Theoretical Tuning of the Autoencoder Bottleneck Layer Dimension: A Mutual Information-based Algorithm

Guillem Boquet, Edwar Macias, Antoni Morell, Javier Serrano and Jose Lopez Vicario
Wireless Information Networking (WIN) Group
Universitat Autònoma de Barcelona (UAB)
Email: {guillem.boquet, edwar.macias, antoni.morell, javier.serrano, jose.vicario}@uab.cat

***Abstract*—Under the transportation field, the literature states that forecasting with excessive number of features can be computational inefficient and undertakes the risk of over-fitting. Because of that, several authors proposed the use of autoencoders (AE) as a way of learning fewer but useful features to enhance the road traffic forecast. Notably, the adequacy of the bottleneck layer dimension of the AE has not been addressed, thus there is no standard way for automatic selection of the dimensionality. We address the problem from an information theory perspective as the reconstruction error is not a reliable indicator of the performance of the subsequent supervised learning algorithm. Hence, we propose an algorithm based on how mutual information and entropy of data evolve during training of the AE. We validate it against two real-world traffic datasets and provide discussion why the entropy of codes is a reliable performance indicator. Compared to the tendency found in the literature, based on trial-and-error methods, the advantage of our proposal is that a practitioner can efficiently find said dimension guaranteeing maximal data compression and reliable traffic forecast.**

***Index Terms*—intelligent transportation systems, traffic forecasting, autoencoder, mutual information, entropy**

I. Introduction

In the road traffic forecasting domain, several challenges remain unsolved [1] despite the effort that researches have put into developing robust forecasting techniques to reduce and manage the risk associated with traffic and to successfully deploy intelligent transportation systems (ITS). Nowadays, in the era of big data, the number of features available from data sources along with the number of available data points in a road traffic network are growing excessively. Forecasting with all those features can be computational inefficient and undertakes the risk of over-fitting. Thus, is essential to reduce the dimension of the feature space before applying a prediction model [2]. Reduction of the dimensions is done by learning the principal components or independent factors of a given data manifold, i.e., feature extraction [3]. Traditionally, low-dimensional representation is obtained by PCA approaches. Besides, the least absolute shrinkage and selection operator (LASSO) is a well-known technique used for feature selection [4]. However, data-based approaches such as deep neural networks (DNN) have become increasingly relevant as current technologies facilitate access to dynamic data and big data. Within this field, features learned by an autoencoder (AE) have been proved in the literature to improve the traffic forecast [5]–[10]. The AE is trained to reconstruct its input through a bottleneck layer with fewer dimensions than the data space. That is, the AE first encodes the input to a hidden representation (or code) of lower dimensions and then decodes it back into a reconstruction. Under AE-based traffic forecast, the encoder is used as input for a regression network which is then trained in a supervised manner. Sometimes, the weights and biases of the encoder are not updated during the supervised training, while other times the entire network is fine-tuned. The latter may be useful for increasing the accuracy of the forecast, but will no longer serve as a non-linear data compression tool as the updated encoder will not match the decoder. In both cases, the dimensionality of the bottleneck layer is crucial and affects the performance of the forecasting system: a too large size may lead to redundant dimensions and high computational cost and a too small size might lead to high information loss. Despite that, in the traffic forecasting domain, authors of [5]–[10] estimate said dimensionality arbitrarily or by trial-and-error methods such as grid search, leaving aside the importance of data compression in the era of big data.

To the best of the author's knowledge, the adequacy of the bottleneck dimension is rarely addressed in the literature and there is no standard way for automatic selection of the dimensionality. We propose an automatic algorithm to estimate the dimensionality to achieve maximal traffic data compression without compromising the performance of the subsequent traffic forecast. Gupta et al. [11] verified experimentally that the reconstruction error is not a reliable indicator of the performance of the final application. Hence, we based our algorithm on a novel and efficient estimator of the entropy and mutual information (MI) and recent advances on the interpretability of the AE from an information theory perspective [12]. Compared to the available approaches, we highly reduce the number of models and training time based on the entropy of the resulting codes as a key performance indicator (KPI) of the traffic forecasting system. In other words, the advantage is that there is no need to train any regression network because the entropy of codes works as a KPI. Additionally, the AE just needs to be trained for fewer epochs because the information quantities converge faster.

II. Methodology

Let $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{n}$ be a batch of traffic data composed of n *traffic samples*. Let $\mathbf{x}_i \in \mathbb{R}^{t \times s}$ be a *traffic sample* where each element represents a real-valued traffic variable (e.g., speed), s

This research is supported by the Catalan Government under Project 2017 SGR 1670 and the Spanish Government under Project TEC2017-84321-C4-4-R co-funded with European Union ERDF funds.

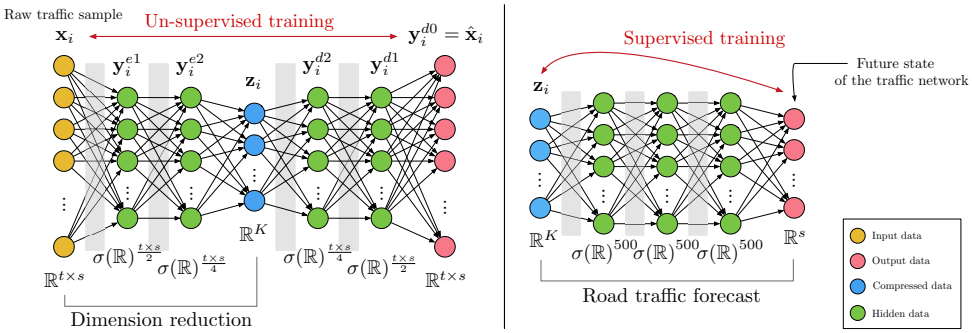

Fig. 1: AE-based traffic forecast. First, given the *traffic sample* $\mathbf{x}_i$, the AE (left) seeks to map it into a lower-dimensional space prior to running the supervised learning algorithm. Later, the compressed sample is used to train a regression model (right) to forecast the future state of the road traffic network. Weights and biases are represented by arrows. Activation functions σ are represented with filled rectangles between node layers. Data space dimensions are shown below each layer.

the number of traffic sensors in the network and t the number of past measures. The purpose of the AE is to enforce the output (i.e., $\hat{\mathbf{X}}$) equal to $\mathbf{X}$ with high fidelity by minimizing the squared reconstruction error $||\mathbf{X} - \hat{\mathbf{X}}||_2^2$. For simplicity but without loss of generality, consider the AE architecture shown in Fig. 1 which is commonly used to forecast traffic. Given its symmetry, we have $dim(\mathbf{z}_i) = K < dim(\mathbf{y}_i^{e2}) = dim(\mathbf{y}_i^{d2}) = dim(\mathbf{y}_i^{e1})/2 = dim(\mathbf{y}_i^{d1})/2 = dim(\mathbf{x}_i)/4$, where superscripts e and d denote encoder and decoder layers, respectively. Because we are interested in the information quantities of the data, we denote the batch of data represented by each layer as $\{\mathbf{X}, \mathbf{Y}^{e1}, \mathbf{Y}^{e2}, \mathbf{Z}, \mathbf{Y}^{d2}, \mathbf{Y}^{d1}, \hat{\mathbf{X}}\}$ where $\mathbf{X}$ is the input data, $\hat{\mathbf{X}}$ its reconstruction, $\mathbf{Z}$ the hidden representations (or codes) by the bottleneck layer and $\mathbf{Y}$ the activations of each hidden layer. Moreover, we assume the existence of an *adequate* dimension D of the given data that when $K < D$ the compressed data will not provide the regression network of enough information to provide a reliable forecast.

A. Estimation of mutual information from mini-batches

Consider $\mathbf{X}$ and $\mathbf{Y}$ as two random variables with a joint probability density function (PDF) $p(\mathbf{x}, \mathbf{y})$ and marginals $p(\mathbf{x})$ and $p(\mathbf{y})$. The Rényi's entropy of order α of $\mathbf{X}$ is defined as

$$H_\alpha(\mathbf{X}) = \frac{1}{1-\alpha} \log \int_{\mathbb{R}} p^\alpha(\mathbf{x}) dx, \tag{1}$$

where $\alpha \geq 0$ and $\alpha \neq 1$. When $\alpha \to 1$, (1) is defined in the limit and yields Shannon's differential entropy. Then, the mutual information $I(\mathbf{X}; \mathbf{Y})$ is defined as the relative entropy between the joint distribution and the product distribution $p(\mathbf{x})p(\mathbf{y})$, which can be expressed as

$$I(\mathbf{X}; \mathbf{Y}) = H(\mathbf{X}) + H(\mathbf{Y}) - H(\mathbf{X}, \mathbf{Y}), \tag{2}$$

where $H(\mathbf{X})$ and $H(\mathbf{Y})$ are the marginal entropies of $\mathbf{X}$ and $\mathbf{Y}$ and $H(\mathbf{X}, \mathbf{Y})$ is their joint entropy. In DNN traffic forecasting, the analytical evaluation of (2) is not possible because (1) requires precise PDF estimation of $\mathbf{X}$ and $\mathbf{Y}$ in high-dimensional space and, therefore, one is forced to estimate its value from a limited number of samples, i.e., mini-batches.

Given the batch $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$, an iid sample of n realizations of $\mathbf{X}$. The Gram matrix $\mathbf{K}$ is obtained from evaluating a real valued positive definite kernel $\kappa : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}$ on all pairs of data points such that $(\mathbf{K})_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$. Then, a matrix-based analogue to Rényi's α-entropy (1) is defined in [13] for a matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ that holds $tr(\mathbf{A}) = 1$ as

$$S_\alpha(\mathbf{A}) = \frac{1}{1-\alpha} \log \left[\sum_{i=1}^n \lambda_i(\mathbf{A})^\alpha \right], \tag{3}$$

where $\lambda_i(\mathbf{A})$ denotes the i^{th} eigenvalue of $\mathbf{A}$, the normalized version of $\mathbf{K}$, that is, $\mathbf{A}_{ij} = \frac{1}{n} \frac{\mathbf{K}_{ij}}{\sqrt{\mathbf{K}_{ii}\mathbf{K}_{jj}}}$. Furthermore, a matrix-based estimation of the joint entropy [13] can be defined as

$$S_\alpha(\mathbf{A}, \mathbf{B}) = S_\alpha \left(\frac{\mathbf{A} \odot \mathbf{B}}{tr(\mathbf{A} \odot \mathbf{B})} \right), \tag{4}$$

where $\odot$ denotes the Hadamard product and the matrix $\mathbf{B}$ is obtained analogously to $\mathbf{A}$, but given $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^n$ samples of the targeted layer from the same realizations of $\mathbf{X}$. Notice that matrices $\mathbf{A}$ and $\mathbf{B}$ simplifies the estimation of the joint distribution to pairwise element multiplication. Finally, the matrix-based Rényi's mutual information is defined as

$$I_\alpha(\mathbf{A}; \mathbf{B}) = S_\alpha(\mathbf{A}) + S_\alpha(\mathbf{B}) - S_\alpha(\mathbf{A}, \mathbf{B}), \tag{5}$$

in analogy with Shannon's definition (2) [13].

B. How information flows during training

Training of the AE is done via back-propagation and stochastic gradient descent (SGD). Both feedforward and backward passes are unidirectional and only depend upon the previous variables, thus forming a Markov chain [14]. Given the AE symmetry and that the output resembles the input when well-trained, it makes sense to divide the chain into two dual Markov processes, $\mathbf{X} \to \mathbf{Y}^{e1} \to \mathbf{Y}^{e2} \to \mathbf{Z}$ and $\mathbf{Z} \to \mathbf{Y}^{d2} \to \mathbf{Y}^{d1} \to \hat{\mathbf{X}}$. Under this framework, the following data processing inequalities (DPI) hold:

1) $I(\mathbf{X}; \mathbf{Y}^{e1}) \geq I(\mathbf{X}; \mathbf{Y}^{e2}) \geq I(\mathbf{X}; \mathbf{Z})$ and $I(\hat{\mathbf{X}}; \mathbf{Y}^{d1}) \geq I(\hat{\mathbf{X}}; \mathbf{Y}^{d2}) \geq I(\hat{\mathbf{X}}; \mathbf{Z})$,
2) $I(\mathbf{X}; \hat{\mathbf{X}}) \geq I(\mathbf{Y}^{e1}; \mathbf{Y}^{d1}) \geq I(\mathbf{Y}^{e2}; \mathbf{Y}^{d2}) \geq I(\mathbf{Z}; \mathbf{Z})$,

thus the role of the AE is to maximize the entropy in the hidden layers [12]. Note that $I(\mathbf{Z}; \mathbf{Z}) = H(\mathbf{Z})$, Fig. 1.

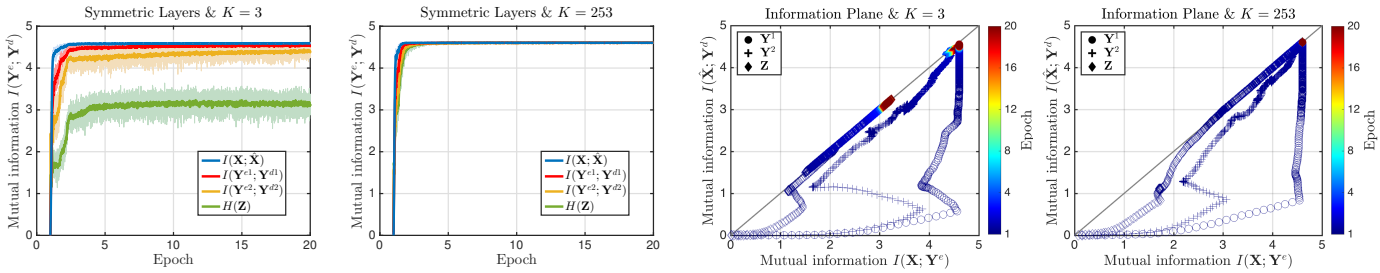

Fig. 2: Information flow during training, Sec. II-B. Information quantities were maximized towards $H(\mathbf{X}) \simeq 4.6$ *nats* at each layer when $K = 253$, contrary to $K = 3$, due to K being larger or lower than the intrinsic dimensionality of the data [12]. All metrics (averaged with a moving average of window size 20 to smooth the results) converged faster before epoch 20, however, MSE converged at around epoch 300. The AE was trained to reconstruct *traffic samples* of PeMS traffic speed data, Sec. III.

We can verify the DPIs by (*i*) modeling during training the information that each hidden layer in the encoder preserves about the input $I(\mathbf{X}; \mathbf{Y}^e)$ with respect to the information that decoder layers preserve on the output $I(\hat{\mathbf{X}}; \mathbf{Y}^d)$ —this two quantities span what is referred to as an Information Plane (IP) [15]— and by (*ii*) modeling the MI between each symmetric layer. Two types of plots that verify the DPIs are shown in Fig. 2, the evolution of said IP and the MI of symmetric layers computed at each training update. The IP shows that all metrics converge to the line $x = y$, where the optimal solution resides when MSE is minimized because $\mathbf{X} = \hat{\mathbf{X}}$. The MI between the input and higher layers is always greater than the MI between the input and lower layers. The training procedure begins by first maximizing the amounts of information in the encoder layers rather than in the decoder. All quantities are maximized as the AE trains on more epochs, converging to $H(\mathbf{X})$. Similarly, the MI of symmetric layers increase to converge to $I(\mathbf{X}; \hat{\mathbf{X}})$, i.e., $H(\mathbf{Z})$ is maximized.

The most interesting behavior for what concerns us is that both type of plots show a different behavior when K is too low, similarly to what [12] found on the MNIST dataset. Specifically, that plots of Fig. 2 when $K = 3$ show that neither $H(\mathbf{Z})$, $I(\mathbf{X}; \mathbf{Z})$ nor $I(\hat{\mathbf{X}}; \mathbf{Z})$ are able to converge to $H(\mathbf{X})$.

C. Looking for the adequate bottleneck dimension

Based on a similar MI behaviour, [12] proposed an algorithm to determine D but with human observation, i.e., not applicable in practice. To solve that, we propose an automatic algorithm based only on the evolution of $H(\mathbf{Z})$. We propose to look for D from an array containing an effective searching range of dimensions, where the lower bound is limited to 1 and the upper bound is limited by the AE architecture, i.e. $(t \times s)/4$. The algorithm proposed consists of searching the sorted array by repeatedly dividing the search interval in half (Algorithm 1) and check if $H(\mathbf{Z}) \simeq H(\mathbf{X})$ holds at each array division when training the AE (Algorithm 2). We provided discussion on why $H(\mathbf{Z})$ does work as a KPI in Sec. III-E, jointly with the results obtained.

In summary, we begin to mini-batch train a new AE with bottleneck layer size given by the dimension K of the middle position of the searching array. At each training update, we use the updated AE to project the data trough each layer and estimate the MI between the input and output data $I_\alpha(\mathbf{X}; \hat{\mathbf{X}})$ using (5) and the entropy of codes $S_\alpha(\mathbf{Z})$ using (3). For our implementation, we trimmed $I_\alpha(\mathbf{X}; \hat{\mathbf{X}})$ to two decimals and, thus, we checked for the condition $S_\alpha(\mathbf{Z}) \simeq I_\alpha(\mathbf{X}; \hat{\mathbf{X}})$ at the end of the epoch. Since the metrics are computed at the mini-batch level, to smooth and speed up computation we estimated the MI and entropy only every 10 mini-batches and averaged it accordingly at the end of each epoch. Note that both procedures are omitted in Algorithm 2 to avoid clutter. Then, we narrow the interval to the lower half if the condition held. Otherwise, we narrow it to the upper half. Finally, we repeatedly check until the interval is empty to find D.

Algorithm 1 Estimation of the bottleneck layer dimension.

Require: Traffic data $\mathbf{X}$, dimension searching range $\boldsymbol{dim}$
$L \leftarrow 0$
$R \leftarrow length(\boldsymbol{dim}) - 1$
while $L \leq R$ **do**
  $D \leftarrow ceil((L + R)/2))$
  $C \leftarrow Algorithm2(\mathbf{X}, \boldsymbol{dim}[D])$
  if C **then**
    $R \leftarrow D - 1$
  else
    $L \leftarrow D + 1$
  end if
end while
return D

Algorithm 2 Condition check on $H(\mathbf{Z})$ while training the AE.

Require: Data $\mathbf{X}$, bottleneck dimension K
$\mathbf{X}_{train}$, $\mathbf{X}_{valid} \leftarrow \mathbf{X}$
$autoencoder \leftarrow getAutoEncoder(K)$
$nb \leftarrow length(\mathbf{X}_{train})/batchsize$
for $epoch \leftarrow 0$ to 1714 **do**
  $\mathbf{X}_{train} \leftarrow randomShuffle(\mathbf{X}_{train})$
  for $batch \leftarrow 0$ to nb **do**
    $\mathbf{X}_b \leftarrow getBatch(\mathbf{X}_{train})$
    $\hat{\mathbf{X}}_b$, $\mathbf{Z}_b \leftarrow autoencoder.train(\mathbf{X}_b)$
    $I(\mathbf{X}_b; \hat{\mathbf{X}}_b)$, $S(\mathbf{Z}_b) \leftarrow getMetrics(\mathbf{X}_b, \mathbf{Z}_b, \hat{\mathbf{X}}_b)$
    $batch \leftarrow batch + 1$
  end for
  $\overline{I}_X$, $\overline{S}_Z \leftarrow getAverage(I(\mathbf{X}_b; \hat{\mathbf{X}}_b), S(\mathbf{Z}_b), nb)$
  if $\overline{S}_Z \simeq \overline{I}_X$ **then**
    return 1
  else if $earslyStop(\overline{I}_X, \overline{S}_Z, \mathbf{X}_{train}, \mathbf{X}_{valid})$ **then**
    return 0
  end if
  $epoch \leftarrow epoch + 1$
end for
return 0

Algorithm 1 runs in logarithmic time in the worst case making $O(\log(length(\boldsymbol{dim})))$ runs of Algorithm 2. To trim the searching range, we applied PCA to the training data and set the upper bound as the number of dimensions that explained the 90% of the variance because it is known that non linear AE have higher modeling capabilities than PCA. Furthermore, we used evenly spaced values within an interval of 2 because we found no critical differences on the prediction accuracy from consecutive dimensions. To compensate for that, we returned $D + 1$ in Algorithm 1.

III. Experimentation

To validate the algorithm, we considered the scenario outlined in Fig. 1 for long-term traffic forecasting using the AE-approach. Then, we provided results on two real-world speed traffic datasets and discussion on $H(\mathbf{Z})$ as a KPI.

A. Brief traffic data description

PeMS. The dataset consist of 5-minute aggregated speed data from 31 loop detectors installed on a south-bound section of Interstate 5 (I-5). Detectors span spaced equally apart 82 km of the highway from post mile (PM) 1.1 to 52.3. Data collected covers the two-year period from 2015 until 2017 [9]. Testing was done on 2016 data while the rest was used for training.

UKM4. The dataset consist of traffic speed data from 19 junctions (J22 to J2) of the English M4 section from Bristol to London [9]. The data are averaged between junctions and aggregated into 15-minute intervals. Junctions span 180 km and consist of different road lengths. Data collected covers the four-year period from 2011 until 2015. The last year of data was used for testing.

B. Hyper-parameter and training details

MI estimation. We set $\alpha = 2$, which provides neutral weighting [16]. We used the already normalized radial basis function (RBF) kernel, $\kappa_G(\mathbf{x}_i, \mathbf{x}_j; \sigma) = \exp\left(\frac{1}{2\sigma^2}||\mathbf{x}_i - \mathbf{x}_j||_2^2\right)$. The *bandwidth* σ of the kernel is a free parameter that needs to be defined because it strongly influences the estimate obtained. We followed Silverman's rule of thumb for Gaussian density estimation $\hat{\sigma} = h\, n^{-1/(4+d)}$, where h is the standard deviation of the samples, n is the sample size and d is the sample dimensionality.

AE model. We trained the AE to reconstruct its normalized input by minimizing the MSE while varying K. Once trained, we compressed the data using only the encoder part to obtain the training data for the forecasting model.

Forecasting model. We set the regression network of Fig. 1 to estimate 1 hour ahead of traffic speed of the whole network ($s = 31$ for PeMS and $s = 19$ for UKM4) using the last three hours of data ($t = 36$ for PeMS and $t = 18$ for UKM4). Note that for supervised training $\mathbf{z}_i = \mathbf{x}_i \in \mathbb{R}^{t \times s}$ when raw data is used and $\mathbf{z}_i \in \mathbb{R}^K$ when compressed data from the AE is used. We added two *dropout* layers after the first two hidden layers with a drop rate of 0.5 to avoid overfitting. Training was done by minimizing the MSE between predictions and normalized labels. We trained both models using SGD with *Adam* optimizer and batch size of 100. We set $\sigma(a) = \frac{1}{1+e^{-a}}$. No hyper-parameter optimization was done. To end the training procedure, we used *earlystopping* with a patience of 30 on a 10% random split of the training data.

C. Evaluation metrics

As benchmark, we computed the root mean squared error (RMSE) of the forecasting model trained with raw data, $RMSE_{bm}$. Then, we defined the $RMSE_{increase}$ as the increment in percentage that suffers the RMSE of the estimation when the network is trained using compressed data with respect to $RMSE_{bm}$, that is,

$$RMSE_{increase}\,[\%] = \frac{RMSE - RMSE_{bm}}{RMSE_{bm}} \times 100.$$

Furthermore, we defined the data compression ratio (DC) as the percentage of reduction that the uncompressed data suffers while using a concrete bottleneck dimension K,

$$DC\,[\%] = \frac{(t \times s) - K}{t \times s} \times 100.$$

By doing so, we can compare the results obtained more efficiently against different datasets with different number of road sensors available. Each experiment was conducted 10 times and we showed the average RMSE, omitting the irrelevant variance of the results for clarity.

D. Results

We trained the AE varying down the size of the bottleneck layer from $K = dim(\mathbf{X})/4$ to $K = 3$ with step-sizes of 10, which took 2 days to compute. Fig. 3 shows the accuracy of the forecast for different sizes of bottleneck layer. The naive forecast consisted on setting the last speed values measured as the estimation. The results of $RMSE_{bm}$ for PeMS and UKM4 data were 7.81 km/h and 11.02 km/h, respectively. Also, accuracy on the well-known MNIST dataset is shown. The adequate dimensions given by our algorithm were $D = 29$ and $D = 15$, estimated in 64 and 11 minutes for PeMS and UKM4, respectively. For MNIST, $D = 19$ was estimated in 6 minutes. The little time needed to find D is clearly an advantage over current trial-and-error methods. Because the first phase of the training [14] is dedicated to bring $I(\mathbf{X}; \hat{\mathbf{X}})$ closer as possible to $H(\mathbf{X})$, one can check the condition in the first stages of the training without waiting for the AE to be fully trained, that is, the MSE to converge, making this criterion more efficient in practice. As it becomes evident from Fig. 3, the RMSE and accuracy exhibited a clear elbow region pattern. The adequate dimensions coincided in the elbow region below the point where the RMSE of the forecast changes its behavior. For $K > D$, that is, higher data compression ratios, the forecast error showed a quasi-linear behavior with almost constant slope. This explains why AE road traffic forecasting literature choose K arbitrarily, as long as K is not very small, and achieved good forecasting performance. For both traffic datasets, D show a similar $RMSE_{increase}$ around 5% despite compressing the input data more than 95%. The data compression ratios achieved depend on the complexity of the road traffic network and data. The 5% degradation on

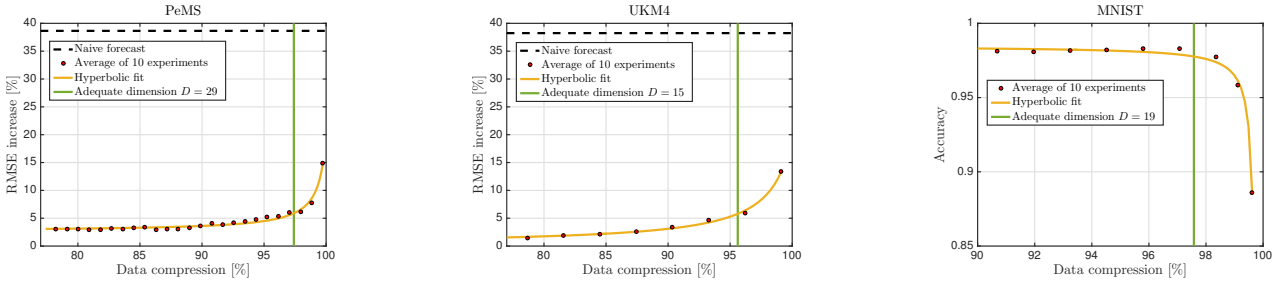

Fig. 3: RMSE increase of traffic forecast vs. data compression ratio, i.e, different values of the bottleneck layer dimension K. The adequate dimension D coinciding in the elbow region is efficiently obtained applying the algorithm proposed in Sec. II-C.

performance can be explained because we trimmed $I_2(\mathbf{X};\hat{\mathbf{X}})$ to two decimals and due to the lost of spatial relations in code space. This results validate our assumptions, showing that the entropy of codes is a KPI of the forecasting system.

E. Discussion on why does $H(\mathbf{Z})$ work as a KPI

When minimizing the MSE, we checked during training if the condition $H(\mathbf{Z}) \simeq H(\mathbf{X})$ held. If the condition did not hold, that is, $H(\mathbf{Z}) < H(\mathbf{X})$, we obtained a degradation on the performance of the regression network when training using that $\mathbf{Z}$. Therefore, we proposed an efficient algorithm that estimates D, the lower dimension that holds $H(\mathbf{Z}) = H(\mathbf{X})$, that is, the maximum compression of data that retain the same information than the original data. Results showed that the entropy of codes always increased if the bottleneck layer dimension was increased, with the upper bound being the entropy of the input or the MI between the input and output when the AE is well-trained. Furthermore, the performance of the subsequent traffic forecast suffered when the entropy of codes was lower than $I(\mathbf{X};\hat{\mathbf{X}})$ or $H(\mathbf{X})$ because of the reduction of information. Given that the forecasting network is usually trained minimizing the MSE using $\mathbf{X}$ with $H(\mathbf{X}) > 0$, we argue that the same network will be able to learn and have similar performance if we provide it with a $\mathbf{Z}$ with $dim(\mathbf{Z}) < dim(\mathbf{X})$ and $H(\mathbf{Z}) \simeq H(\mathbf{X})$ to learn from. In other words, when the AE is well-trained in terms of generalization to reconstruct its input through a subspace, $\mathbf{Z}$ must contain useful traffic characteristics that define $\mathbf{X}$. Otherwise, the AE would not be able to reconstruct the $\mathbf{X}$ faithfully. Thus, the minimization of MSE guarantees that useful traffic characteristics are learned for traffic forecasting. At the same time, the maximization of $H(\mathbf{Z})$ guarantees that $\mathbf{Z}$ has the same capacity in terms of information than $\mathbf{X}$. In that sense, we left further theoretical proof as future work.

IV. Conclusion

Trial-and-error methods are the standard way to automatic select the dimensionality of the bottleneck layer dimension of AE in road traffic forecasting. We explored the AE training dynamics from an information theory perspective, monitoring the flow of mutual information and entropy of traffic data of each layer at mini-batch level. We proposed and validated experimentally an algorithm to estimate the size of the bottleneck layer to achieve reliable forecasting performance and maximal data compression. Finally, we discussed why the entropy of codes is a reliable KPI of the subsequent traffic forecasting network and left theoretical proof as future work.

References

[1] I. Lana, J. Del Ser, M. Velez, and E. I. Vlahogianni, "Road traffic forecasting: Recent advances and new challenges," *IEEE Intelligent Transportation Systems Magazine*, vol. 10, no. 2, pp. 93–109, 2018.

[2] S. Yang and S. Qian, "Understanding and predicting travel time with spatio-temporal features of network traffic flow, weather and incidents," *arXiv preprint arXiv:1901.06766*, 2019.

[3] D. Pavlyuk, "Feature selection and extraction in spatiotemporal traffic forecasting: a systematic literature review," *European Transport Research Review*, vol. 11, no. 1, p. 6, 2019.

[4] N. G. Polson and V. O. Sokolov, "Deep learning for short-term traffic flow prediction," *Transportation Research Part C: Emerging Technologies*, vol. 79, pp. 1–17, 2017.

[5] Y. Lv, Y. Duan, W. Kang, Z. Li, F.-Y. Wang *et al.*, "Traffic flow prediction with big data: A deep learning approach." *IEEE Trans. Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2015.

[6] H.-F. Yang, T. S. Dillon, and Y.-P. P. Chen, "Optimized structure of the traffic flow forecasting model with a deep learning approach," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2371–2381, 2016.

[7] T. Zhou, G. Han, X. Xu, Z. Lin, C. Han, Y. Huang, and J. Qin, "δ-agree adaboost stacked autoencoder for short-term traffic flow forecasting," *Neurocomputing*, vol. 247, pp. 31–38, 2017.

[8] R. Yu, Y. Li, C. Shahabi, U. Demiryurek, and Y. Liu, "Deep learning: A generic approach for extreme condition traffic forecasting," in *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 2017, pp. 777–785.

[9] G. Boquet, J. L. Vicario, A. Morell, and J. Serrano, "Missing data in traffic estimation: A variational autoencoder imputation method," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 2882–2886.

[10] S. Zhang, Y. Yao, J. Hu, Y. Zhao, S. Li, and J. Hu, "Deep autoencoder neural networks for short-term traffic congestion prediction of transportation networks," *Sensors*, vol. 19, no. 10, p. 2229, 2019.

[11] P. Gupta, R. E. Banchs, and P. Rosso, "Squeezing bottlenecks: exploring the limits of autoencoder semantic representation capabilities," *Neurocomputing*, vol. 175, pp. 1001–1008, 2016.

[12] S. Yu and J. C. Principe, "Understanding autoencoders with information theoretic concepts," *Neural Networks*, vol. 117, pp. 104–123, 2019.

[13] L. G. S. Giraldo, M. Rao, and J. C. Principe, "Measures of entropy from data using infinitely divisible kernels," *IEEE Transactions on Information Theory*, vol. 61, no. 1, pp. 535–548, 2014.

[14] R. Shwartz-Ziv and N. Tishby, "Opening the black box of deep neural networks via information," *arXiv preprint arXiv:1703.00810*, 2017.

[15] N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," in *2015 IEEE Information Theory Workshop (ITW)*. IEEE, 2015, pp. 1–5.

[16] S. Yu, L. G. S. Giraldo, R. Jenssen, and J. C. Principe, "Multivariate extension of matrix-based renyi's α-order entropy functional," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.