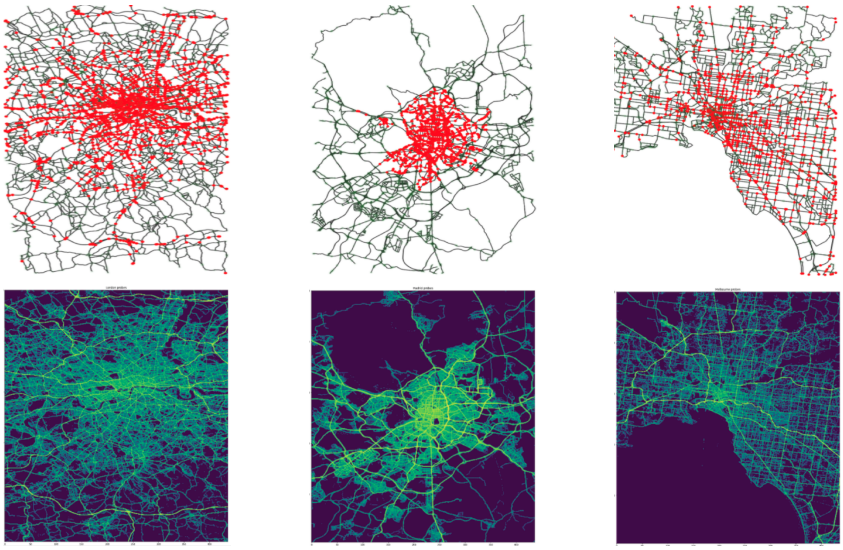


Fig. 15: Validation MSE and MC for 30 epochs. Unfiltered plot without any smoothing

Appendix C: Sensor locations and exact sizes of urban pixels using HERE traffic data

The vehicle loop detector (VLD) sensor locations are shown in Fig. 16 and the true aspect ratios of urban heatmaps are shown in Fig. 17. Finally, Table 9 summarises the exact dimensions of the urban tile for each city at three scales.



(a) VLD locations in London

(b) VLD locations in Madrid

(c) VLD locations in Melbourne

Fig. 16: Distribution of VLDs across the road network in various cities. Image reproduced with permission from data providers

Table 9: Dimensions of urban pixels at various scales for three cities

City	Scale (s)	Height (km)	Width (km)
London	25	1.97	1.20
	55	0.89	0.55
	105	0.47	0.29
Madrid	25	0.82	0.87
	55	0.37	0.39
	105	0.19	0.21
Melbourne	25	2.19	1.53
	55	0.99	0.70
	105	0.52	0.37

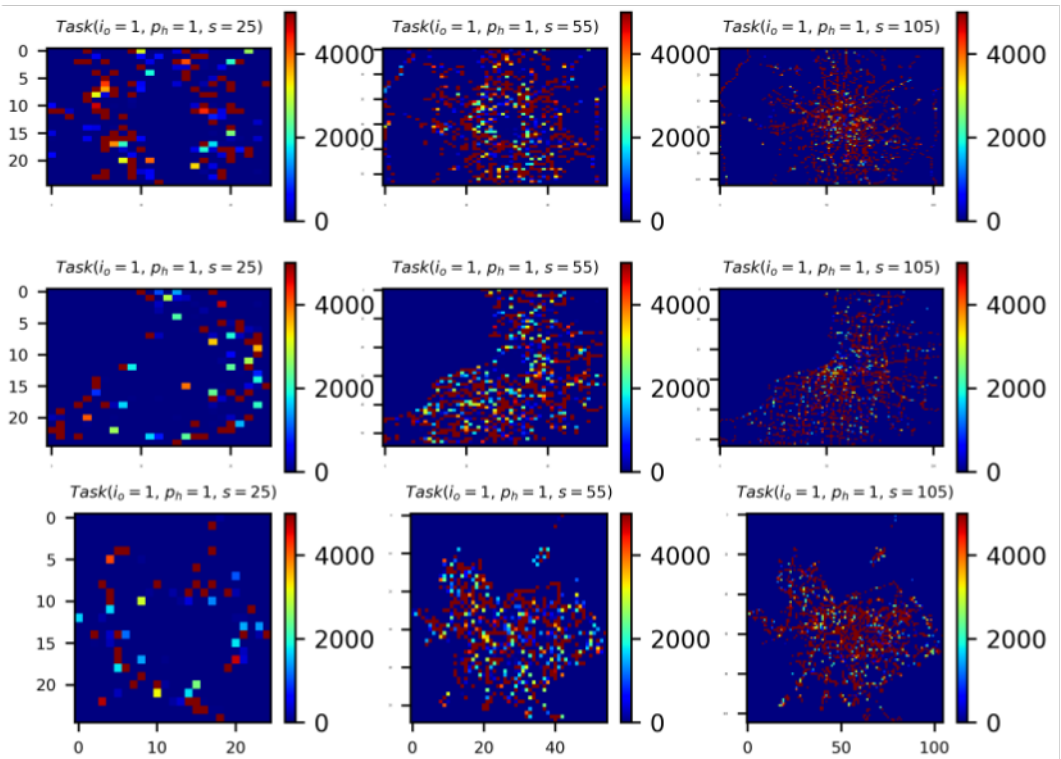


Fig. 17: Aggregated values of Traffic Volume post binning for all three cities assuming three scales: $s = 25$, $s = 55$, and $s = 105$