Progressive Transmission of High-Dimensional Data Features for Inference at the Network Edge

Qiao Lan[1], Qunsong Zeng[1], Petar Popovski[2], Deniz Gündüz[3], and Kaibin Huang[1]
[1]Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong
[2]Department of Electronic Systems, Aalborg University, Aalborg, Denmark
[3]Department of Electrical and Electronic Engineering, Imperial College London, London, UK
Email: {qlan, qszeng, huangkb}@eee.hku.hk, petarp@es.aau.dk, d.gunduz@imperial.ac.uk

***Abstract*—Uploading high-dimensional features from edge devices to an edge server over wireless channels creates a communication bottleneck for *edge inference*. To tackle the challenge, we propose the *progressive feature transmission* (ProgressFTX) protocol, which minimizes the overhead by progressively transmitting features until a target confidence level is reached. The control of the protocol to accelerate inference is designed with two key operations. The first, *importance-aware feature selection*, guides the server to select the most discriminative feature dimensions. The second is *transmission-termination control* such that the feature transmission is stopped when the incremental uncertainty reduction by further transmission is outweighed by its communication cost. The indices of the selected features and transmission decision are fed back to the device in each slot. The sub-optimal policy is obtained for classification using a convolutional neural network. Experimental results on a real-world dataset shows that ProgressFTX can substantially reduce the communication latency compared to conventional feature pruning and random feature transmission.**

***Index Terms*—Edge computing, edge AI, progressive transmsision.**

I. Introduction

Recent years have witnessed the extensive deployment of *artificial intelligence* (AI) technologies at the network edge to gain fast access to and processing of mobile data. This gives rise to two active research challenges: (1) *edge learning*, where AI models are trained via distributed machine learning; and (2) *edge inference*, which is the theme of this work and deals with operating of such models at edge servers to provide inference services [1], [2]. To reduce communication overhead and protect data privacy, the state-of-the-art edge inference algorithms build upon an architecture termed *split inference*, in which the model is partitioned into *device* and *server* sub-models [3]. Using the device sub-model, an edge device extracts features from a raw data sample and uploads them to a server, which then uses these features to compute an inference result and sends it back to the device. However, it is challenging to upload high-dimensional features over resource-constrained wireless channels. To improve the communication efficiency, we propose and optimize a simple protocol, termed *progressive feature transmission* (ProgressFTX).

This work was partly supported by the Villum Investigator Grant "WATER" from the Velux Foundation, Denmark, and partly supported by the UK EPSRC (EP/T023600/1) under the CHIST-ERA program (CHISTERA-18-SDCDN-001).

The current research tackling the communication bottleneck in edge inference can be categorized into two main directions: joint source-and-channel coding and feature pruning. The key of the former is utilizing an autoencoder architecture, consisting of an encoder for the device and an decoder for the server sub-models, jointly trained to simultaneously perform inference and efficient transmission [4], [5]. For the latter, the principle is to discard features according to their various importance levels. A series of importance evaluation methods have been presented, including observing the effect of removing a feature on the inference performance [6], and Taylor expansion of the error induced by pruning [7]. The proposed ProgressFTX targets split inference and aims at achieving a higher efficiency than these existing one-shot feature pruning techniques by employing, in addition to importance awareness, a stochastic control mechanism according to the channel state.

We consider the scenario in which split inference is deployed for classification tasks. Based on the ProgressFTX protocol, a device progressively transmits selected features to improve classification accuracy, until a target accuracy is reached or the expected communication cost becomes too high, as informed by the server. The principle of progressive transmission also underpins *hybrid automatic repeat request* (HARQ) [8], a basic mechanism for communication reliability. ProgressFTX substantially differs from HARQ as our protocol is a cross-disciplinary design targeting both high classification accuracy and low communication overhead. In addition, the incremental redundancy in ProgressFTX refers to the use of increasing number of features, rather than punctured versions of the encoder output in HARQ. The HARQ operations of punctured error control coding, combining, and error detection are replaced with feature extraction, feature cascading, and classification, respectively, in the context of ProgressFTX.

The contribution of this work is the design of ProgressFTX protocol and its sub-optimal control policy. The trade-off between uncertainty reduction and communication cost increase due to transmitting more features motivates the optimization of ProgressFTX, which is formulated as a problem of optimal stochastic control with the dual objective: minimizing the uncertainty and communication cost, respectively. The problem is solved for general *convolutional neural network* (CNN) models by two practical algorithms for importance-aware

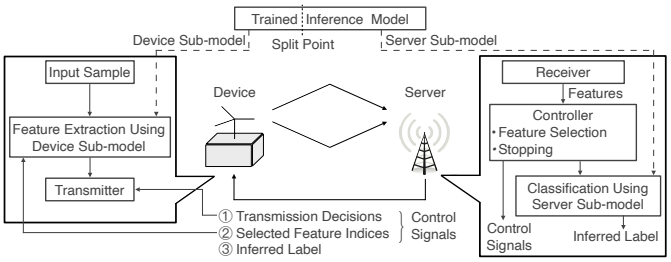


Fig. 1. An edge inference system.

feature selection and stopping control, respectively. First, we evaluate the importance of a feature map using the gradients of associated model parameters readily available from training. Second, we advocate the use of a low-complexity regression model that is trained to predict the incremental inference uncertainty. The model with close-to-optimal performance allows for a simple linear search to find the sub-optimal stopping time of ProgressFTX. Experimental results demonstrate the gain of ProgressFTX beyond conventional one-shot feature pruning and random feature selection schemes in terms of communication latency.

II. Models and Metrics

We consider the edge inference system in Fig. 1, where local data at an edge device are compressed into features and sent to a server for remote inference on a trained model.

A. CNN Classification Model

A CNN model comprises multiple *convolutional* (CONV) layers followed by multiple *fully-connected* (FC) layers. To implement split inference, the model is divided into device and server sub-models, which are represented by functions $f_{\mathsf{d}}(\cdot)$ and $f_{\mathsf{s}}(\cdot)$, respectively. Following existing designs [3], [6], the splitting point is *chosen* right after a CONV layer. Let N denote the number of CONV filters in the last layer of $f_{\mathsf{d}}(\cdot)$, each of which outputs a feature map with height L_{h} and width L_{w}. The set of all feature-map indices is denoted as $\mathcal{W} = \{1, 2, ..., N\}$. The tensor of all feature maps from an arbitrary input sample is denoted as $\mathbf{X} \in \mathbb{R}^{N \times L_{\mathsf{h}} \times L_{\mathsf{w}}}$, and the n-th map is $\mathbf{X}(n)$. Let $\mathbf{X}_k$ denote the tensor of cumulative feature maps received by the server by slot k, and $\mathcal{W}_k$ is the corresponding index set of feature maps. Then, $\mathbf{X}_k$ can be related to $\mathbf{X}$ by

$$\mathbf{X}_k(n) = \begin{cases} \mathbf{X}(n), & n \in \mathcal{W}_k, \\ \mathbf{0}, & \text{otherwise.} \end{cases} \tag{1}$$

In slot k, the server sub-model can compute the posteriors $\Pr(\ell|\mathbf{X}_k)$ for the input $\mathbf{X}_k$ via the forward propagation [1]. Then, the label $\hat{\ell}$ is estimated by posterior maximization: $\hat{\ell} = \text{argmax}_\ell \Pr(\ell|\mathbf{X}_k)$. As a metric of inference performance, inference uncertainty is measured using the *entropy of posteriors* commonly adopted for *deep neural network* (DNN) classifiers [9]. Given the input $\mathbf{X}_k$, the metric, denoted as $H(\mathbf{X}_k)$, is given as

$$H(\mathbf{X}_k) \triangleq -\sum_{\ell=1}^{L} \Pr(\ell|\mathbf{X}_k) \log \Pr(\ell|\mathbf{X}_k). \tag{2}$$

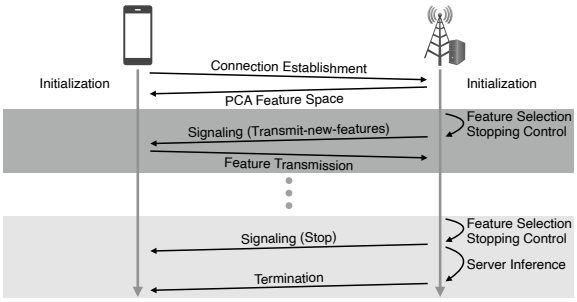


Fig. 2. Illustration of the ProgressFTX protocol.

B. Communication Model

Consider uplink transmission of feature maps for remote inference at the server. Each feature map is quantized with a sufficiently high resolution of Q bits such that quantization errors are negligible. The basic unit of transmitted data is a *feature map*, which is a $L_{\mathsf{h}} \times L_{\mathsf{w}}$ matrix. The time of the communication channel is divided into slots, each spanning T seconds. The device is allocated a narrow-band channel with the bandwidth denoted as B and the gain in slot k as g_k. We consider a Gaussian channel, where $g_k = g_0$ for all k and is known to both the transmitter and receiver. The transmit *signal-to-noise ratio* (SNR) is fixed as ρ and the rate is matched to the channel as $R_0 = B \log_2(1+\rho g_0)$. Accordingly, the transmission rate (in feature-maps/slot) for a slot can be written as $Y_0 = \lfloor \frac{R_0 T}{Q L_{\mathsf{h}} L_{\mathsf{w}}} \rfloor$.

III. ProgressFTX Protocol and Control Problem

A. ProgressFTX Protocol

The ProgressFTX protocol implements a time window of continuous progressive feature transmission of a single data sample. If the remote inference fails to achieve the target accuracy, the task is declared a failure for a latency sensitive application (e.g., autonomous driving) or otherwise another instance of progressive transmission may be attempted after some random delay. The steps in the ProgressFTX protocol are illustrated in Fig. 2 and described as follows.

1) *Feature selection:* At the beginning of a slot intended for feature transmission, the server selects the (indices of) feature maps and informs the device to transmit the corresponding features. The selection of feature maps depends on their importance levels (as elaborated in Section IV-A), which are available from the training phase and used in the ensuing edge inference phase. This method is data-sample independent; and hence, it can be efficiently implemented at the server which does not have access to the samples. Specifically, the server records received feature maps in $\mathbf{X}_k$ and their indices in $\mathcal{W}_k \subseteq \mathcal{W}$. The indices of selected features maps are stored in $\mathcal{S}_k$, where $\mathcal{S}_k \subseteq \mathcal{W} \setminus \mathcal{W}_k$. A subset $\mathcal{S}_k$ satisfying this constraint is termed an *admissible subset*. Moreover, the number of features to be transmitted, $|\mathcal{S}_k|$, is limited by the number of untransmitted features and the transmission rate: $|\mathcal{S}_k| = \min\{N - |\mathcal{W}_k|, Y_0\}$.
2) *Stopping control:* The number of features of a particular sample needed for remote classification to meet the

accuracy requirement is sample-dependent. Neither the device nor the server has prior knowledge of this number since each has access to either the sample or the model but not both. Online stopping control aims at minimizing the number of transmitted features under the said requirement. Its procedure is described as follows while its optimization problem is formulated in Section III-B and solved in Section IV-B. Let b_k indicate the server's decision on whether the device should transmit features in slot k (i.e., $b_k = 1$), or *stop* the progressive transmission (i.e., $b_k = 0$). The decision is made using a regression model for uncertainty prediction (see Section IV-B). If the decision is to transmit, proceed to the next step; otherwise, go to Step 6.

3) *Feedback:* The decisions on feature selection and stopping control are communicated to the device by feedback: *(a)* If $b_k = 1$, the feedback contains a *transmit-new-features signal* and the indices of the features in $\mathcal{S}_k$ that the server wants the device to transmit in the current slot. *(b)* If the server decides to stop the transmission (i.e., $b_k = 0$), a *stop signal* is fed back to instruct the device to terminate feature transmission.
4) *Feature transmission:* Upon receiving a transmit-new-features signal with indices $\mathcal{S}_k$, the device transmits the *incremental* feature tensor comprising selected features:
$$\Delta \mathbf{X}_k = \left[\mathbf{X}(n_1), \mathbf{X}(n_2), ..., \mathbf{X}(n_{|\mathcal{S}_k|})\right]^T, \quad (3)$$
where $n_1, ..., n_{|\mathcal{S}_k|}$ are indices in $\mathcal{S}_k$. At the end of slot-k, the server updates the set of received features $\mathcal{W}_{k+1} = \mathcal{W}_k \bigcup \mathcal{S}_k$ and the partial feature tensor $\mathbf{X}_{k+1} = f_{\mathsf{a}}(\mathbf{X}_k, \Delta \mathbf{X}_k)$, where the assemble operator f_{a} works as follows. For $n \in \mathcal{W}_k$, f_{a} copies the already received feature maps from $\mathbf{X}_k$ as $\mathbf{X}_{k+1}(n) = \mathbf{X}_k(n)$; for $n \in \mathcal{S}_k$, it retrieves the newly received feature maps as $\mathbf{X}_k$ as $\mathbf{X}_{k+1}(n) = \Delta \mathbf{X}_k(n')$ where n' is index of the element in the sequence $[n_1, n_2, ..., n_{|\mathcal{S}_k|}]$ that equals to n; $\mathbf{X}_{k+1}(n) = \mathbf{0}$ otherwise.
5) *Server inference:* The server infers an estimated label $\hat{\ell}$ and its uncertainty level using the partial feature tensor $\mathbf{X}_k$ and a trained model as discussed in Section II-A. Then we restart ProgressFTX for another data sample.
6) *Termination:* Upon receiving a *stop signal*, the device terminates the process of progressive transmission. The latest estimated label is downloaded to the device.

In summary, feature selection, stopping control, feature transmission and uncertainty evaluation are executed sequentially in each slot. The transmission is terminated at the time when the uncertainty is evaluated to fall below the target or uncertainty reduction is outweighed by the corresponding communication cost.

B. Control Problem Formulation

ProgressFTX has two objectives: 1) maximize the uncertainty reduction (or equivalently, improvement in inference accuracy), and 2) minimize the communication cost. The optimal stochastic control is formulated as a dynamic programming problem as follows. We define the system *state* observed by the server at the beginning of slot k as the following tuple
$$\theta_k \triangleq (\mathbf{X}_k, \mathcal{W}_k). \quad (4)$$
Recall that $\mathbf{X}_k$ represents the received partial feature tensor, and $\mathcal{W}_k$ the indices of received features maps. The control policy, denoted as Ω, maps θ_k to the feature-selection *action*, $\mathcal{S}_k$, and stopping-control action, b_k, $\Omega : \theta_k \to (\mathcal{S}_k, b_k)$. The net reward of transitioning from θ_k to θ_{k+1} is the decrease in inference uncertainty minus the communication cost, which can be written as $u(\theta_k, \Omega) = H(\mathbf{X}_k) - H(\mathbf{X}_{k+1}) - b_k c_0$, where c_0 denotes the transmission cost of one slot. For sanity check, $u(\theta_k, \Omega) = 0$ if $b_k = 0$. The net reward for slot k requires server's evaluation based on transmitted features if the control decision is proceeding transmission; and thus, is unknown before making the decision for the slot. Hence, the expected net reward should be considered as the utility function. Without loss of generality, consider K slots with the current slot set as slot 1. Then the system utility is defined as the expectation of the sum rewards over K slots conditioned on the system state and control policy:
$$U(\theta_1, \Omega) \triangleq \mathbb{E}_{\{\theta_k\}_{k=2}^K} \left[\sum_{k=1}^{K} u(\theta_k, \Omega) \;\middle|\; \theta_1, \Omega \right]. \quad (5)$$
The ProgressFTX control problem can be readily formulated as the following dynamic program:

$$\text{(P1)} \quad \begin{aligned} \max_{\Omega} \quad & U(\theta_1, \Omega) \\ \text{s.t.} \quad & b_k \in \{0, 1\},\ k = 1,\ 2,\ ...,\ K, \\ & b_{k+1} \leq b_k,\ k = 1,\ 2,\ ...,\ K-1, \\ & |\mathcal{S}_k| = \min\{N - |\mathcal{W}_k|, Y_0\},\ k = 1,\ 2,\ ...,\ K, \\ & \mathcal{S}_k \subseteq (\mathcal{W} \setminus \mathcal{W}_k),\ k = 1,\ 2,\ ...,\ K. \end{aligned}$$

The complexity of solving Problem P1 using a conventional iterative algorithm (e.g., value iteration) is prohibitive due to the curse of dimensionality as the state space in (5) is not only high dimensional but also partially continuous. The difficulty is overcome in the sequel via two practical algorithms.

IV. ProgressFTX for CNN Classifiers

Given the complex architecture of a CNN model, ProgressFTX for a CNN classifier cannot be directly designed by an optimization approach. To overcome the difficulty, we leverage the following two principles, namely importance-aware feature selection and optimal stopping, to design the ProgressFTX control algorithms for the CNN model.

A. Importance-aware Feature Selection

Consider the feature selection step in the ProgressFTX protocol in Section III-A. We propose the use of a suitable metric of feature importance presented in [7], which is introduced next. Consider the parameters $\{w_m\}$ of the last CONV layer of the device sub-model. Let $\frac{\partial \mathcal{L}}{\partial w_m}$ denote the m-th entry of the gradient, i.e., the partial derivative of the learning loss function $\mathcal{L}$ w.r.t. parameter w_m, which is available from the last round

of model training as computed using the back-propagation algorithm. Then its importance can be measured using the associated reduction in learning loss from retaining w_m, approximated by the following first-order Taylor expansion of the squared loss of prediction errors [7]: $\tilde{I}(m) = \left(\frac{\partial \mathcal{L}}{\partial w_m} \cdot w_m\right)^2$. The importance of the n-th feature map is defined to be the summed importance of parameters in the n-th filter outputting the map: $g(n) = \sum_{w_m \in \text{ the } n\text{-th filter}} \tilde{I}(m)$. Since $\left\{\frac{\partial \mathcal{L}}{\partial w_m}\right\}$ are readily available in training, the server is able to obtain the value of $\{g(n)\}$ after training $f_{\mathsf{d}}(\cdot)$ and form a lookup table for reference during ProgressFTX control. Given $\{g(n)\}$, importance-aware feature selection for each slot is performed by selecting Y_0 most important filters, whose feature maps will be transmitted in the slot. It should be reiterated that the number of selected feature maps is communication-rate dependent and may vary over slots.

B. Stopping Control Based on Uncertainty Prediction

The optimal stopping control for a CNN model is stymied by the difficulty in finding a tractable yet accurate approximation of the expected classification uncertainty. To tackle the challenge, we resort to an algorithmic approach in which a regression model is trained to predict the uncertainty function of feature maps to be transmitted in the following K slots given the feature maps already received. One particular architecture of regression models in the literature is adopted [10]. To be able to predict the uncertainty of a future partial feature-map tensor, both the current partial feature-map tensor and the selected index subset are needed. Since the prediction is based on the current partial feature maps, ProgressFTX control for CNN models is sample-dependent. The architecture contains concatenation of two streams of regression features extracted from the feature-map tensor, $\mathbf{X}_k$, and the index subset $\mathcal{S}_k$, into one intermediate feature map, fed into the deeper layers of the regression model (see Fig. 3).

To train the regression model, a training dataset and a prediction loss function need to be properly designed, given the trained classification model and its dataset. The training dataset on the server, denoted as $\mathcal{D}$, comprises labeled samples $\{\mathbf{D}_i, H_i\}$, each with the a tuple of $\mathbf{D}_i = (\mathbf{X}_{(i)}, \mathcal{S}_{(i)})$ and a scalar label H_i, $i = 1, 2, ..., |\mathcal{D}|$. Consider a tensor of all feature maps extracted by the server sub-model $f_{\mathsf{d}}(\cdot)$ from an arbitrary sample, denoted by $\mathbf{X}$. The first entry in $\mathbf{D}_i$, $\mathbf{X}_{(i)}$, is a tensor of arbitrary partial feature maps drawn from $\mathbf{X}$, which represents the feature maps already received by the server. The second entry $\mathcal{S}_{(i)}$ is an admissible subset of indexes representing feature maps to be transmitted (hence not in $\mathbf{X}_{(i)}$). Then the label H_i is the exact inference uncertainty generated by the server sub-model $f_{\mathsf{s}}(\cdot)$ for the input of a tensor $\mathbf{X}'_{(i)}$ comprising both feature maps in $\mathbf{X}_{(i)}$ and the feature maps indexed in $\mathcal{S}_{(i)}$ drawn from $\mathbf{X}$, i.e., $H_i = -\mathbb{E}_\ell \left[\Pr\left(\ell|\mathbf{X}'_{(i)}\right) \log \Pr\left(\ell|\mathbf{X}'_{(i)}\right)\right]$. Next, to train the prediction model, the loss function is designed to be the mean-square error between the predicted result and the ground-truth. A *stochastic gradient descent* (SGD) optimizer is adopted to train the model, denoted as $f_{\mathsf{r}}(\cdot|\mathbf{W}_{\mathsf{r}})$ with parameters $\mathbf{W}_{\mathsf{r}}$, by minimizing the loss function:

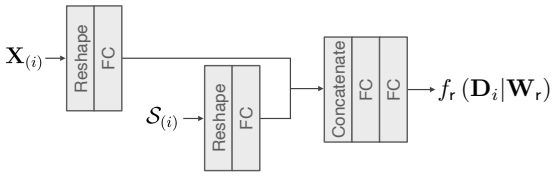


Fig. 3. Illustration of the architecture for regression models.

$$\min_{\mathbf{W}_{\mathsf{r}}} \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \left[f_{\mathsf{r}}(\mathbf{D}_i|\mathbf{W}_{\mathsf{r}}) - H_i\right]^2. \tag{6}$$

Given the current state θ_1 and the trained inference uncertainty predictor $f_{\mathsf{r}}(\cdot|\mathbf{W}_{\mathsf{r}})$, the online stopping control problem is

$$\min_{k^* \in \{0,1,...,K\}} f_{\mathsf{r}}\left(\left(\mathbf{X}_1, \bigcup_{k=1}^{k^*} \mathcal{S}_k^\star\right)\middle|\mathbf{W}_{\mathsf{r}}\right) + c_0 k^*. \tag{7}$$

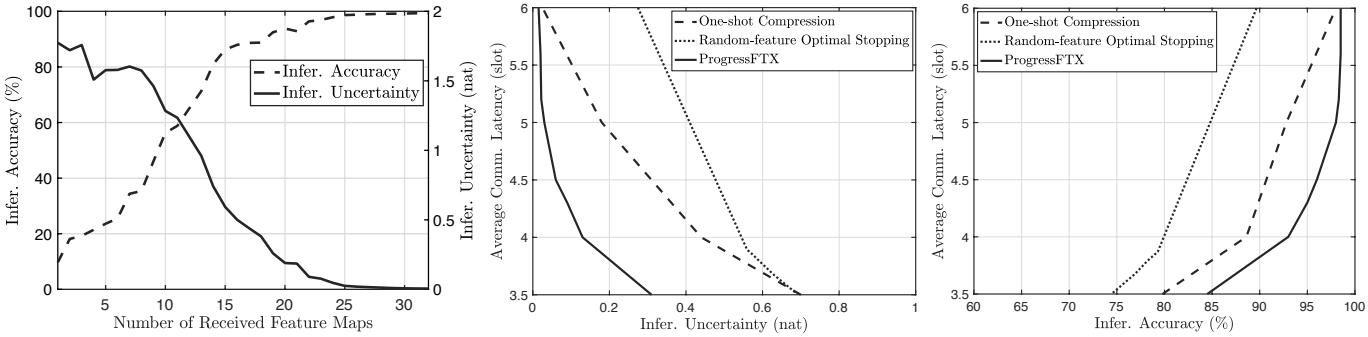
The sub-optimal stopping time from the current slot, namely $k^\star = \underset{k^* \in \{0,1,..,K\}}{\arg\min} f_{\mathsf{r}}\left(\left(\mathbf{X}_1, \bigcup_{k=1}^{k^*} \mathcal{S}_k^\star\right)\middle|\mathbf{W}_{\mathsf{r}}\right) + c_0 k^*$, can be determined by linear search. Then the stopping decision in the current slot is $b_1^\star = \min\{1, k^\star\}$. Even for a Gaussian channel, the number of transmission slots for different samples differ due to their requirements of feeding different numbers of features into classifier to reach the same target confidence level if possible.

V. Performance Evaluation

A. Experimental Settings

The experimental setups are designed as follows, unless specified otherwise. Each feature is quantized at a high resolution, $Q = 64$ bits/feature. The horizon in online scheduling is set as $K = 5$ slots. We use the popular MNIST dataset of handwritten digits for training and testing the well-known CNN model LeNet as in [7]. The Gaussian channel has a bandwidth of $B = 2.6$ MHz, the slot duration of $T = 10$ milliseconds, and the channel SNR $= 4$ dB. The corresponding transmission rate is $Y_0 = 4$ feature-maps/slot. As illustrated in Fig. 3, the uncertainty predictor designed in Section IV comprises two input layers. The first one reshapes an input tensor for partial feature maps into a 512×1 vector. The second one reshapes the feature selection input into a 256×1 vector, where the coefficient is 1 if the corresponding feature map index is selected or 0 otherwise. These two vectors are concatenated and then fed into three FC layers with 120, 40, and 10 neurons, respectively. There is one neuron in the last output layer providing a prediction of uncertainty. The test mean-square error of the uncertainty predictor trained for 50 epochs is low to 0.1.

Two benchmarks are considered. The first one, termed *one-shot compression*, uses the classic approach of model compression: given importance-aware feature selection, the number of features to transmit, $Y_0 k^\star$, is determined prior to transmission such that it meets an uncertainty requirement H_0. The value of



(a) Effect of the Number of Received Feature Maps (b) Inference Uncertainty (c) Inference Accuracy

Fig. 4. Effect of the number of received feature maps on CNN classification (a), and comparison of average communication latency between ProgressFTX and benchmarks with varying target (b) inference uncertainty or (c) inference accuracy.

$k^\star$ is solved from $\text{argmin}_{k^*} f_{\mathsf{r}}\left(\left(\mathbf{0}, \bigcup_{k=1}^{k^*} \mathcal{S}_k^\star\right)|\mathbf{W}_{\mathsf{r}}\right) \geq H_0$. This scheme lacks transmit/stop feedback. The second one, termed *random-feature optimal stopping*, modifies the sub-optimal ProgressFTX by removing feature importance awareness and instead selecting features randomly.

B. CNN Classification

To implement split inference, the split point of LeNet is chosen to be right after the second CONV layer in the model. As a result, the device can choose from 32 4×4 feature maps for transmission to the server. Given these settings, at most 8 slots are required to transmit all maps. First, the curves of inference performance (i.e., accuracy and uncertainty) versus the number of transmitted feature maps, selected with importance awareness, are plotted in Fig. 4(a). Particularly, the accuracy is observed to rapidly grow and the uncertainty quickly reduces as this number increases. The result validates the effectiveness of importance-aware feature selection for shortening the communication duration given target inference accuracy (or expected uncertainty).

We define the *average communication latency* of a transmission scheme as the average number of transmission slots required to meet the target inference accuracy or expected uncertainty. The average communication latency of ProgressFTX and two benchmarks are compared for varying target accuracy and expected uncertainty levels in Fig. 4. ProgressFTX is observed to outperform the benchmarks over the considered ranges of inference uncertainty and accuracy. For instance, given 4 transmission slots on average, ProgressFTX achieves the uncertainty of 0.13 and accuracy of 93% which are at least 65% lower and 6.9% higher than the benchmarks. For the target accuracy of 93%, one can observe from Fig. 4(c) that ProgressFTX requires on average 4 slots while one-shot compression requires 5 slots, corresponding to 20% latency reduction for the former.

We define the *transmission probability* of a feature map as the fraction of samples whose accurate inference requires the transmission of that feature map. For further comparison, the transmission probabilities of different feature maps are plotted against their importance levels in Fig. 5. One can observe that the transmission probability is almost uniform over unpruned/all feature dimensions for the one-shot/random-feature optimal stopping scheme. This indicates their lack of feature importance awareness. In contrast, the probabilities for ProgressFTX are highly skewed with higher probabilities for more important feature maps and vice versa. The skewness arises from the importance-aware feature selection as well as the stochastic control of transmissions which are the key reasons for the performance gain of ProgressFTX over the benchmark schemes.

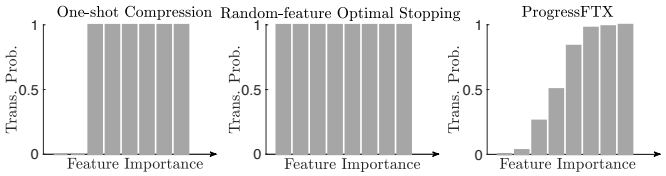


Fig. 5. The transmission probabilities of feature maps for the case of CNN classification and Gaussian channel with target inference accuracy of 98.5%.

REFERENCES

[1] M. Chen, D. Gündüz, K. Huang, W. Saad, M. Bennis, A. V. Feljan, and H. V. Poor, "Distributed learning in wireless networks: Recent progress and future challenges," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3579–3605, 2021.

[2] G. Zhu, D. Liu, Y. Du, C. You, J. Zhang, and K. Huang, "Toward an intelligent edge: Wireless communication meets machine learning," *IEEE Commun. Mag.*, vol. 58, no. 1, pp. 19–25, 2020.

[3] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge AI: On-demand accelerating deep neural network inference via edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 447–457, 2020.

[4] J. Shao and J. Zhang, "Communication-computation trade-off in resource-constrained edge inference," *IEEE Commun. Mag.*, vol. 58, no. 12, pp. 20–26, 2020.

[5] M. Jankowski, D. Gündüz, and K. Mikolajczyk, "Wireless image retrieval at the edge," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 89–100, 2021.

[6] J. Guo, W. Ouyang, and D. Xu, "Channel pruning guided by classification loss and feature importance," in *Proc. AAAI Conf. Artificial Intell. (AAAI)*, New York, NY, USA, Feb. 7-11, 2020.

[7] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, "Importance estimation for neural network pruning," in *Proc. IEEE/CVF Conf. Comput. Vision Pattern Recogn. (CVPR)*, Long Beach, CA, USA, Jun. 16-20, 2019.

[8] S. Lin and P. Yu, "A hybrid ARQ scheme with parity retransmission for error control of satellite channels," *IEEE Trans. Commun.*, vol. 30, no. 7, pp. 1701–1719, 1982.

[9] D. Liu, G. Zhu, Q. Zeng, J. Zhang, and K. Huang, "Wireless data acquisition for edge learning: Data-importance aware retransmission," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 406–420, 2021.

[10] D. F. Specht, "A general regression neural network," *IEEE Trans. Neural Netw.*, vol. 2, no. 6, pp. 568–576, 1991.