Accepted to Proc. IEEE Globecom 2023

Dynamic Encoding and Decoding of Information for Split Learning in Mobile-Edge Computing: Leveraging Information Bottleneck Theory

Omar Alhussein[†]*, Moshi Wei[‡], Arashmid Akhavain*
[†]Department of Electrical Engineering and Computer Science, Khalifa University, Abu Dhabi, UAE
*Advanced Networking Team, Huawei Ottawa Research & Development Centre, Ottawa, Canada
[‡]Department of Computer Science, York University, Toronto, Canada
omar.alhussein@ku.ac.ae, arashmid.akhavain@huawei.com, moshiwei@yorku.ca

***Abstract*—Split learning is a privacy-preserving distributed learning paradigm in which an ML model (e.g., a neural network) is split into two parts (i.e., an encoder and a decoder). The encoder shares so-called latent representation, rather than raw data, for model training. In mobile-edge computing, network functions (such as traffic forecasting) can be trained via split learning where an encoder resides in a user equipment (UE) and a decoder resides in the edge network. Based on the data processing inequality and the information bottleneck (IB) theory, we present a new framework and training mechanism to enable a dynamic balancing of the transmission resource consumption with the informativeness of the shared latent representations, which directly impacts the predictive performance. The proposed training mechanism offers an encoder-decoder neural network architecture featuring multiple modes of complexity-relevance tradeoffs, enabling tunable performance. The adaptability can accommodate varying real-time network conditions and application requirements, potentially reducing operational expenditure and enhancing network agility. As a proof of concept, we apply the training mechanism to a millimeter-wave (mmWave)-enabled throughput prediction problem. We also offer new insights and highlight some challenges related to recurrent neural networks from the perspective of the IB theory. Interestingly, we find a compression phenomenon across the temporal domain of the sequential model, in addition to the compression phase that occurs with the number of training epochs.**

***Index Terms*—information bottleneck, NFV, semantic communications, split learning, wireless edge learning**

I. Introduction

The dawn of the sixth-generation (6G) era is to bring forth a new paradigm in communication networks, driven by the increasing demand for ultra-fast, intelligent, and pervasive connectivity. As the digital revolution permeates every aspect of our lives, there is an even greater need to increase networking efficiency, enhance network automation, and embed native intelligence. In 1949, Shannon and Weaver categorized the problem of communications into three levels, namely (*i*) the technical: transmission of symbols; (*ii*) the semantic: transmission of meaning; and (*iii*) the effectiveness: effect of semantic information exchange. While Shannon's communication model considers the technical aspect only, there is a growing interest in re-examining this fundamental consideration and incorporate semantics to the 6G fabric.

[§]This work was conducted primarily at Huawei Ottawa Research & Development Centre.

Enabled by network softwarization and function virtualization, mobile-edge computing rely on various network functions to optimize the network's resources. Existing network functions, such as traffic forecasting, traffic classification, packet scheduling, are increasingly being implemented using experience-driven model-free machine learning (ML) techniques. There is also an emergence of new functions related to real-time sensing and analytics such as object detection and tracking. Also, with 5G's service-based architecture, (over-the-top) application functions can be hosted natively by service providers in core networks, e.g., to enable virtual reality and close-proximity gaming. Given the increasing adoption and diversity of network and application functions, there is a need to optimize the communication and processing resources to support and automate such network and application functions.

Hardware technology, as predicted by Moore's law, has been unable to keep up with the exponential growth of computational and storage requirements for modern ML models, resulting in a concerning widening gap [1]. New distributed learning paradigms are needed to efficiently utilize available resources while maintaining privacy and security. Split learning has emerged as a distributed learning approach that offers a unique combination of benefits for the wireless edge in terms of efficiency, privacy, and flexibility [2], [3].

In this paper, we identify and address a limitation with existing split learning based predictive network and application functions; the network substrate experiences time-varying usage behaviour and traffic patterns mainly due to the time-varying and random behavior of users. Moreover, network and application functions need to cater towards diverse quality of service and predictive requirements. We need to adapt the informativeness of encoded data in a dynamic manner based on network conditions and application requirements to ensure that the network remains communication-efficient and flexible under a wide range of scenarios. The information bottleneck (IB) method, originally proposed by Tishby, Pereira, and Bialek, can provide a new pair of lens for analyzing and improving deep learning models [4], [5]. The IB method attempts to find the best tradeoff between the compression with regard to input data and the preservation of relevant information needed for a specific task based on the input. See Sections II and III for a continued discussion.

Building on this foundation, this paper introduces an adaptive neural network encoding and decoding framework that adjusts the complexity-relevance tradeoff in response to network conditions and application requirements. We offer a training procedure that trains the neural network in a cascaded (or tandem) fashion while connecting intermediate layers from the encoder to the decoder to enable multiple modes of communication with varying complexity-relevance modes. We apply the training procedure to a mmWave 5G throughput prediction problem by utilizing the Lumos5G dataset [6]. The dataset captures throughput as perceived by applications running on a user equipment (UE) along with correlated features such as the longitude, latitude, and received signal strength.

Furthermore, this work is among the first to apply and investigate the IB theory in the context of sequential (time-series) problems and models. We provide new insights and identify important challenges while partially addressing them. First, we find that incorporating the temporal dimension of sequential models into the IB analysis is essential for a comprehensive understanding of the training process. Second, we find that estimating the mutual information (MI) can be challenging due to the large hidden temporal states and sampling limitations. Therefore, visualizing a 3-dimensional information plane that incorporates the hidden temporal states and employing metrics such as the conditional MI become vital to assess the redundancy of the temporal states, upon which the number of states to be estimated can be reduced. A key finding is that compression not only occurs as the training progresses (as has been reported in the seminal IB works [4], [5]), but it also occurs across the temporal dimension of the sequential model (i.e., across the hidden temporal states).

The rest of the paper is organized as follows. Section II provides a brief background on IB. Section III provides an overview of relevant literature. Section IV presents the proposed dynamic framework and training mechanism. Section V discusses the use case of mmWave-enabled throughput prediction under a split learning setup. In Section VI, we conduct experiments to confirm the viability of the proposed mechanism on the Lumos5G dataset, along with a highlight of aforementioned challenges and new insights. Section VII provides concluding remarks.

II. The Information Bottleneck Framework

Tishby and Zaslavsky suggested viewing each individual layer in a neural network as a random variable [5]. Therefore, a neural network can be considered as a Markov chain of successive representations. Using MI to gauge the flow of information can be advantageous for two reasons. First, compared to statistical correlation, MI is a more general measure of statistical dependence with a common scale (e.g., bits or nats). Second, it is an invariant measure, i.e.,

$$I(X;Y) = I\big(\phi(X), \varphi(Y)\big), \tag{1}$$

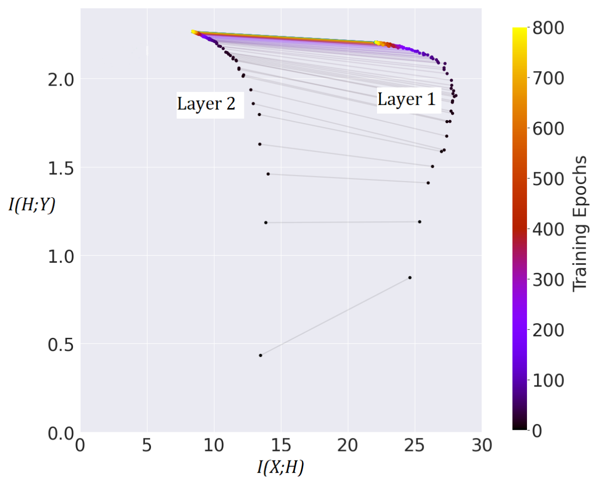

Fig. 1. Information plane showing the evolution of two layers as the training progresses. Both layers exhibit a fitting phase followed by a compression phase.

where $I(\cdot;\cdot)$ denotes the MI between two random variables, and $\phi(\cdot)$ and $\varphi(\cdot)$ are deterministic functions. The invariance property implies that MI can provide a unifying pair of lens to probe and compare neural networks regardless of the architecture. IB provides a computational framework for finding the optimal tradeoff between the compression of input data X and the preservation of information about target label Y by minimizing the Lagrangian,

$$\min_{p(H|X),p(H|T),p(H)} I(X;H) - \beta I(H;Y) \tag{2}$$

where β determines the level of relevant information captured by neural network layer H.

A useful graphical tool is the information plane which exhibits the MI of a hidden layer with respect to target Y versus the MI of the hidden layer with respect to input X. An example of the information plane is shown in Fig. 1 (obtained from our simulations). Each curve in the information plane corresponds to a layer. In the beginning, all layers gain information with respect to both X and Y as the training progresses, which is called the fitting phase. At some point in the training, $I(X;H)$ starts to reduce, exhibiting a compression phase. Saxe et al. provided arguments and set of experiments that shows that compression occurs only when the layers contain a double-saturating activation function (such as tanh and sigmoid) [7]. Chelombiev et al. showed that, by using adaptive and robust estimation techniques, compression can occur without necessarily having double-edge saturation in activation functions [8]. This speaks to the importance of deploying sensitive and robust estimation techniques.

III. Literature Survey

Rate distortion theory, of Shannon and Kolmogorov, characterizes the tradeoff between the signal representation and the average distortion of the reconstructed signal [9]. In the beginning of this century, Tishby et al. proposed IB as a generalized distortion theory [4], [10]. In 2015 and 2017, Tishby et al. showed how the IB framework applies to deep neural networks [5], [11]. Since then, there has been a great interest in the IB theory and its applications [12]–[14].

The literature can be classified into three categories. One category attempts to further develop, analyze, and scrutinize the fundamentals of the IB theory [7], [14]–[17]. A second category applies IB principles to analyze and interpret the inner workings of ML models, while a third category applies IB principles and observations to improve deep learning-based algorithms and applications [18]. The three categories overlap as further development on the theory and deeper probing into the models can lead to improving it.

To this paper's context, IB can fit in semantic communications [19]–[22] and wireless-edge learning [23], [24]. Notably, Beck et al. consider a semantic communication task such that a message is transmitted while preserving the relevant meaning [25]. They cast the problem as an IB problem that allows messages to be compressed while preserving the relevant information as possible. Pezone et al. propose a goal-oriented system for edge-learning based on the IB framework [26]. They adapt β to optimize the tradeoff between the complexity and the relevance of the encoded information in order to minimize the energy consumption under delay constraints. Similarly, based on IB, Binucci et al. employ convolutional encoders at the edge to compress relevant data before being offloaded to an edge station [22].

IV. Adaptive Split Encoder-Decoder with feedback signals

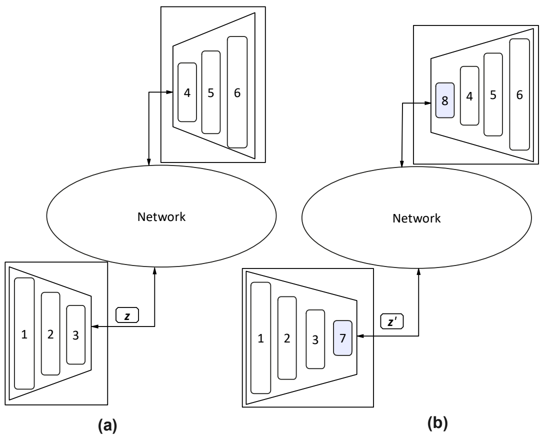

Fig. 2. Split encoder-decoder neural network where encoder produces two latent codes; in (a) the encoder sends code z, whereas in (b) the encoder augments one more layer and sends code z'.

Network conditions and application requirements can vary over time and between network slices. With the information plane in mind, the intuition is to selectively choose which hidden layer's output from the encoder to transmit to the decoder. Consider an optimized neural network where the neural network hyper-parameters (i.e., number of layers, number of nodes, learning rate) are optimized to provide the best achievable predictive performance (e.g., through hyper-parameter search). Denote the output of the optimized neural network encoder by latent representation $\boldsymbol{z}$ (as shown in Fig. 2(a)). If we add a new bottleneck layer with output $\boldsymbol{z}'$ to the trained encoder as shown in Fig. 2(b), following the data processing inequality and by construction, the new encoder-decoder neural network is less optimal where adding an additional bottleneck layer loses crucial information due to further imposed compression. In Figs. 2(a) and 2(b), $I(X;H_3) \geq I(X;H_7)$, entailing that code $\boldsymbol{z}'$ requires fewer bits than code $\boldsymbol{z}$ for encoding. It follows that the decoder's predictive performance receiving code $\boldsymbol{z}'$ is at most equal to or worse than the decoder's performance receiving code $\boldsymbol{z}$. By combining both encoders in Figs. 2(a) and 2(b), we can get a dynamic and adaptive neural network encoder that can vary the informativeness of latent representation depending on which layer is selected as the transmitted bottleneck. In doing so, one can train two decoders that receive representations $\boldsymbol{z}$ and $\boldsymbol{z}'$, respectively. A more compact solution that results in one encoder and decoder is summarized in Algorithm 1. Train a first encoder-decoder neural network (line 1). Freeze the trained layers, add a new layer to the encoder and to the decoder, respectively (lines 2-6), and retrain the overall network. Create a (skip) connection between the output of the trained encoder and the trained decoder (line 5). Finally, ensure that the new layers results in a less optimal predictive performance after training (line 7), where the performance gap can be tuned by trial and error.

Algorithm 1 Cascaded training procedure to achieve two modes of complexity-relevance tradeoff

```
1: Encoder1, Decoder1 ← Train([Encoder1, Decoder1])
2: Freeze(Encoder1, Decoder2)
3: NN2Encoder ← [Encoder1 + new layer A]
4: NN2Decoder ← [new layer B + Decoder1]
Ensure: Layer A output = layer B input
5: Connect Encoder1 and Decoder1
6: Encoder2, Decoder2 ← Train([Encoder2,Decoder2])
Ensure: $I(Y;\mathrm{Decoder1Output}) \leq I(Y;\mathrm{Decoder2Output})$
```

Figure 3 depicts the general framework. Here, an orchestrator can obtain key-performance indicators for each network function from an oracle, and monitors the neural network's performance. Based on the performance of the decoder and network conditions, the orchestrator can instruct the network element containing the neural network encoder to transmit either $\boldsymbol{z}$ and $\boldsymbol{z}'$.

V. Use Case: mmWave Throughput Prediction

Consider a dual-connectivity setup where a UE is connected to a macro base station (BS) and a mmWave-based micro BS, as shown in Fig. 4(a). The user can get user-plane connections with high throughput via a mmWave channel while having reliable control-plane (and user-plane) connection to the macro BS. Since mmWave based beams are highly directed, mobile users experience highly- and widely-varying throughput that

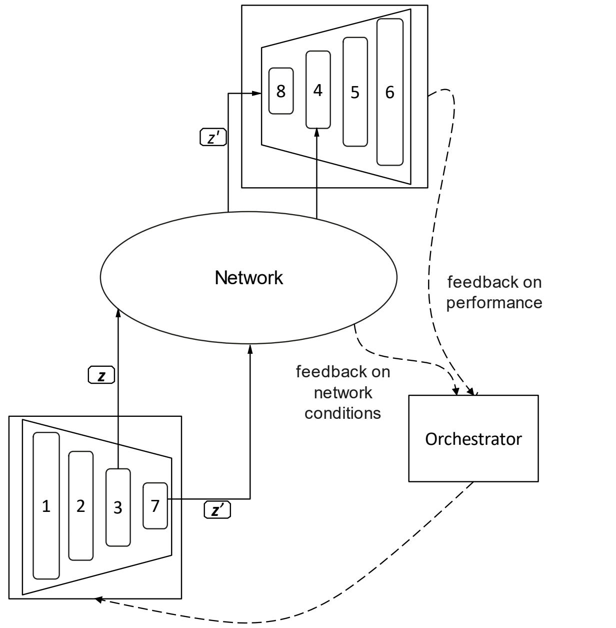

Fig. 3. Dynamic split neural network where the encoder can send either intermediate latent code z or less informative latent code z'. An orchestrator monitors for network conditions and receives feedback about the decoder's performance, and instructs the encoder to select which latent code to transmit.

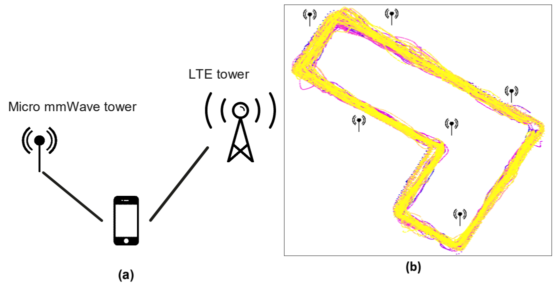

Fig. 4. (a) UE in dual-connectivity mode to a micro-mmWave based tower and a macro-base station; (b) 1300 meter loop area in Minneapolis downtown area based on the Lumos5G dataset [6].

correlates with spatio-temporal features in the scene [6]. Many factors, such as position of the user, proximity of contending users to each other, static and dynamic obstacles, user's movement patterns and direction, have an effect on the perceived throughput. The authors of [6] captured some of these features in the Lumos5G dataset which consists of 70,000 samples. Each sample includes 11 features and an associated perceived throughput by the UE. The features are longitude, latitude, moving speed, compass direction, and six LTE and new-radio signal strength measurements, c.f. [6, Table 1]. The samples are collected along a 1300 meter loop area in Minneapolis downtown area as shown in Fig. 4(b). Throughput prediction is useful for over-the-top applications or network slices, such as video streaming, real-time gaming, and virtual reality. For example, augmented reality can cache/request information ahead of time depending on the predicted throughput. Some slices/applications would require more accurate throughput

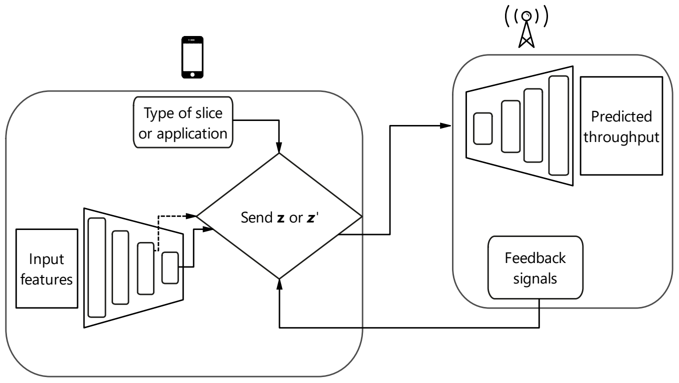

Fig. 5. Adaptive encoder-decoder network distributed between the UE and the edge network. The encoder is trained/configured (using Algorithm 1) to send intermediate latent representations based on feedback signals.

prediction performance compared to others.

In general, some of the useful spatio-temporal features can be observed by the user equipment while others are can be obtained by the access network through sensing or network measurements, e.g., distance of the user to mm-wave BS, and dynamic/static obstacles in the vicinity. Therefore, a split encoder-decoder model with encoders residing in the UEs and a decoder residing in the access network is suitable to learn experiences from the different users while minimizing feature exchange and maintaining some level of privacy.

The proposed dynamic split encoder-decoder framework can be utilized for the mmWave throughput prediction problem. In Fig. 5, a neural network encoder prompts the input features collected by the UE (or received by the UE from other sources). Additionally, the encoder can get performance requirements from the current application being used. The access network can send back a feedback to the user of network conditions, such as whether there is congestion on the allocated control-plane band or whether computing hardware is congested in the mobile edge. Depending on the feedback received and the type of slice/application being used, the UE can decide whether to send the less-informative output of the encoder (which in Fig. 5 is the output of the 4th layer) or the more-informative output from the intermediate layer.

Unlike some benchmark datasets used in the IB literature [4], [5], [7], here we have a time-series prediction problem. The input and output time series can be modeled as random processes $X_t, Y_t \in \mathbb{R}^D$, where $t = 1, \ldots, T$ and D correspond to the number of input features. At each timestep t, X_t has a corresponding latent state for each layer l, $H_t^{(l)}$. We utilize a long-short term memory (LSTM) network for the encoder and a time-distributed dense neural network for the decoder, as shown in Fig. 6. For the first phase of training, we train two LSTM layers for the encoder followed by a time-distributed Dense neural network for the decoder. Hyper-parameter search shows that two to three neural network layers provide a good performance for the Lumos5G dataset. Following Algorithm 1, we add a third LSTM layer at the encoder's output and a time-distributed Dense layer at the decoder's input. Then, we create an intermediate connection between the second layer of the encoder and the second layer of the decoder. For each inference query, the decoder receives either $H_T^{(2)}$ or $H_T^{(3)}$ *but not both*. Here, note that latent codes transmitted from $H^{(3)}$ goes through an extra layer of processing at the decoder compared to latent codes transmitted from $H^{(2)}$.

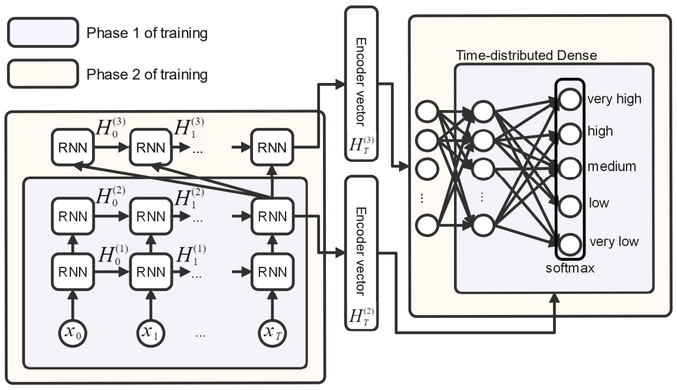

Fig. 6. Encoder-decoder LSTM-Dense model architecture trained in a cascaded fashion.

VI. New Insights and Analysis

In this section, we test the main proposal on the Lumos5G dartaset where we construct the adaptive LSTM-Dense encoder-decoder neural network following Algorithm 1 as shown in Fig. 6. The first two LSTM layers have 128 cells, while the added layer after training has 32 cells. A common approach to estimate the mutual information using the binning method which was employed in the original IB papers [4], [5]. The binning method quantizes the data into a number of bins, followed by estimating the mutual information. Binning is sensitive on the choice of the bin size and resultant boundaries, which affects the accuracy and consistency of the mutual information estimate. We utilize the kernel density estimator by Kolchinsky and others [7], [27], [28] and the Gaussian-copula MI (GCMI) estimator by Ince and others [29] for $I(Y;T)$ and $I(X;T)$, respectively. The GCMI estimator is robust to multidimensional variables and different marginal distributions. Moreover, it can be extended to higher order quantities such as conditional mutual information, which we strategize that it will be needed with sequential or recurrent neural network models. In sequential models, we not only care about the relation of a hidden layer with respect to the input and the output, but we may need to quantify the effect of a hidden state on a hidden state from a previous timestep, e.g., $\mathcal{I}(X;H_j|H_{j-1},H_{j-2})$.

Similar to [6], the input consists of $T = 20$ timesteps, and the output of the decoder provides a classification for 20 timesteps. The size of the testing set is set to 10% of the dataset, the learning rate is 10^{-2}, and the batch size is 256.

From the perspective of the decoder, the first layer of the encoder's LSTM network has a temporal state for each timestep, i.e., $H^{(1)} = [H_1^{(1)}, H_2^{(2)}, \ldots, H_T^{(2)}]$, while the second layer conveys only the final temporal state, i.e., $H^{(2)} = [H_T^{(2)}]$. To view the information plane for the first phase of training, we need to measure $I(\cdot;H^{(1)})$ and $I(\cdot;H^{(2)})$. However, $H^{(1)}$ consists of 20 hidden temporal states, making the estimation of the MI very difficult. To reduce the size of the hidden state, we perform the following steps. First, Fig. 7 shows a 3-dimensional plot of $I(H_t^{(1)};y_\tau)$ versus timestep t and the number of epochs for $\tau = 5$. We observe that the last temporal

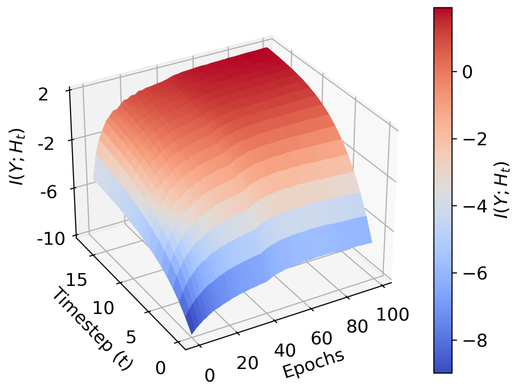

Fig. 7. 3D Information curve: $I(H_t;Y)$ with respect to timestep t and number of training epochs.

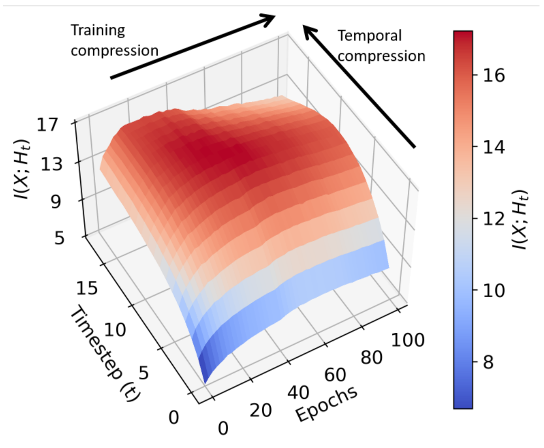

Fig. 8. 3D Information curve: $I(H_t;X)$ with respect to timestep t and number of training epochs. The information curve shows compression behavior with the training epochs and the hidden temporal states.

state ($H_T^{(1)}$) contains the largest amount of information, where $I(H_t^{(1)};y_\tau)$ increases monotonically with t. Second, Fig. 8 shows a 3-dimensional plot of $I(X_1,\ldots X_t;H_1,\ldots,H_t)$ versus timestep t and the number of epochs. Interestingly, we find that compression not only occurs as the training progresses (i.e., with the number of epochs), but it also occurs across the hidden temporal states, which to our knowledge has not been reported before in the open literature. The observed trends in Figs. 7 and 8 were found to be consistent for other values of τ. The trends imply that there can be strong redundancy across the hidden temporal states and that the last few temporal states can be sufficient to represent all the hidden states. Thanks to the flexibility of the GCMI estimation technique [29], we can also measure the conditional MI to gauge the amount of redundancy contained in the temporal hidden states. We find that $I(x_1,\ldots,x_T;H_T^{(1)}|H_{T-1}^{(1)}) = 14.24$ bits, $I(x_1,\ldots,x_T;H_T^{(1)}|H_{T-1}^{(1)},H_{T-2}^{(1)}) = 3.23$ bits, and $I(x_1,\ldots,x_T;H_T^{(1)}|H_{T-1}^{(1)},H_{T-2}^{(1)},H_{T-3}^{(1)}) = 2.37$ bits. The conditional MI keeps decreasing as we condition on earlier hidden states. This implies that useful information are sufficiently represented in the last few temporal states. Therefore, for the information plane in Fig. 9, we consider

$$H^{(1)} \approx [H_T^{(1)}, H_{T-1}^{(1)}, H_{T-2}^{(1)}, H_{T-3}^{(1)}]. \tag{3}$$

Fig. 9 shows the information plane for both training phases. In the first training phase, the two layers converge to $I(H;y_t) = 2.3$ bits approximately, and $I(H;X_1,\ldots,X_T)$ of 17 bits for the first layer and 9 bits for the second layer. Following Algorithm 1, in the second phase, we freeze the trained neural network layers, add a new bottleneck layer to the encoder, and re-train the neural network. In the second training

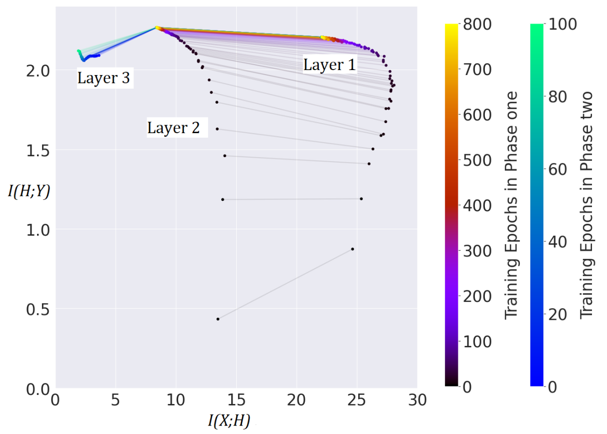

Fig. 9. The information plane showing the evolution of the neural network layers' MI profiles, $I(H;Y)$ versus $I(H;X)$, with the number of epochs. There are two training phases where starting from phase 2, the encoder's neural network layers get frozen to train the added layer.

phase, the added layer converges to $I(H_T^{(3)};x_1,\ldots,x_T) = 2.5$ bits and $I(H_T^{(3)};y_t) = 2.25$ bits. Note how adding the new bottleneck layer reduces $I(H_T^{(3)};y_t)$ and correspondingly the prediction performance, which is intended by design. Using the framework explained in Sections IV and V, we can adaptively switch between the outputs of encoder layer 2 and encoder layer 3, respectively.

The impact of a reduction in the target MI and the predictive performance of a network function on the overall network performance remains unclear. We suggest framing this issue as an optimization or search problem, aiming to minimize signaling overhead or feature exchange under specific conditions without significantly compromising network's performance. Additionally, this study highlights the challenge of estimating the MI in sequential models.

VII. Conclusions

Based on the IB theory and the data processing inequality, this paper presents a dynamic framework and a training mechanism to tune the informativeness of the shared latent representation for split-learning based network functions. This dynamic tunability provides flexibility to address varying network conditions and application requirements. We apply the training mechanism to a mmWave throughput prediction problem using the Lumos5G dataset as a proof of concept. This paper highlights (*i*) the importance of incorporating the temporal domain into the IB analysis, and (*ii*) the challenge of estimating the information plane in sequential models which can contain a large number of hidden temporal states. Interestingly, we also observe a compression phenomena that occurs across the temporal domain in sequential models.

References

[1] A. Gholami, Z. Yao, S. Kim, M. W. Mahoney, and K. Keutzer, "AI and memory wall," *RiseLab Medium Post*, 2021.

[2] M. G. Poirot, P. Vepakomma, K. Chang, J. Kalpathy-Cramer, R. Gupta, and R. Raskar, "Split learning for collaborative deep learning in healthcare," *arXiv:1912.12115*, 2019.

[3] W. Wu, M. Li, K. Qu, C. Zhou, X. Shen, W. Zhuang, X. Li, and W. Shi, "Split learning over wireless networks: Parallel design and resource management," *IEEE J. Selected Areas Commun.*, vol. 41, no. 4, pp. 1051–1066, 2023.

[4] N. Tishby, F. C. Pereira, and W. Bialek, "The information bottleneck method," *arXiv:physics/0004057*, 2000.

[5] N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," in *Proc. IEEE Inf. theory workshop*. IEEE, 2015, pp. 1–5.

[6] A. Narayanan, E. Ramadan, R. Mehta, X. Hu, Q. Liu, R. A. Fezeu, U. K. Dayalan, S. Verma, P. Ji, T. Li *et al.*, "Lumos5G: Mapping and predicting commercial mmwave 5G throughput," in *Proc. ACM Internet Meas. Conf.*, 2020, pp. 176–193.

[7] A. M. Saxe, Y. Bansal, J. Dapello, M. Advani, A. Kolchinsky, B. D. Tracey, and D. D. Cox, "On the information bottleneck theory of deep learning," *J. Stat. Mech.: Theor. Exp.*, vol. 2019, no. 12, p. 124020, 2019.

[8] I. Chelombiev, C. Houghton, and C. O'Donnell, "Adaptive estimators show information compression in deep neural networks," *arXiv:1902.09037*, 2019.

[9] T. M. Cover, *Elements of information theory*. John Wiley & Sons, 1999.

[10] N. Slonim, "The information bottleneck: Theory and applications," Ph.D. dissertation, Hebrew University of Jerusalem Jerusalem, Israel, 2002.

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

[12] H. Hafez-Kolahi and S. Kasaei, "Information bottleneck and its applications in deep learning," *arXiv:1904.03743*, 2019.

[13] B. C. Geiger and G. Kubin, "Information bottleneck: Theory and applications in deep learning," *Entropy*, vol. 22, no. 12, 2020.

[14] Z. Ye, "Awesome information bottleneck," https://github.com/ZIYU-DEEP/Awesome-Information-Bottleneck, 2022.

[15] A. Kolchinsky, B. D. Tracey, and S. Van Kuyk, "Caveats for information bottleneck in deterministic scenarios," *arXiv:1808.07593*, 2018.

[16] A. Achille and S. Soatto, "Emergence of invariance and disentanglement in deep representations," *J. Mach. Learn. Res.*, vol. 19, no. 1, pp. 1947–1980, 2018.

[17] Z. Piran, R. Shwartz-Ziv, and N. Tishby, "The dual information bottleneck," *arXiv:2006.04641*, 2020.

[18] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, "Learning deep representations by mutual information estimation and maximization," *arXiv:1808.06670*, 2018.

[19] E. C. Strinati and S. Barbarossa, "6G networks: Beyond shannon towards semantic and goal-oriented communications," *Comput. Netw.*, vol. 190, p. 107930, 2021.

[20] H. Xie, Z. Qin, G. Y. Li, and B.-H. Juang, "Deep learning enabled semantic communication systems," *IEEE Trans. Signal Process.*, vol. 69, pp. 2663–2675, 2021.

[21] Z. Qin, X. Tao, J. Lu, and G. Y. Li, "Semantic communications: Principles and challenges," *arXiv:2201.01389*, 2021.

[22] D. Gündüz, Z. Qin, I. E. Aguerri, H. S. Dhillon, Z. Yang, A. Yener, K. K. Wong, and C.-B. Chae, "Beyond transmitting bits: Context, semantics, and task-oriented communications," *IEEE J. Selected Areas Commun.*, vol. 41, no. 1, pp. 5–41, 2022.

[23] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, "Wireless network intelligence at the edge," *Proc. IEEE*, vol. 107, no. 11, pp. 2204–2239, 2019.

[24] X. Wang, Y. Han, V. C. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Commun. Surv. Tutor.*, vol. 22, no. 2, pp. 869–904, 2020.

[25] E. Beck, C. Bockelmann, and A. Dekorsy, "Semantic information recovery in wireless networks," 2023.

[26] F. Pezone, S. Barbarossa, and P. Di Lorenzo, "Goal-oriented communication for edge learning based on the information bottleneck," in *Proc. IEEE ICASSP*, 2022, pp. 8832–8836.

[27] A. Kolchinsky and B. D. Tracey, "Estimating mixture entropy with pairwise distances," *Entropy*, vol. 19, no. 7, p. 361, 2017.

[28] A. Kolchinsky, B. D. Tracey, and D. H. Wolpert, "Nonlinear information bottleneck," *Entropy*, vol. 21, no. 12, p. 1181, 2019.

[29] R. A. Ince, B. L. Giordano, C. Kayser, G. A. Rousselet, J. Gross, and P. G. Schyns, "A statistical framework for neuroimaging data analysis based on mutual information estimated via a gaussian copula," *Human brain mapping*, vol. 38, no. 3, pp. 1541–1573, 2017.