Progressive Feature Transmission for Split Classification at the Wireless Edge

Qiao Lan, *Graduate Student Member, IEEE*, Qunsong Zeng, *Member, IEEE*, Petar Popovski, *Fellow, IEEE*, Deniz Gündüz, *Fellow, IEEE*, and Kaibin Huang, *Fellow, IEEE*

***Abstract*—We consider the scenario of inference at the *wireless edge*, in which devices are connected to an edge server and ask the server to carry out remote classification, that is, classify data samples available at edge devices. This requires the edge devices to upload high-dimensional features of samples over resource-constrained wireless channels, which creates a communication bottleneck. The conventional feature pruning solution would require the device to have access to the inference model, which is not available in the current split inference scenario. To address this issue, we propose the *progressive feature transmission* (ProgressFTX) protocol, which minimizes the overhead by progressively transmitting features until a target confidence level is reached. A control policy is proposed to accelerate inference, comprising two key operations: *importance-aware feature selection* at the server and *transmission-termination control*. For the former, it is shown that selecting the most important features, characterized by the largest discriminant gains of the corresponding feature dimensions, achieves a sub-optimal performance. For the latter, the proposed policy is shown to exhibit a threshold structure. Specifically, the transmission is stopped when the incremental uncertainty reduction by further feature 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 control policy is first derived for the tractable case of linear classification, and then extended to the more complex case of classification using a *convolutional neural network*. Both Gaussian and fading channels are considered. Experimental results are obtained for both a statistical data model and a real dataset. It is shown that ProgressFTX can substantially reduce the communication latency compared to conventional feature pruning and random feature transmission strategies.**

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* [1], [2], where data are used to train large-scale AI models via distributed machine learning; and (2) *edge inference* [1], [3], which is the theme of this work and deals with operating of such models at edge servers to provide remote-inference services that enable emerging mobile applications, such as e-commerce or smart cities. For instance, a large-scale remote classifier (e.g., Google or Tencent Cloud) can empower a device to classify hundreds of object classes. To reduce communication overhead and protect data privacy, a typical edge-inference algorithm involves a device extracting features from raw data and transmitting them to an edge server for inference. To improve the communication efficiency, we propose and optimize a simple protocol, termed *progressive feature transmission* (ProgressFTX).

The state-of-the-art edge learning algorithms build upon an architecture termed *split inference* [4]–[10], in which the model is partitioned into *device* and *server* sub-models [5], [7]. 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. The device-server splitting of the computation load can be adjusted by shifting the splitting point of the model. It is proposed in [6] to adapt the splitting point to the communication rate so as to meet a latency requirement. Split inference can support adaptive model selection via compression of a full-size *deep neural network* (DNN) model into numerous reduced versions [5]. The hostility of the wireless link between the device and server introduces a communication bottleneck for split inference. This issue is addressed by the framework of joint source-and-channel coding developed over a series of works [7]–[10], featuring the use of an autoencoder pair of encoder and decoder as device and server sub-models, jointly trained to simultaneously perform inference and efficient transmission. In view of prior works, there is a lack of a communication-efficient solution that can adapt to a practical time-varying channel. Particularly, with the exception of [9], [10], the aforementioned joint source-and-channel coding approaches assume a static channel and require model retraining whenever the propagation environment changes significantly. On the other hand, the joint source-and-channel coding schemes addressing fading channels are applicable to analog transmission only [9], [10]. While feature transmission, as opposed to raw data uploading, leads to substantial overhead reduction, the communication bottleneck still exists due to the high dimensionality of features

The work of K. Huang described in this paper was substantially supported by a fellowship award from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. HKU RFS2122-7S04). The work was also supported by Guang-dong Basic and Applied Basic Research Foundation under Grant 2019B1515130003, Hong Kong Research Grants Council under Grants 17208319, and Shenzhen Science and Technology Program under Grant JCYJ20200109141414409. The work of P. Popovski was, in part, supported by the Villum Investigator Grant "WATER" from the Velux Foundation, Denmark. D. Gündüz received support from the UK EPSRC (grants no. EP/T023600/1 and EP/W035960/1) under the CHIST-ERA program. This article was presented in part at the 2022 IEEE International Workshop on Signal Processing Advances in Wireless Communication (SPAWC). Corresponding author: K. Huang.

Q. Lan, Q. Zeng, and K. Huang are with Department of Electrical and Electronic Engineering at The University of Hong Kong, Hong Kong SAR, China (Email: {qlan, qszeng, huangkb}@eee.hku.hk).

P. Popovski is with Department of Electronic Systems at Aalborg University, Aalborg, Denmark (Email: petarp@es.aau.dk).

D. Gündüz is with Department of Electrical and Electronic Engineering at Imperial College London, London, UK (Email: d.gunduz@imperial.ac.uk).

required for satisfactory inference performance [11]. For instance, GoogLeNet, a celebrated *convolutional neural network* (CNN) model, generates 256 28×28 feature maps at the output of a convolutional (inception) layer, resulting in a total of 2×10^6 real coefficients [12]. One solution for overcoming the communication bottleneck is to prune features according to their heterogeneous importance levels [7], [9], [13]. There are several representative approaches for importance-aware pruning in DNN models. The first approach is to evaluate the importance of a feature by observing the effect of its removal on the inference performance [13]–[15]. The second approach devises explicit importance measures, including the sum of cross-entropy loss and reconstruction error [14], Taylor expansion of the error induced by pruning [15], and symmetric divergence [16]. These algorithms outperform the naïve approach that assigns a higher importance to a larger parametric magnitude. The last approach, known as learning-driven coding, relies on an auto-encoder to intelligently identify and encode discriminant features to improve the communication efficiency [7], [10]. The proposed ProgressFTX protocol builds on the existing split-inference architecture to reduce the communication overhead via progressive transmissions targeting the specific task of classification. Such a design promises to achieve a higher efficiency than the existing one-shot feature selection/pruning, by leveraging both the importance awareness in feature selection as well as stochastic control according to the channel state.

Here we consider the scenario in which split inference is to be deployed to perform classification tasks, termed *split classification*, that finds a large number of applications ranging from surveillance to autonomous driving. For a given data sample, its classification accuracy improves as more features are sent to the server. This fact naturally motivates ProgressFTX to reduce the communication overhead. Specifically, based on the protocol, a device progressively transmits selected subsets of features until a target classification accuracy is reached, as informed by the server, or the expected communication cost becomes too high. The principle of progressive transmission also underpins *automatic repeat request* (ARQ), a basic mechanism for reliability in wireless networks. The reliability is achieved by repeating the transmission of a data packet until it is successfully received. Among the established communication protocols, ProgressFTX is most similar to *hybrid ARQ* (HARQ) with incremental redundancy [17]. Integrated with forward error correction, HARQ sequentially transmits multiple punctured versions of the encoder output for the same input packet. Upon retransmission, the reliability of the received packet is checked by combining all received versions and performing error detection on the combined result, thereby progressively improving the reliability. Despite the similarity, ProgressFTX differs from HARQ in two main aspects. First, ProgressFTX is a cross-disciplinary design aiming at achieving both a high classification accuracy and low communication overhead, while HARQ targets communication reliability. Second, the incremental redundancy in ProgressFTX differs from that for HARQ and refers to the use of an increasing number of features to improve classification accuracy. For example, using 16 feature maps from MNIST leads to an accuracy of 93% and additional 4 feature maps boost the accuracy to 98%. As a result, 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 proposal of the ProgressFTX protocol and the design of a sub-optimal, yet efficient policy for controlling the protocol. While a transmitted feature reduces the uncertainty in classification, its transmission incurs communication cost. This motivates the optimization of ProgressFTX, which is formulated as a problem of optimal stochastic control with dual objectives of minimizing both the uncertainty and communication cost. The problem is solved analytically for the case of linear classifier and the results are extended to a general model (e.g., CNN).

- *ProgressFTX for linear classifiers:* The optimization problem is decomposed into two sub-problems that reflect the two main operations at the server: feature selection and optimal stopping problems. The key idea to get tractable solutions is to approximate the commonly used entropy of posteriors by constructing a sufficiently tight upper bound as a convex function of the differential Mahalanobis distance from a feature vector to each pair of class centroids of the data distribution. By solving this approximate version of the original problem, the resultant sub-optimal strategy of the server is to follow the order of decreasing feature importance as measured by its entropy. The optimal stopping problem is solved for two channel types, Gaussian and fading with no CSIT, respectively. The sub-optimal policies for both are derived in closed form and are observed to exhibit a simple threshold based structure. The threshold based policies specify criteria for stopping ProgressFTX when the incremental reward (i.e., uncertainty reduction) by further feature transmission is too small, or the expected communication cost outweighs the reward.
- *ProgressFTX for CNN classifiers*: The problems of importance-aware feature selection and optimal stopping for CNN classifiers are more complex than the case of linear classification due to the complexity of CNNs, rendering an analytical approach intractable. We tackle the challenge by designing and numerically evaluating two practical algorithms. First, we evaluate the importance of a feature map using the gradients of associated model parameters generated in the process of training the inference model. Second, we advocate the use of a low-complexity regression model trained to predict the inference uncertainty. The model allows finding the sub-optimal stopping time of ProgressFTX by a simple linear search.

Experiments using both statistical and real datasets have been conducted. The results demonstrate that ProgressFTX with importance-aware feature selection and transmission termination can substantially reduce the number of channel uses when benchmarked against the schemes of conventional one-shot feature compression or ProgressFTX with random feature selection.

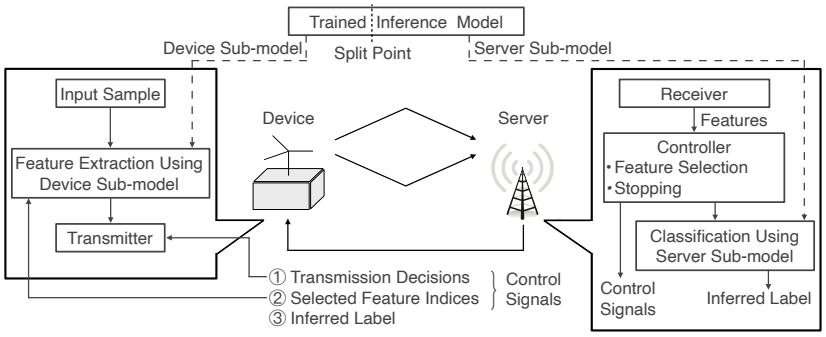


Fig. 1. An edge inference system.

II. Models and Metrics

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

A. Data Distribution Model

While CNN-based classification targets a generic distribution, in the case of linear classification, for tractability, we assume the following data distribution. Each M-dimensional data sample is assumed to follow a *Gaussian mixture* (GM) distribution and is associated with one of L classes. Note that the GM model is widely used to model clustered data distribution in practice, e.g., biomedical and speech processing applications [16], [18]. We assume that the L classes have uniform priors. Given the data distribution, the server computes the N-dimensional feature space with $N \leq M$ using *principal components analysis* (PCA) [18]. At the beginning of the edge-inference process from Fig. 1, the server sends the feature space to the device for the purpose of feature extraction: a feature vector is extracted from each sample by projection onto this feature space. This results in an arbitrary feature vector, denoted as $\mathbf{x} \in \mathbb{R}^N$, distributed as a GM in the feature space. Each class ℓ is a multivariate Gaussian distribution $\mathcal{N}(\mu_\ell, \mathbf{C}_\ell)$ with a centroid $\mu_\ell \in \mathbb{R}^N$ and a covariance matrix $\mathbf{C}_\ell \in \mathbb{R}^{N\times N}$. For tractability and still in practical relevance [19], the covariance matrices are assumed identical [18]: $\mathbf{C}_\ell = \mathbf{C}$, $\forall \ell$, where $\mathbf{C}$ is diagonalized by PCA and known to the server. Then the *probability density function* (PDF) of $\mathbf{x}$ can be written as

$$f(\mathbf{x}) = \frac{1}{L}\sum_{\ell=1}^{L} \mathcal{N}\left(\mathbf{x}|\mu_\ell, \mathbf{C}\right), \tag{1}$$

where $\mathcal{N}\left(\mathbf{x}|\mu_\ell, \mathbf{C}\right)$ denotes the Gaussian PDF with mean μ_ℓ and covariance $\mathbf{C}$.

B. Communication Model

Consider uplink transmission of feature vectors for remote inference at the server. Each feature is quantized with a sufficiently high resolution of Q bits such that quantization errors are negligible. We consider both the cases of linear and CNN classification that are elaborated in the next subsection. The coherence time of the communication channel is divided into slots, each spanning T seconds.[1] In the former, the number of features that can be transmitted in slot k is given by $Y_k = \lfloor \frac{R_k T}{Q} \rfloor$, where R_k is the communication rate in bits/second. We refer to Y_k as the transmission rate in features/slot. In the case of CNN classification, the basic unit of transmitted data is a *feature map*, which is a $L_{\mathsf{h}} \times L_{\mathsf{w}}$ matrix. Correspondingly, the transmission rate (in feature-maps/slot) for a slot can be written as $Y_k = \lfloor \frac{R_k T}{Q L_{\mathsf{h}} L_{\mathsf{w}}} \rfloor$. The rounding effect is negligible by assuming a sufficiently high communication rate such that a relatively large number of features or feature maps can be transmitted within each slot (e.g. $Y_0 > 10$ features/slot or feature-maps/slot). For both linear and CNN classification, the number of successfully transmitted features (or feature maps) depends on the channel and the number of features not received yet, and thus may vary from slot to slot.

The device is allocated a narrow-band channel with the bandwidth denoted as B and the gain in slot k as g_k. Two types of channels are considered. The first is the Gaussian channel with a static channel gain, where $g_k = g_0$ for all k and is known to both the transmitter and the receiver. The transmit *signal-to-noise ratio* (SNR) is fixed as ρ and the rate is matched to the channel to be $R_0 = B\log_2(1+\rho g_0)$. The transmission rate is then fixed as $Y_k = Y_0 = \lfloor \frac{R_0 T}{Q} \rfloor$. The second one is fading channel, where g_k is an *independent and identically distributed* (i.i.d.) random variable over slots. It is assumed that the transmitter does not know the channel gains $\{g_k\}$ and uses a fixed rate R_0. An outage occurs if the channel gain falls below g_0 [21]. Let the outage probability be denoted by $p_{\mathsf{o}} = \Pr(g_k < g_0)$. As a final remark, note that the number of features that are available to the server is known with the Gaussian model, while it is unpredictable with the fading model due to random outage events.

C. Two Classification Models

We consider two classification models with progressive feature transmission that the device transmits a subset of features (feature maps) in each slot.

1) Linear Classification Model: Let $x(n)$ denote the n-th feature (coefficient) of an arbitrary feature vector $\mathbf{x}$, and $\mathcal{W}_k \subseteq \mathcal{W}$ denote the cumulative index set of features received by

[1]The channel coherence time is typically not larger than several milliseconds, which is smaller than or comparable with the practical latency requirements of split inference applications ranging from 10 ms to several seconds [20].

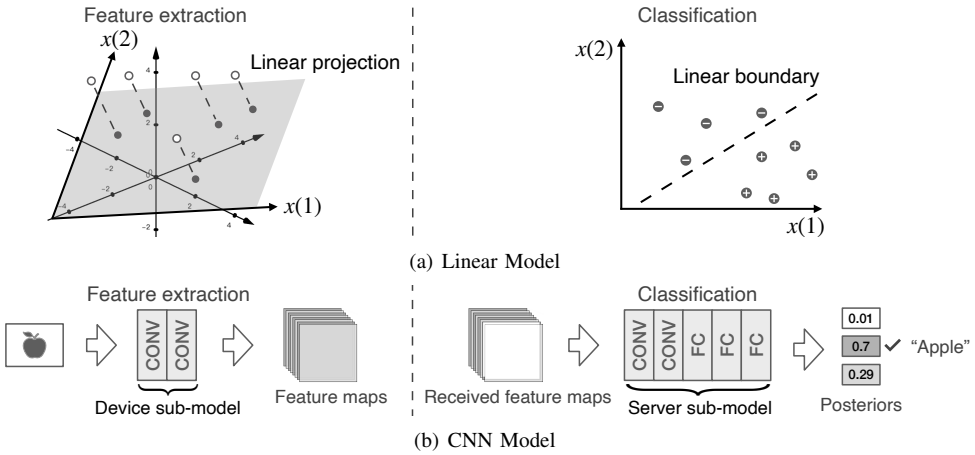


Fig. 2. Linear and CNN classification models for split inference.

the server by the end of slot k, where $\mathcal{W} = \{1, 2, ..., N\}$, respectively. The corresponding *partial* feature vector, denoted as $\mathbf{x}_k$, can be defined by setting its coefficients as

$$x_k(n) = \begin{cases} x(n), & n \in \mathcal{W}_k, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

In each slot, the server can estimate the label of the object $\mathbf{x}$ by inputting the partial vector $\mathbf{x}_k$ into a classifier.

The *maximum likelihood* (ML) classifier for the distribution (1) is a linear one [18], and relies on a classification boundary between a class pair being a hyper-plane in the feature space (see Fig. 2(a)). Due to uniform prior on the object classes, the ML classifier is equivalent to a *maximum a posterior* (MAP) classifier and the estimated label $\hat{\ell}$ is obtained as

$$\begin{aligned} \hat{\ell} &= \underset{\ell}{\text{argmax}} \ \Pr(\mathbf{x}_k|\ell, \mathcal{W}_k) \\ &= \underset{\ell}{\text{argmax}} \ \exp\left(-\frac{1}{2}\sum_{n\in\mathcal{W}_k} \frac{[x_k(n) - \mu_\ell(n)]^2}{C_{n,n}}\right), \end{aligned} \tag{3}$$

where $C_{n,n}$ denotes the n-th diagonal entry of the covariance matrix $\mathbf{C}$. The *Mahalanobis distance* refers to the distance from the sample $\mathbf{x}_k$ to the centroid of class-k in the $|\mathcal{W}_k|$-dimensional feature subspace. Given the ground-truth label for $\mathbf{x}_k$ being ℓ, let $z_\ell(k)$ denote the half squared Mahalanobis distance: $z_\ell(k) \triangleq \frac{1}{2}\sum_{n\in\mathcal{W}_k} \frac{[x_k(n)-\mu_\ell(n)]^2}{C_{n,n}}$. Then the problem of linear classification reduces to one of Mahalanobis distance minimization: $\hat{\ell} = \text{argmin}_\ell \ z_\ell(k)$.

2) 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 as shown in Fig. 2 (b), which are represented by functions $f_{\mathsf{d}}(\cdot)$ and $f_{\mathsf{s}}(\cdot)$, respectively. Following existing designs in [13], [14], 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 tensor of all extracted 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 as $\mathbf{X}(n)$. Let $\mathbf{X}_k$ denote the tensor of *cumulative* feature maps received by the server by slot k, and $\mathcal{W}_k$ the corresponding *cumulative* index set of received feature maps. Then $\mathbf{X}_k$ can be related to $\mathbf{X}$ as

$$\mathbf{X}_k(n) = \begin{cases} \mathbf{X}(n), & n \in \mathcal{W}_k, \\ \mathbf{0}, & \text{otherwise.} \end{cases} \tag{4}$$

We define the system state for this generic model as $\theta_k = (\mathbf{X}_k, \mathcal{W}_k)$. In slot k, the server sub-model can compute the posteriors $\Pr(\ell|\mathbf{X}_k)$ for the input $\mathbf{X}_k$ using the forward-propagation algorithm [3]. Then, the label $\hat{\ell}$ is estimated by posterior maximization: $\hat{\ell} = \text{argmax}_\ell \ \Pr(\ell|\mathbf{X}_k)$.

D. Classification Metrics

For linear and CNN classification, relevant metrics are characterized as follows.

1) Inference Uncertainty: Inference uncertainty is measured using the *entropy of posteriors* commonly used for ML and DNN classifiers [22], [23]. The metric, denoted as $H(\mathbf{x}_k)$ for a partial feature vector (or $H(\mathbf{X}_k)$ for a feature tensor), is given by

$$H(\mathbf{x}_k) \triangleq -\sum_{\ell=1}^{L} \Pr(\ell|\mathbf{x}_k, \mathcal{W}_k) \log \Pr(\ell|\mathbf{x}_k, \mathcal{W}_k).$$

In the particular case of linear classification with the model in (1), $H(\mathbf{x}_k)$ can be rewritten as

$$H(\mathbf{x}_k) = \log\left(\sum_{\ell=1}^{L} e^{-z_\ell(k)}\right) + \frac{\sum_{\ell=1}^{L} z_\ell(k) e^{-z_\ell(k)}}{\sum_{\ell=1}^{L} e^{-z_\ell(k)}}, \tag{5}$$

where the derivation is straightforward and thus omitted.

2) Discriminant Gain: For linear classification, pairwise discriminant gain measures the discernibility between two classes in a subspace of the feature space. We adopt the well known metric of *symmetric Kullback-Leibler (KL) divergence* defined as follows [16]. Consider a class pair, say classes ℓ and ℓ', and a feature subspace spanning the dimensions indexed by the set $\mathcal{S} = \{n_1, n_2, ..., n_{|\mathcal{S}|}\} \subseteq \mathcal{W}$. Let $G_{\ell,\ell'}(\mathcal{S})$ denote the corresponding pairwise discriminant gain. Based on the feature distribution model, the feature vector in this subspace is a

uniform mixture of L Gaussian components with mean $\tilde{\mu}_\ell$ and covariance matrix $\tilde{\mathbf{C}}$ ($\ell = 1, 2, ..., L$), where $\tilde{\mu}_\ell(i) = \mu_\ell(n_i)$ and

$$\tilde{C}_{i,j} = \begin{cases} C_{n_i,n_i}, & i = j, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

Based on the metric of symmetric KL divergence, pairwise discriminant gain is evaluated as,

$$\begin{aligned} G_{\ell,\ell'}(\mathcal{S}) &\triangleq \mathsf{KL}\left(\mathcal{N}(\tilde{\mu}_\ell, \tilde{\mathbf{C}}) \parallel \mathcal{N}(\tilde{\mu}_{\ell'}, \tilde{\mathbf{C}})\right) \\ &\quad + \mathsf{KL}\left(\mathcal{N}(\tilde{\mu}_{\ell'}, \tilde{\mathbf{C}}) \parallel \mathcal{N}(\tilde{\mu}_\ell, \tilde{\mathbf{C}})\right) \\ &= (\tilde{\mu}_\ell - \tilde{\mu}_{\ell'})^T \tilde{\mathbf{C}}^{-1} (\tilde{\mu}_\ell - \tilde{\mu}_{\ell'}) \\ &= \sum_{n \in \mathcal{S}} g_{\ell,\ell'}(n), \end{aligned} \tag{7}$$

where $g_{\ell,\ell'}(n) = \frac{[\mu_\ell(n) - \mu_{\ell'}(n)]^2}{C_{n,n}}$. The term $g_{\ell,\ell'}(n)$ is named the pairwise discriminant gain in dimension-n. We define the average discriminant gain over all $\frac{L(L-1)}{2}$ class pairs as

$$\bar{G}(\mathcal{S}) = \frac{2}{L(L-1)} \sum_{\ell < \ell' \le L} G_{\ell,\ell'}(\mathcal{S}) = \sum_{n \in \mathcal{S}} \bar{g}(n), \tag{8}$$

where $\bar{g}(n) = \frac{2}{L(L-1)} \sum_{\ell < \ell' \le L} g_{\ell,\ell'}(n)$ is the average discriminant gain of dimension-n. We refer to (8) as the *additivity* of discriminant gains. For binary classification, the subscripts of $g_{1,2}(n)$ and $G_{1,2}(\mathcal{S})$ are omitted for simplicity, yielding $g(n)$ and $G(\mathcal{S})$.

3) Feature Importance: The discriminant gain defined in (7) for a linear classifier, which underpins the associated metric of feature importance, is not applicable to a CNN model. In this case, we adopt another metric from [15]. 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. the 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 of learning loss from retaining w_m, approximated by the following first-order Taylor expansion of the squared loss of prediction errors [15]: $\tilde{I}(m) = \left(\frac{\partial \mathcal{L}}{\partial w_m} \cdot w_m\right)^2$.

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. 3 and described as follows.

1) *Feature selection:* At the beginning of a slot intended for feature transmission, the server selects the feature dimensions and informs the device to transmit corresponding features. For the case of linear classification, feature selection uses the importance metric of discriminant gain (see Section II-D); for the other case of CNN classification, the selection of feature maps depend on their importance levels (i.e., descent gradients as elaborated in Section VI-A) which are available from the training phase and to be used in the ensuing phase of edge inference. Two facts should be emphasized: 1) both methods are data-sample independent, which facilitates efficient implementation at the server since it has no access to samples; and 2) feature importance is evaluated using a trained model, but not during the training. Specifically, the server records received features in $\mathbf{x}_k$ and their indices in $\mathcal{W}_k \subseteq \mathcal{W}$. The indices of selected features 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: $|\mathcal{S}_k| = \min\{N - |\mathcal{W}_k|, Y_k\}$.
2) *Stopping control:* The number of features of a particular sample needed for remote classification to meet the accuracy requirement is sample dependent (for both linear and CNN classifications). 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 V. 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 derived policy for the case of linear classification (see Section V) or a regression model for uncertainty prediction (see Section VI-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 of one of three available control signals described as follows: *(a)* If the channel is not in outage in the preceding slot, the feedback contains a *transmit-new-features signal* and indices of features in $\mathcal{S}_k$ that instructs the device to transmit the features selected by the server in the current slot. *(b)* Otherwise, the feedback is a *retransmission signal* instructing the device to retransmit the features transmitted in the preceding slot. In this case, a link-layer HARQ protocol is employed to enable retransmission. *(c)* 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 vector, $\Delta\mathbf{x}_k$, that comprises the selected features:

$$\Delta\mathbf{x}_k = \left[x(n_1), x(n_2), ..., x(n_{|\mathcal{S}_k|})\right]^T, \tag{9}$$

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

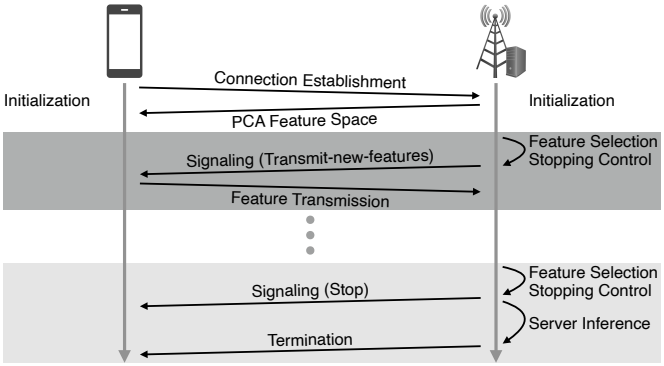


Fig. 3. Illustration of the ProgressFTX protocol.

$\mathcal{W}_{k+1} = \mathcal{W}_k \bigcup \mathcal{S}_k$ and the partial feature vector $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{M}\Delta\mathbf{x}_k$, where the entry in the i-th row and j-th column of the placement matrix $\mathbf{M}$ is

$$M_{i,j} = \begin{cases} 1, & j \leq |\mathcal{S}_k| \text{ and } i = n_j, \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

5) *Server inference:* The server infers an estimated label $\hat{\ell}$ and its uncertainty level using the partial feature vector $\mathbf{x}_k$ and a trained model as discussed in Section II-C. Then we go to Step 1 and 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.

Remark 1 (*ProgressFTX for General Inference Applications*)**.** The proposed ProgressFTX protocol is generic and applicable to other inference applications besides classification, e.g., regression. The extension of the current design to a different application is straightforward and involves replacing the feature importance measures (i.e., discriminant gain or uncertainty) and the inference performance metric (i.e. classification accuracy) with metrics suitable for the corresponding application. Taking regression as an example, the well-known ReliefF estimator [24] can be adopted to measure feature importance, while the performance metric can be the mean-squared error. The optimized control of ProgressFTX can be modified for new measures and metrics without changing the procedure or the design principle.

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

$$\theta_k \triangleq (\mathbf{x}_k, \mathcal{W}_k)\,. \tag{11}$$

Recall that Y_0 represents the transmission rate in slot k, $\mathbf{x}_k$ the received partial feature vector, and $\mathcal{W}_k$ the indices of received features. The control policy, denoted as Ω, maps θ_k to the feature-selection *action*, $\mathcal{S}_k$, and stopping-control action, b_k, $\Omega : \theta_k \rightarrow (\mathcal{S}_k, b_k)$. 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

$$u(\theta_k, \Omega) = H(\mathbf{x}_k) - H(\mathbf{x}_{k+1}) - b_k c_0, \tag{12}$$

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 or retransmisson; 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:

$$U(\theta_1, \Omega) \triangleq \mathbb{E}_{\{\theta_k\}_{k=2}^K} \left[\sum_{k=1}^{K} u(\theta_k, \Omega) \;\middle|\; \theta_1, \Omega \right]. \tag{13}$$

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 the problem using a conventional iterative algorithm (e.g., value iteration) is prohibitive due to the curse of dimensionality as the state space in (13) is not only high dimensional but also partially continuous. The difficulty is overcome in the sequel via analyzing the structure of the sub-optimal policy with proper approximations.

IV. ProgressFTX Control – Importance-Aware Feature Selection

In this section, targeting linear classification, a sub-optimal feature selection policy for the ProgressFTX protocol is obtained in closed form by approximating the objective function in Problem (P1).

A. Approximation of Expected Uncertainty

1) Binary Classification: For ease of exposition, we first consider the simplest case of binary classification (i.e., $L = 2$) before extending the results to the general case. The goal of the subsequent analysis is to approximate the expected uncertainty function of unknown features to be transmitted in the following K slots, given the features already received by the server,

which is the objective of Problem (P1) in (13). To begin with, the objective can be rewritten as

$$U(\theta_1,\Omega) = \mathbb{E}_{\{\Delta\mathbf{x}_k\}_{k=1}^K}\left[\sum_{k=1}^K H(\mathbf{x}_k) - H(\mathbf{x}_{k+1}) \;\middle|\; \theta_1,\Omega\right] - c_0\sum_{k=1}^K b_k.$$

Based on the fact that feature distributions in different dimensions are independent (as the covariance matrix is diagonal), telescoping gives

$$U(\theta_1,\Omega) = H(\mathbf{x}_1) - \mathbb{E}_{\{\Delta\mathbf{x}_k\}_{k=1}^K}\left[H(\mathbf{x}_{K+1}) \;\middle|\; \theta_1,\Omega\right] - c_0\sum_{k=1}^K b_k. \quad (14)$$

In the remainder of the sub-section, we show that the expected uncertainty term in $U(\theta_1,\Omega)$ can be reasonably approximated by a monotone decreasing function of the average discriminant gain of selected features, which facilitates the subsequent feature-selection optimization.

Consider first the feature-selection decisions for transmission over K slots specified by $\mathcal{S}=\bigcup_{k=1}^K \mathcal{S}_k$. Upon receiving the selected features $\Delta\mathbf{x}$, the partial feature vector $\mathbf{x}_1$ is updated as $\mathbf{x}_{K+1} = \mathbf{x}_1 + \mathbf{M}^{(K)}\Delta\mathbf{x}$, where the placement matrix $\mathbf{M}^{(K)}$, defined similarly as $\mathbf{M}$ in (10), places all the features transmitted over K slots as recorded in $\mathcal{S}$. For an arbitrary feature vector $\mathbf{x}$, define the differential-distance as $\delta(\mathbf{x}) = z_1(\mathbf{x}) - z_2(\mathbf{x})$, where z_1 and z_2 represent the half squared Mahalanobis distances from $\mathbf{x}$ to the centroids of classes 1 and 2, respectively. For ease of notation, let $\delta(\mathbf{x}_1)$ and $\delta(\mathbf{x}_{K+1})$ be denoted as δ_1 and δ_{K+1}, respectively. In particular,

$$\delta_{K+1} = \sum_{n\in(\mathcal{W}_1\bigcup\mathcal{S})} \frac{[\mu_2(n)-\mu_1(n)][2x(n)-\mu_2(n)-\mu_1(n)]}{2C_{n,n}}. \quad (15)$$

Note that δ_{K+1} is a random variable from the server's perspective before receiving the selected features. Given the GM model (1), the distribution of δ_{K+1} is characterized next, followed by a simplified expression of uncertainty (17), while the derivation is straightforward, and thus it is omitted.

Lemma 1. Given the partial feature vector $\mathbf{x}_1$ and feature-selection decision $\mathcal{S}$, the differential distance δ_{K+1} is a mixture of two Gaussian components with the following conditional PDF:

$$f_{\delta_{K+1}}(\delta|\mathbf{x}_1,\mathcal{S}) = \frac{1}{2}\Bigg[\mathcal{N}\left(\delta \;\middle|\; \delta_1+\frac{G(\mathcal{S})}{2}, G(\mathcal{S})\right) + \mathcal{N}\left(\delta \;\middle|\; \delta_1-\frac{G(\mathcal{S})}{2}, G(\mathcal{S})\right)\Bigg], \quad (16)$$

where $G(\mathcal{S}) = G_{1,2}(\mathcal{S})$ is the total discriminant gain of features selected in $\mathcal{S}$ given in (7).

Given a differential distance δ, the inference uncertainty $H(\mathbf{x})$ in (5) can be simplified as:

$$H(\delta) = \log\left(1+e^{-\delta}\right) + \frac{\delta}{e^{\delta}+1}. \quad (17)$$

Then the expected uncertainty resulting from the feature-selection decision $\mathcal{S}$ can be written as

$$\mathbb{E}\left[H(\delta_{K+1}) \;\middle|\; \theta_1,\mathcal{S}\right] = \int_{-\infty}^{\infty} H(\delta) f_{\delta_{K+1}}(\delta|\mathbf{x}_1,\mathcal{S})\,\mathrm{d}\delta \triangleq \bar{H}(\delta_1, G(\mathcal{S})). \quad (18)$$

In view of the difficulty in deriving the above integral in closed form, we resort to its approximation using an upper bound as shown in the following lemmas.

Lemma 2. Given $\delta\in\mathbb{R}$, the inference uncertainty $H(\delta)$ can be upper bounded by

$$H^{\mathsf{ub}}(\delta) = (1+|\delta|)e^{-|\delta|}. \quad (19)$$

Proof. The result follows from the fact that the ratio $\frac{H(\delta)}{H^{\mathsf{ub}}(\delta)}$ has the maximum value of $\log 2$, which is smaller than one, at $\delta=0$. □

For large δ, the bound is tight since $\lim_{|\delta|\to\infty}\frac{H(\delta)}{H^{\mathsf{ub}}(\delta)}=1$. Using Lemma 2, we can bound the expected uncertainty as $\bar{H}(\delta_1,G(\mathcal{S}))\le\bar{H}^{\mathsf{ub}}(\delta_1,G(\mathcal{S}))$ with

$$\bar{H}^{\mathsf{ub}}(\delta_1,G(\mathcal{S})) = \int_{-\infty}^{\infty} H^{\mathsf{ub}}(\delta) f_{\delta_{K+1}}(\delta|\mathbf{x}_1,\mathcal{S})\,\mathrm{d}\delta, \quad (20)$$

where $H^{\mathsf{ub}}(\delta)$ is given in Lemma 2. The above analysis allows approximating $\bar{H}(\delta_1,G(\mathcal{S}))$ by a simple exponential upper bound as shown in Lemma 3, proven in Appendix A. This approximation is also validated numerically in Fig. 4.

Lemma 3. The expected uncertainty can be upper bounded by an exponential function as

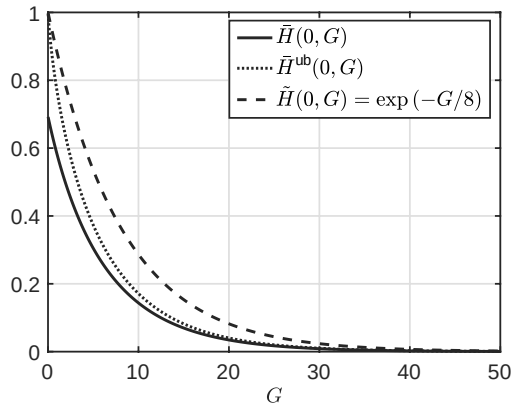
$$\bar{H}(\delta_1,G(\mathcal{S})) \le \bar{H}^{\mathsf{ub}}(\delta_1,G(\mathcal{S})) \le \tilde{H}(\delta_1,G(\mathcal{S})) \triangleq c_1 e^{-c_2 G(\mathcal{S})}, \quad (21)$$

where c_1 and c_2 are positive constants depending on δ_1.

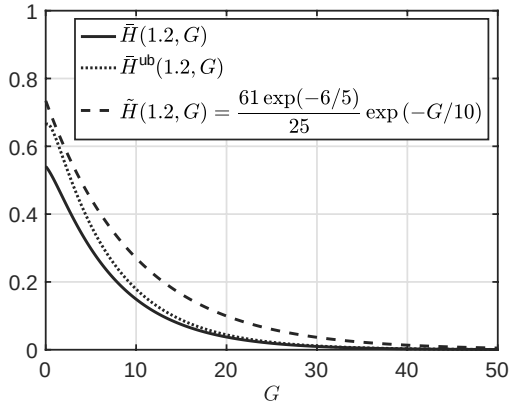
2) Extension to Multi-class Classification: The above analysis for binary classification can be generalized for multi-class classification (i.e., $L\ge 3$) as follows. The inference uncertainty in the case of multi-class classification can be upper bounded in terms of pairwise uncertainties,

$$H(\mathbf{x}_k) \le \sum_{\ell=1}^{L-1}\sum_{\ell'=\ell+1}^{L}\log\left(e^{-z_\ell(k)}+e^{-z_{\ell'}(k)}\right) + \frac{z_\ell(k)e^{-z_\ell(k)}+z_{\ell'}(k)e^{-z_{\ell'}(k)}}{e^{-z_\ell(k)}+e^{-z_{\ell'}(k)}} = \sum_{\ell=1}^{L-1}\sum_{\ell'=\ell+1}^{L} H\left(\delta_k^{(\ell,\ell')}\right),$$

where $\delta_k^{(\ell,\ell')}\triangleq z_\ell(k)-z_{\ell'}(k)$. The bound follows from well-known entropy inequalities [25], which state that the joint entropy of $\frac{L(L-1)}{2}$ pairwise estimated labels is not smaller than the entropy of the estimated label (as a function of the former), and not larger than the sum of the entropies of the pairwise



(a) $\delta_1 = 0$



(b) $\delta_1 = 1.2$

Fig. 4. Comparisons between the expected inference uncertainty and its upper bounds in (21) for (a) $\delta_1 = 0$ or (b) $\delta_1 = 1.2$.

estimated labels, i.e., $H\left(\delta_k^{(\ell,\ell')}\right)$. The equality is approached in the asymptotic limit when there exists an infinitely small $\delta_k^{(\ell,\ell')}$ compared with other pairwise differential distances, i.e., degrades to binary classification. Moreover, the distribution of $\delta_k^{(\ell,\ell')}$ conditioned on a partial feature vector $\mathbf{x}_1$ and a feature-selection decision $\mathcal{S}$ is a mixture of L scalar Gaussian components, similar to its counterpart in binary classification (16). As a result, an upper bound on the conditional expected inference uncertainty can be obtained following the approach for binary classification, i.e., substituting $H\left(\delta_k^{(\ell,\ell')}\right)$ with $H^{\mathsf{ub}}\left(\delta_k^{(\ell,\ell')}\right)$ as that in (20). Given the above straightforward procedure, the remainder of the paper assumes binary classification to simplify notation and exposition.

B. Sub-Optimal Feature Selection

First, Problem (P1) reduces to the problem of optimal feature selection by conditioning on k^* subsequent transmissions, i.e., transmission is stopped at slot (k^*+1). The decision can be captured by defining a sequence of virtual transmission rates $\{\tilde{Y}_k\}$ such that $\tilde{Y}_k = Y_0$ for $k \leq k^*$ and $g_k \geq g_0$, while $\tilde{Y}_k = 0$ otherwise. Moreover, we approximate the term $\bar{H}(\delta_1, G(\mathcal{S}))$ in the objective function in Problem (P1) by its upper bound $\tilde{H}(\delta_1, G(\mathcal{S}))$ derived in Lemma 3. Given the stopping decision and approximate objective, Problem (P1) reduces to the following problem.

$$\text{(P2)}\quad \begin{aligned} \min_{\{\mathcal{S}_k\}_{k=1}^K} \quad & \tilde{H}(\delta_1, G(\mathcal{S})) \\ \text{s.t.} \quad & |\mathcal{S}_k| = \min\{N - |\mathcal{W}_k|, \tilde{Y}_k\},\ k = 1,2,...,K, \\ & \mathcal{S}_k \subseteq (\mathcal{W} \setminus \mathcal{W}_k)\,,\ k = 1,2,...,K, \\ & \mathcal{S} = \bigcup_{k=1}^K \mathcal{S}_k. \end{aligned}$$

It follows from the monotonicity of the approximate objective function that the sub-optimal policy for feature selection in each slot is to choose from not yet received features, whose indices correspond to the maximum discriminant gains. Let the importance of each feature be associated with its discriminant gain. Then the proposed policy, termed *importance-aware feature selection*, is as presented in Algorithm 1.

V. ProgressFTX Control – Transmission-Termination Control

Given the importance-aware feature selection process in the preceding section, we describe the remaining design of ProgressFTX to solve the problem of optimal stopping control. Following the notation used in Section IV-B, transmission is assumed to stop in slot (k^*+1), resulting in k^* transmitted incremental feature vectors. The stopping-control indicators $\{b_k\}$ and k^* can be related as

$$b_k = \begin{cases} 1, & \forall 1 \leq k \leq k^*, \\ 0, & \forall k^*+1 \leq k \leq K. \end{cases} \tag{22}$$

As before, for tractability, we approximate the objective in Problem (P1) (see (14)) using the upper bound $\tilde{H}(\delta_1, G(\mathcal{S}))$ in Lemma 3. Then given the optimally selected (w.r.t. the approximate objective) and transmitted features $\{\mathcal{S}_k^\star\}$, Problem (P1) can be approximated as

$$\text{(P3)}\quad \begin{aligned} \min_{k^*} \quad & \mathbb{E}_{\{g_k\}_{k=1}^K}\left[\tilde{H}(\delta_1, G(\mathcal{S}^\star)) \;\middle|\; \theta_1\right] + c_0 k^* \\ \text{s.t.} \quad & k^* \in \{0,1,...,K\}, \\ & \mathcal{S}^\star = \bigcup_{\substack{1\leq k\leq k^* \\ g_k \geq g_0}} \mathcal{S}_k^\star. \end{aligned}$$

Note the dependence of the equivalent optimal (for the approximate objective) feature selection $\mathcal{S}^\star$ on the random channel gain g_k. Problem (P3) belongs to the class of optimal-stopping problems [26]. In operation, the server makes a binary decision (i.e., $b_1 = 0$ or 1) on whether to transmit in the current slot. The solution for Problem (P3) can be converted into the control decision, b_1, by

$$b_1 = \min\{1, k^*\}. \tag{23}$$

The control policy (also known as the stopping rule) is derived for both the Gaussian and fading channels in the following subsections.

A. Transmission Termination with Gaussian Channels

In this subsection, we address the optimal stopping problem for Gaussian channels, where $g_k = g_0, \forall k$. In this case, the

Algorithm 1: Importance-aware Feature Selection at the Server

Input: Set of all feature indices $\mathcal{W}$, set of received feature indices $\mathcal{W}_1$, and virtual transmission rates $\{\tilde{Y}_k\}_{k=1}^{k^*}$;

Initialize: Selected feature subsets $\mathcal{S}_k^\star = \emptyset, \forall k = 1, 2, ..., k^*$;

for $k = 1, 2, \cdots, k^*$ **do**

1: *Feature selection for one slot.*

for $i = 1, 2, \cdots, \min\{N - |\mathcal{W}_k|, \tilde{Y}_k\}$ **do**

Select the most important feature from the admissible set $(\mathcal{W} \setminus \mathcal{W}_k \setminus \mathcal{S}_k^\star)$, and add the selected feature index to $\mathcal{S}_k^\star$, i.e.,

$$\mathcal{S}_k^\star := \mathcal{S}_k^\star \bigcup \left\{ \underset{n \in (\mathcal{W} \setminus \mathcal{W}_k \setminus \mathcal{S}_k^\star)}{\operatorname{argmax}} \; g(n) \right\};$$

end for

2: *Update.* Update the set of virtually received features' indices as $\mathcal{W}_{k+1} = \mathcal{W}_k \bigcup \mathcal{S}_k^\star$;

end for

Output: $\mathcal{S}_1^\star, \mathcal{S}_2^\star, ..., \mathcal{S}_{k^*}^\star$.

sub-optimal feature subsets $\{\mathcal{S}_k^\star\}$ for all current and future slots can be determined by the importance aware feature selection (i.e., Algorithm 1) at the beginning of the current slot. Given the total throughput in k^* proceeding transmissions, $Y_0 k^*$, the total number of features selected for transmission is $|\bigcup_{k=1}^{k^*} \mathcal{S}_k^\star| = \min\{N - |\mathcal{W}_1|, Y_0 k^*\}$ where $N - |\mathcal{W}_1|$ specifies the number of untransmitted features in the current slot. Let $\mathrm{G}^\star(\theta_1, k^*)$ denote the discriminant gain of importance-aware selected features subject to a number of transmissions k^* and conditioned on θ_1. Then the cumulative discriminant gain over $k^\star$ transmissions is obtained as $G(\mathcal{S}^\star) = \mathrm{G}^\star(\theta_1, k^*)$. Problem (P3) can be rewritten for a Gaussian channel as

$$\text{(P4)} \quad \begin{aligned} \min_{k^*} \quad & \tilde{H}\left(\delta_1, \mathrm{G}^\star(\theta_1, k^*)\right) + c_0 k^* \\ \text{s.t.} \quad & k^* \in \{0, 1, ..., K\}. \end{aligned}$$

A discrete function $d(k)$ is called convex if the inequality $d(k-1) + d(k+1) \geq 2d(k)$ holds for every k in its domain [27]. The Problem (P4) is convex. To prove this, we first rewrite the cumulative discriminant gain as

$$\begin{aligned} \mathrm{G}^\star(\theta_1, k^*) &= \max_{\substack{\mathcal{S} \subseteq (\mathcal{W} \setminus \mathcal{W}_k) \\ |\mathcal{S}| = \min\{N - |\mathcal{W}_1|, Y_0 k^*\}}} \sum_{n \in \mathcal{S}} g(n) \\ &= \sum_{k=1}^{k^*} G\left(\mathcal{S}_k^\star\right). \end{aligned} \tag{24}$$

Using the fact that $G(\mathcal{S}_k^\star) \geq G(\mathcal{S}_{k+1}^\star)$ for all k, one can infer from the last equation that

$$\mathrm{G}^\star(\theta_1, k^* - 1) + \mathrm{G}^\star(\theta_1, k^* + 1) \leq 2\mathrm{G}^\star(\theta_1, k^*), \tag{25}$$

and thereby conclude that $\mathrm{G}^\star(\theta_1, k^*)$ is *concave with respect to* (w.r.t.) k^*. Combining this fact and that $\tilde{H}(\delta_1, G)$ is a convex and monotone decreasing function of G shows that Problem (P4) is also convex. Based on a standard approach in discrete convex optimization [27], the optimal solution for Problem (P4) can be derived as

$$k^\star = \min\{K, \; \tilde{k}\}, \tag{26}$$

where $\tilde{k}$ is given by

$$\begin{aligned} \tilde{k} = \min\{k \in \{0, ..., K-1\} \mid \tilde{H}\left(\delta_1, \mathrm{G}^\star(\theta_1, k)\right) \\ - \tilde{H}\left(\delta_1, \mathrm{G}^\star(\theta_1, k+1)\right) \leq c_0\}. \end{aligned} \tag{27}$$

Combining (23), (26), and (27) leads to the following main result of the section.

Theorem 1 (*Stopping Control for Gaussian Channels*)**.** The stopping rule for Gaussian channels, which solves Problem (P4), is that the transmission in ProgressFTX should be terminated if $\tilde{H}\left(\delta_1, 0\right) - \tilde{H}\left(\delta_1, \mathrm{G}^\star(\theta_1, 1)\right) \leq c_0$. In other words, the stopping policy is given as

$$b_1^\star = \begin{cases} 0, & R \leq c_0, \\ 1, & \text{otherwise}, \end{cases} \tag{28}$$

where the incremental reward (i.e., reduction in classification uncertainty) from transmission in the current slot is denoted as $R = \tilde{H}\left(\delta_1, 0\right) - \tilde{H}\left(\delta_1, \mathrm{G}^\star(\theta_1, 1)\right)$.

The stopping policy in Theorem 1, which is optimal for Problem (P4) while sub-optimal for Problem (P1), is interpreted as follows. As k^* grows, the transmission cost $c_0 k^*$ accumulates at a constant speed c_0, while the incremental reward from transmission in slot k^* monotonically decreases due to the convexity of the function $\tilde{H}\left(\delta_1, \mathrm{G}^\star(\theta_1, k^*)\right)$ (see Lemma 3 and (25)). For this reason, $k^\star$ is exactly the maximum number of subsequent transmissions such that the incremental reward from transmitting in slot $(k^\star + 1)$ is no larger than the per-slot transmission cost, yielding the stopping threshold. In the special case of $k^\star = 0$, transmissions should be terminated in the current slot.

The server algorithm for controlling ProgressFTX, including both feature selection and stopping, is summarized in Algorithm 2, where the index k refers to the k-th slot in an edge inference task instead of that for slot k from a current slot in control problems. At each slot, the computation complexity of importance-aware feature selection in Algorithm 2 is $O(Y_0)$ as there are Y_0 features to be chosen, while the complexity of stopping control is simply $O(1)$.

B. Transmission Termination with Fading Channels

In this sub-section, the stopping policy for Gaussian channels is extended to the case of fading channels with outage, where no CSIT is available. This technique can be extended to the case with instantaneous CSIT by averaging the expected uncertainty bound $\tilde{H}$ over random channel states. In the case of no CSIT, it requires that two issues be addressed. The first is the retransmission strategy upon the occurrence of an outage event. Given importance-aware feature selection, the features that fail to be received due to channel outage are most important ones among the undelivered features, and outweigh the expected communication cost. Following few straightforward steps omitted due to limited space, this leads to the following strategy.

Algorithm 2: ProgressFTX Control at the Server for Gaussian Channels

Initialize: Observe the system state of the first slot θ_1;
repeat:
1: *Feature selection.* Determine sub-optimal feature subset $\mathcal{S}_k^\star$ by Algorithm 1;
2: *Stopping control.* Evaluate the stopping control indicator $b_k^\star$ by (28);
3: **if** $b_k^\star = 1$ **then:**
1) *Feature transmission.* Receive the incremental feature vector $\Delta\mathbf{x}_k$;
2) *Partial feature vector update.* Update the partial feature vector by the rule $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{M}\Delta\mathbf{x}_k$, with the placement matrix $\mathbf{M}$ given by (10);
3) *System state evolution.* Evolve to state θ_{k+1};
end if
until Scheduled stopping $b_k^\star = 0$;
Server inference. Infer the label $\hat{\ell}$ using (3);
Termination. Feedback the inferred label $\hat{\ell}$ to device.

Lemma 4 (*Optimality of Feature Retransmission*)**.** To minimize the upper bound of classification uncertainty, it is optimal to retransmit an incremental feature vector, whose transmission fails due to channel outage, in the next slot.

The second issue is whether the optimal stopping problem in the current case retains the convexity of its Gaussian-channel counterpart. The answer is positive as shown in the sequel. Given the fixed communication rate R_0 and channel outage probability p_o, the transmitted features can be successfully received with probability $1 - p_\text{o}$. The number of successful transmissions out of k^* trials, denoted by k', follows a Binomial distribution, $\text{Binom}\,(k^*, 1-p_\text{o})$. The conditional *probability mass function* (PMF) of k' is given by

$$p(k'|k^*) = \binom{k^*}{k'}(1-p_\text{o})^{k'} p_\text{o}^{k^*-k'},\ k' = 0, 1, \cdots, k^*. \quad (29)$$

Based on the retransmission strategy in Lemma 4, the total discriminant gain of features in k' successfully received incremental feature vectors is given by $\text{G}^\star(\theta_1, k')$ defined in (24). Combining $\text{G}^\star(\theta_1, k')$ and (29), the expectation of the upper bound of uncertainty after k^* times of transmission is denoted as $\Phi(\theta_1, k^*)$ and given by

$$\begin{aligned}\Phi(\theta_1, k^*) &\triangleq \mathbb{E}_{k'}\left[\tilde{H}\left(\delta_1, \text{G}^\star(\theta_1, k')\right)\big|k^*\right] \\ &= \sum_{k'=0}^{k^*} \tilde{H}\left(\delta_1, \text{G}^\star(\theta_1, k')\right) p(k'|k^*). \quad (30)\end{aligned}$$

Then Problem (P3) can be rewritten for a fading channel with outage as

$$\text{(P5)} \quad \begin{aligned}\min_{k^*}\ & \Phi(\theta_1, k^*) + c_0 k^* \\ \text{s.t.}\ & k^* \in \{0, 1, ..., K\}.\end{aligned}$$

Lemma 5. The problem of optimal stopping for a fading channel with outage, namely Problem (P5), is convex.

Proof. See Appendix B. □

Given the convexity and following the arguments used to solve Problem (P4), the optimal solution for Problem (P5) is derived to be

$$k^\star = \min\{K, \tilde{k}\}, \quad (31)$$

where $\tilde{k}$ in this case is given by

$$\tilde{k} = \min\{k \in \{0, ..., K-1\} \mid \Phi\left(\theta_1, 0\right) - \Phi\left(\theta_1, k\right) \le c_0\}. \quad (32)$$

The stopping rule follows from (23), (31), and (32), leading to the following:

Theorem 2 (*Transmission Termination for Fading Channels*)**.** The stopping rule for a fading channel with outage, which solves Problem (P5), is that the transmission in ProgressFTX should be terminated if $\Phi(\theta_1, 0) - \Phi(\theta_1, 1) \le c_0$. The corresponding stopping policy is given as

$$b_1^\star = \begin{cases} 0, & R \le \frac{c_0}{1-p_\text{o}}, \\ 1, & \text{otherwise}, \end{cases} \quad (33)$$

where the incremental reward $R = \tilde{H}\left(\delta_1, 0\right) - \tilde{H}\left(\delta_1, \text{G}^\star(\theta_1, 1)\right)$ follows that in Theorem 1.

Comparing Theorems 1 and 2, the stopping policy in the case of fading channel retains the threshold-based structure of its Gaussian-channel counterpart. Nevertheless, the transmission threshold is higher in the former than that in the latter (by a factor of $\frac{1}{1-p_\text{o}}$) due to the additional communication cost caused by channel outage. To be specific, given the same level of uncertainty reduction, the needed number of transmission, say k, in the case of Gaussian channel is expected to increase to $\frac{k}{1-p_\text{o}}$ when fading is present. As a result, a requirement on inference accuracy achievable in the Gaussian channel case may not be affordable in the fading channel case.

ProgressFTX control in Algorithm 2 can be easily modified to account for outage in fading channels as follows. First, the stopping control indicator $b_k^\star$ shall be evaluated by (33) instead. Next, Step 3 should be revised as follows. If the transmission in sub-step 1) is successful, then sub-step 2) is partial feature vector update as presented in Algorithm 2; otherwise, sub-step 2) involves the server sending a retransmission signal to the device for retransmission. The computation complexity of the modified algorithm remains the same as Algorithm 2.

VI. ProgressFTX for CNN Classifiers

Given the complex architecture of a CNN model, ProgressFTX for a CNN classifier cannot be directly designed using an optimization approach as in the preceding sections for its linear counterpart. To overcome the difficulty, we leverage the principles underpinning the optimized ProgressFTX policies, namely importance-aware feature selection and optimal stopping, for the linear model to design their counterparts under the CNN model as follows.

A. Importance-aware Feature Selection

Consider the step of feature selection in the ProgressFTX protocol in Section III-A. The importance of the n-th feature map is defined to be the summed importance of

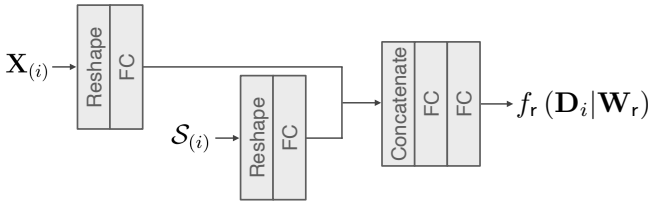


Fig. 5. Illustration of the architecture for regression models.

parameters in the n-th filter outputting the map: $g_{\mathsf{c}}(n) = \sum_{w_m \text{ in the } n\text{-th filter}} \tilde{I}(m)$. Since $\left\{\frac{\partial \mathcal{L}}{\partial w_m}\right\}$ are readily available in training, the server is able to obtain the value of $\{g_{\mathsf{c}}(n)\}$ after training $f_{\mathsf{d}}(\cdot)$ and form a lookup table for reference during ProgressFTX control. Given $\{g_{\mathsf{c}}(n)\}$, the importance-aware feature selection for each slot is performed by selecting a given number of the 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. Algorithm 1 for the linear classifier can be modified accordingly with no change in the computation complexity. The details are straightforward, and thus omitted.

B. Stopping Control Based on Uncertainty Prediction

Consider the step of stopping control in the ProgressFTX protocol in Section III-A. The optimal stopping control in the current case is stymied by the difficulty in finding a tractable and accurate approximation of the expected classification uncertainty as a function of input features similarly to Lemma 3 for linear classification. 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 by the server. One particular popular architecture of regression models in the literature is adopted [28], [29]. To be able to predict the uncertainty for a future partial feature-map tensor, both the current partial feature-map tensor and the selected index subset are needed as the input. Since the underpinning prediction is based on the current partial feature maps, ProgressFTX control for CNN models is also 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 concatenated feature tensor along the selected concatenation axis. This is implemented by a concatenation layer which then feeds the concatenated tensor into the deeper layers of the regression model (see Fig. 5).

To train the regression model, a training dataset and a prediction loss function need to be properly designed. 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 and 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 represent 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:

$$\min_{\mathbf{W}_{\mathsf{r}}} \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \left[f_{\mathsf{r}}(\mathbf{D}_i|\mathbf{W}_{\mathsf{r}}) - H_i\right]^2. \tag{34}$$

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 written as

$$\min_{k^* \in \{0,1,...,K\}} f_{\mathsf{r}}\left(\left(\mathbf{X}_1, \bigcup_{k=1}^{k^*} \mathcal{S}_k^\star\right) \middle| \mathbf{W}_{\mathsf{r}}\right) + c_0 k^*. \tag{35}$$

The sub-optimal stopping time from the current slot, namely

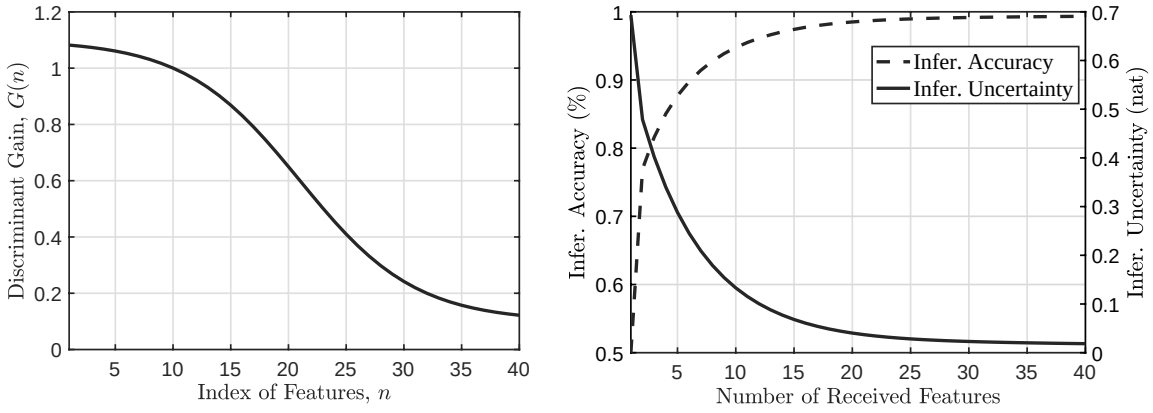
$$k^\star = \underset{k^* \in \{0,1,..,K\}}{\arg\min} f_{\mathsf{r}}\left(\left(\mathbf{X}_1, \bigcup_{k=1}^{k^*} \mathcal{S}_k^\star\right) \middle| \mathbf{W}_{\mathsf{r}}\right) + c_0 k^*, \tag{36}$$

can be determined by linear search. Then the stopping decision in the current slot is $b_1^\star = \min\{1, k^\star\}$. At each slot in online scheduling, the computation complexity of importance-aware feature selection is $O(Y_0)$. Let N_{in} denote the total dimension of two inputs to the uncertainty predictor f_{r}. The computation complexity of stopping control (36) is $O(N_{\mathsf{in}}K)$. As a remark, 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 the classifier to reach the same target confidence level if it is possible.

VII. Experimental Results

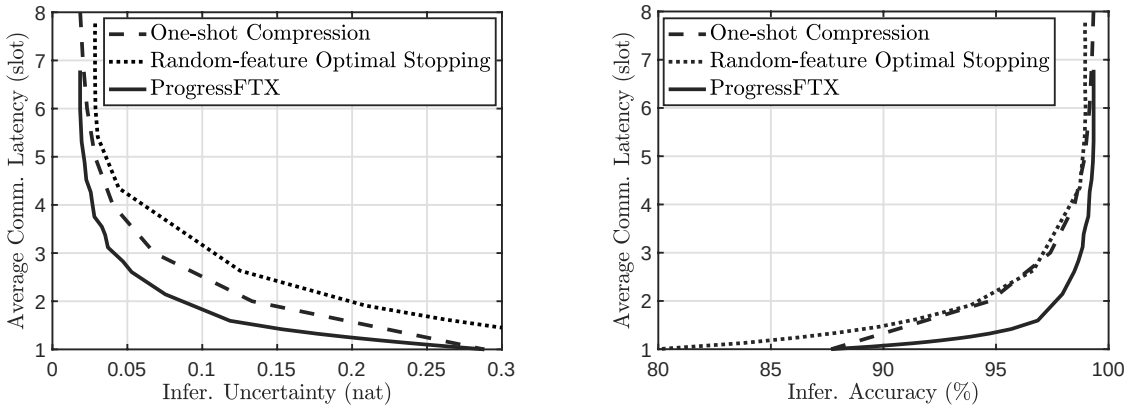
A. Experimental Settings

The experimental setups are designed as follows, unless specified otherwise. Each feature is quantized at a high resolution, $Q = 64$ bits/feature, for digital transmission. The horizon in online scheduling is set as $K = 5$ slots. A statistical (GM) dataset is used for training and testing the linear model and the popular MNIST dataset of handwritten digits for the CNN model. The settings in the two cases are described as follows. Consider the case of linear classification. For the GM dataset, there are $L = 2$ classes with $N = 40$ features in total for each sample. Due to the small number of features, only a narrow-band channel is needed whose bandwidth is set as $B = 20$ kHz. The slot duration is $T = 10$ milliseconds. For the case of Gaussian channel, the channel SNR is set as 4 dB such that the transmission rate is $Y_0 = 5$ features/slot. For the case of fading channel, the transmit rate is fixed at $Y_0 = 5$ features/slot while the outage probability is $p_{\mathsf{o}} = 0.1$. For the case of CNN, we use the well-known LeNet [15]. The Gaussian channel in this case has a bandwidth of $B = 2.6$ MHz, the slot duration



(a) Discriminant Gains (b) Inference Accuracy and Uncertainty

Fig. 6. (a) The discriminant gains of features and (b) their effects on inference performance (i.e., inference accuracy and uncertainty) in the case of linear classification.



(a) Inference Uncertainty (b) Inference Accuracy

Fig. 7. Comparison of average communication latency between ProgressFTX and benchmarking schemes in the case of linear classification and Gaussian channel for (a) inference uncertainty or (b) inference accuracy.

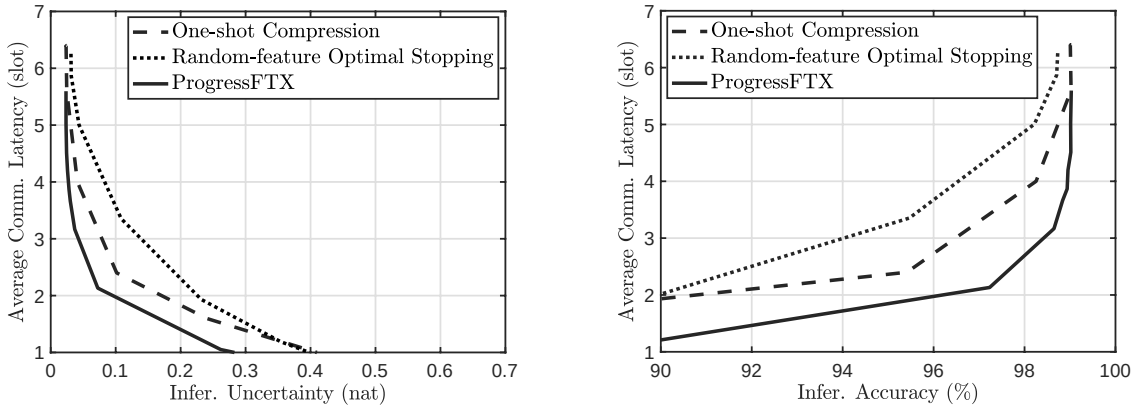
of $T = 10$ milliseconds, and the channel SNR $= 4$ dB. The corresponding transmission rate is $Y_0 = 4$ feature-maps/slot.

The optimized control of ProgressFTX in the case of CNN classification requires an uncertainty predictor as designed in Section VI. Its architecture comprises two input layers, as illustrated in Fig. 5. 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 fully-connected layers with 100, 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 benchmarking schemes are considered. The first scheme, 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 $k^\star$ is solved from $\text{argmin}_{k^*} \tilde{H}(0, \text{G}^\star(\theta_1, k^*)) \geq H_0$ for linear classifiers or $\text{argmin}_{k^*} f_{\mathsf{r}}\left(\left(\mathbf{0}, \bigcup_{k=1}^{k^*} \mathcal{S}_k^\star\right) | \mathbf{W}_{\mathsf{r}}\right) \geq H_0$ for CNN classifiers, respectively. The drawback of this scheme is that its lack of ACK/NACK feedback as for ProgressFTX makes the device inept in minimizing the number of features to achieve an exact uncertainty level. In the current experiments, the device has to instead target the expected uncertainty approximated in (21) for the case of linear classification or predict the uncertainty in the CNN case. The second scheme, termed *random-feature optimal stopping*, modifies the optimized ProgressFTX by removing feature importance awareness and instead selecting features randomly, which hence belongs to the class of optimal stopping techniques with sub-optimal performance.

B. Linear Classification Case

First, we evaluate the effect of importance-aware feature selection on inference performance. To this end, discriminant gains of different feature dimensions, which are arranged in a decreasing order, are plotted in Fig. 6(a). In Fig. 6(b), the levels of inference accuracy and uncertainty are plotted against the number of received features arranged in the same order. One can observe the monotonic reduction of accuracy and growth of uncertainty as the number of features increases. The result confirms the usefulness of importance awareness in



(a) Inference Uncertainty (b) Inference Accuracy

Fig. 8. Comparison of average communication latency between ProgressFTX and benchmarking schemes in the case of linear classification and fading channel with varying target (a) inference uncertainty or (b) inference accuracy.

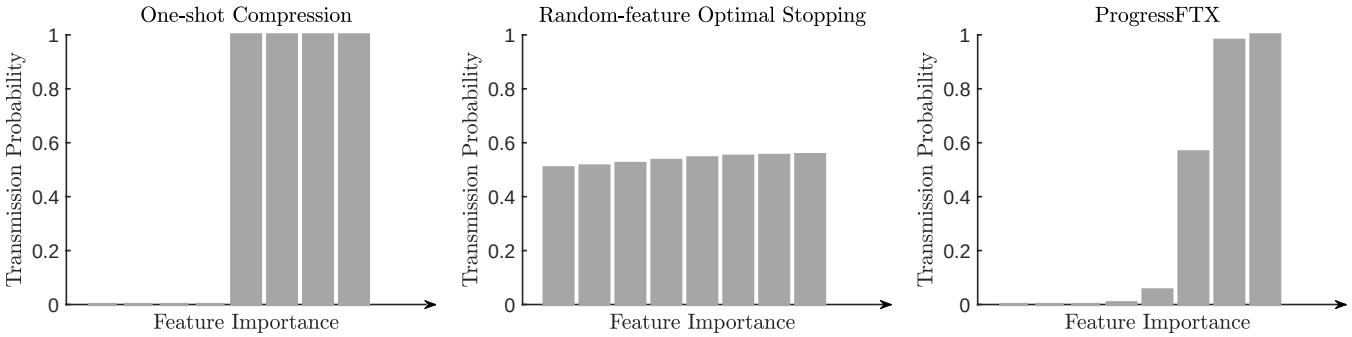


Fig. 9. The transmission probabilities of feature dimensions for the case of linear classification and Gaussian channel with target inference accuracy of 98.5%.

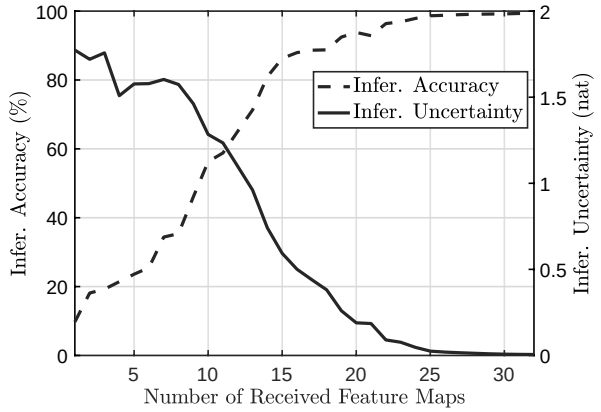
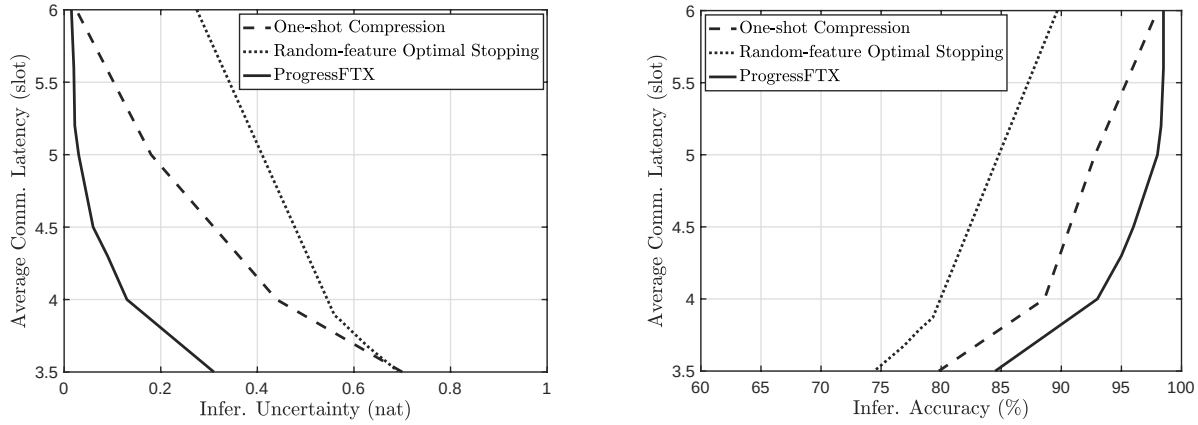


Fig. 10. Effect of the number of received feature maps on CNN classification accuracy with importance-aware feature selection.

feature selection for shortening the communication duration given target inference accuracy (or expected uncertainty).

Define the *average communication latency* of a transmission scheme for edge inference as the average number of transmission slots required for meeting a requirement on inference accuracy or expected uncertainty. Considering a Gaussian channel, the average communication latency of ProgressFTX and two benchmarking schemes are compared for varying target accuracy and expected uncertainty in Fig. 7. As observed from the figures, the proposed ProgressFTX technique achieves the lowest latency based on both the criteria of achieving targeted uncertainty and accuracy. For instance, two benchmarking schemes require in average 2.2 slots to achieve an accuracy of 95% while ProgressFTX only requires 1.4. In terms of uncertainty, in average 2.8 slots are used to achieve uncertainty of 0.05 for ProgressFTX. In comparison, the average latency for one-shot compression and random-feature optimal stopping are about 39% and 50% higher, respectively. Moreover, ProgressFTX achieves the lowest average inference uncertainty of 0.018 and the highest accuracy of 99%. The average communication latency of the three schemes are also compared in the case of a fading channel in Fig. 8. In the presence of fading, ProgressFTX continues to outperform benchmarking schemes. The above comparisons demonstrate the gains of importance awareness in feature selection and stochastic control of transmission.

Define the *transmission probability* of a feature dimension as the fraction of samples whose inference requires the transmission of the feature in this dimension. To further compare ProgressFTX with benchmarking schemes, the curves of transmission probability of each feature dimension versus its importance level (i.e., discriminant gain) are plotted in Fig. 9 for the case of linear classification and Gaussian channel. One can observe that the transmission probability is almost *uniform* over unpruned feature dimensions for the scheme of one-shot compression and all dimensions for the scheme of random-feature optimal stopping. This indicates their lack of



(a) Inference Uncertainty (b) Inference Accuracy

Fig. 11. Comparison of average communication latency between ProgressFTX and benchmarking schemes for the case of CNN classification and Gaussian channel with varying target (a) inference uncertainty or (b) inference accuracy.

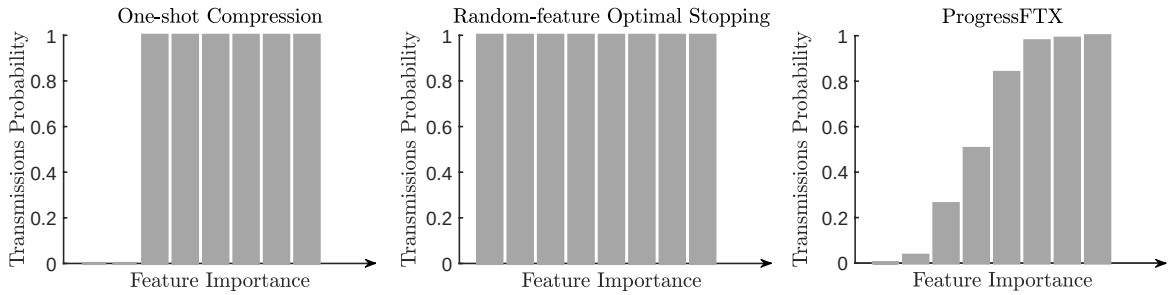


Fig. 12. The transmission probabilities of feature maps for the case of CNN classification and Gaussian channel with target inference accuracy of 98.5%.

feature importance awareness. In contrast, the probabilities for ProgressFTX are highly skew with higher probabilities for more important feature dimensions 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.

C. CNN Classification Case

We consider the case of CNN classification where ProgressFTX is controlled using the algorithms developed in Section VI. 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. To rein in the complexity of model architecture and training, we consider a Gaussian channel as commonly assumed in the literature (see e.g., [5], [9]). Given these settings, one sample per inference task and $Y_0 = 4$ feature maps transmitted in one slot, at most 8 slots are required to transmit all maps.

First, the curves of inference performance (i.e., inference accuracy and uncertainty) versus the number of transmitted feature maps, selected with importance awareness, are plotted in Fig. 10. The trade-offs are similar to those in the linear-classification case in Fig. 6(b). Particularly, due to importance-aware feature selection, the accuracy is observed to rapidly grow and the uncertainty quickly reduces as the number of received feature maps increases.

The comparisons in Fig. 7 is repeated in Fig. 11 for the CNN classifier. The same observation can be made that ProgressFTX outperforms the benchmark schemes over the considered ranges of inference uncertainty and accuracy. For instance, given 4 transmission slots in average, ProgressFTX achieves the uncertainty of 0.13 and accuracy of 93% which are at least 65% lower and 6.9% higher than the benchmarking schemes. As another example, for the target accuracy of 93%, one can observe from Fig. 11(b) that ProgressFTX requires in average 4 slots while one-shot compression requires 5, corresponding to 20% latency reduction for the former. Last, the transmission probabilities of different feature maps are plotted against their importance levels in Fig. 12. The observations are similar to those for the linear classifier in Fig. 9.

VIII. Concluding Remarks

In this paper, a novel feature transmission technique called ProgressFTX has been proposed for communication-efficient edge inference. ProgressFTX progressively selects and transmits important features until the desired inference performance is reached, or terminates the transmission judiciously to reduce the transmission cost. Comprehensive experimental results demonstrate its benefits in reducing the communication latency under given inference performance targets. This first study of progressive transmission for split inference opens a new

direction to overcome communication bottlenecks in edge inference. In particular, the ProgressFTX protocol can be extended to multiuser systems by applying a traditional multi-access scheme and designing a suitable resource allocation algorithm. The principle of resource allocation is to be aware of the heterogeneous channel states and accuracy requirements among devices, as well as to match devices' importance levels to CSIT, so as to enhance the overall system throughput in terms of the number of completed tasks (or the number of served users) within a given time window. Other interesting topics for further study include adaptive power control and user scheduling for multiuser ProgressFTX.

APPENDIX

A. Proof of Lemma 3

Substituting (16) and (19) into (20), one can observe that the targeted integral consists of a group of integrals of Gaussian and exponential functions. This kind of integral is well-known in the literature that not to tend itself for a closed-form expression but can be expressed in terms of multiple *complementary error functions* [30], [31]. Due to limited space, we do not present the expression of $\bar{H}^{\mathsf{ub}}(\delta_1, G(\mathcal{S}))$ but directly report a fact derived from the expression, $\lim_{G(\mathcal{S})\to\infty} \frac{\bar{H}^{\mathsf{ub}}(\delta_1,G(\mathcal{S}))}{\left(e^{\frac{\delta_1}{2}}+e^{-\frac{\delta_1}{2}}\right)(G(\mathcal{S}))^{-\frac{1}{2}}e^{-\frac{G(\mathcal{S})}{8}}} = \frac{16}{9}\sqrt{\frac{2}{\pi}}$. This results in $\bar{H}^{\mathsf{ub}}(\delta_1, G(\mathcal{S})) = O\left(\left(e^{\frac{\delta_1}{2}}+e^{-\frac{\delta_1}{2}}\right)(G(\mathcal{S}))^{-\frac{1}{2}}e^{-\frac{G(\mathcal{S})}{8}}\right)$, where $O(\cdot)$ is the Bachmann–Landau notation establishing an equivalence between shrinking rates of two functions. Consequently, given any δ_1, $\bar{H}^{\mathsf{ub}}(\delta_1, G(\mathcal{S}))$ *monotonically* decays to zero at a *super-exponential* rate in the asymptotic regime of $G(\mathcal{S})$. This scaling law proves Lemma 3.

B. Proof of Convexity of Problem (P5)

To show the convexity of Problem (P5), it is equivalent to prove the following inequality holds for $k^* \geq 1$, that is $\Phi(\theta_1, k^*-1)+\Phi(\theta_1, k^*+1)-2\Phi(\theta_1, k^*) \triangleq \nabla^2\Phi(\theta_1, k^*) \geq 0$, where $\nabla^2\Phi(\theta_1, k^*)$ is the second-order difference of $\Phi(\theta_1, k^*)$ w.r.t. its second argument. The outline of this proof is to first construct a function holding constantly zero second-order difference, which then is shown to be the lower bound of the counterpart of the target function. First, we construct a function $\tilde{\Phi}(k^*) \triangleq \sum_{k'=0}^{k^*}(l_1k'+l_2)\,p(k'|k^*)$, where $l_1k'+l_2$ is an *arbitrary* linear function with constants l_1 and l_2. The closed-form expression of $\tilde{\Phi}(k^*)$ can be easily evaluated to be $\tilde{\Phi}(k^*) = l_1k^*(1-p_{\mathsf{o}})+l_2$, given the binomial distribution of k' conditioned on k^*. Consequently, the second-order difference of $\tilde{\Phi}(k^*)$ always equals to zero, i.e.,

$$\nabla^2\tilde{\Phi}(k^*) = 0, \ \forall k^*. \tag{37}$$

The analytical form of the second-order difference of PMF w.r.t. k^*, which is denoted as $\nabla^2 p(k'|k^*) = p(k'|k^*-1) + p(k'|k^*-1) - 2p(k'|k^*)$, is given by

$$\nabla^2 p(k'|k^*) = p_{\mathsf{o}}^{k^*-k'}(1-p_{\mathsf{o}})^{k'}\frac{\prod_{i=k^*-k'+2}^{k^*-1} i}{\prod_{i=1}^{k'} i}\eta(k'), \tag{38}$$

where

$$\eta(k') \triangleq \big[\, p_{\mathsf{o}}^{-1}(k^*-k'+2)(k^*-k') + p_{\mathsf{o}}k^*(k^*+1) - 2k^*(k^*-k'+1)\big].$$

Observes from (38) that the sign of $\nabla^2 p(k'|k^*)$ is determined by $\eta(k')$, a convex quadrature function of k'. Specifically, $\eta(0) \geq 0$ and $\eta(k^*) \geq 0$ hold. We conclude that given k^*, $\nabla^2 p(k'|k^*)$ is non-positive only in one closed integral of *consecutive* integers k', where the left and right end-points are denoted by $k_1 > 0$ and $k_2 < k^*$, respectively.

We expand $\nabla^2\Phi(\theta_1, k^*)$ and $\nabla^2\tilde{\Phi}(k^*)$ as $\nabla^2\Phi(\theta_1, k^*) = \sum_{k'=0}^{k^*}\tilde{H}\left(\delta_1, \mathrm{G}^\star(\theta_1, k')\right)\nabla^2 p(k'|k^*)$ and $\nabla^2\tilde{\Phi}(k^*) = \sum_{k'=0}^{k^*}(l_1k'+l_2)\,\nabla^2 p(k'|k^*)$, respectively. Choose $l_1 \leq 0$ and $l_2 \geq 0$ such that $l_1k'+l_2 > 0$ if $k' \geq k_1$ and $l_1k'+l_2 \geq 0$ if $k' > k_1$. Due to $\tilde{H}\left(\delta_1, \mathrm{G}^\star(\theta_1, k')\right) \geq 0$, one has

$$\sum_{k'=k_2+1}^{k^*}\tilde{H}\left(\delta_1, \mathrm{G}^\star(\theta_1, k')\right)\nabla^2 p(k'|k^*) \geq 0 \geq \sum_{k'=k_2+1}^{k^*}\tilde{H}\left(\delta_1, \mathrm{G}^\star(\theta_1, k')\right)\nabla^2 p(k'|k^*). \tag{39}$$

Next, we note that $\tilde{H}\left(\delta_1, \mathrm{G}^\star(\theta_1, k')\right)$ is strictly convex and monotonically decreasing w.r.t. k' while $l_1k'+l_2$ is linear. It is always able to find an $l_1 \leq 0$ with sufficiently small $|l_1|$ such that

$$\sum_{k'=0}^{k_2}\tilde{H}\left(\delta_1, \mathrm{G}^\star(\theta_1, k')\right)\nabla^2 p(k'|k^*) \geq \sum_{k'=0}^{k_2}\tilde{H}\left(\delta_1, \mathrm{G}^\star(\theta_1, k')\right)\nabla^2 p(k'|k^*) \geq 0. \tag{40}$$

Combining (37), (39) and (40) leads to the desired results, which completes the proof.

$$\nabla^2\Phi(\theta_1, k^*) \geq \nabla^2\tilde{\Phi}(k^*) = 0. \tag{41}$$

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

[4] X. Huang and S. Zhou, "Dynamic compression ratio selection for edge inference systems with hard deadlines," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8800–8810, 2020.

[5] W. Shi, Y. Hou, S. Zhou, Z. Niu, Y. Zhang, and L. Geng, "Improving device-edge cooperative inference of deep learning via 2-step pruning," in *Proc. IEEE INFOCOM Workshops*, Paris, France, Apr. 29 - May 2, 2019.

[6] 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.

[7] 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.

[8] J. Shao and J. Zhang, "Bottlenet++: An end-to-end approach for feature compression in device-edge co-inference systems," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC WKSHPS)*, Dublin, Ireland, Jun. 7-11, 2020.

[9] M. Jankowski, D. Gündüz, and K. Mikolajczyk, "Joint device-edge inference over wireless links with pruning," in *Proc. IEEE Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, Atlanta, GA, USA, May 26-29, 2020.

[10] 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.

[11] X. Xu, Y. Ding, S. X. Hu, M. Niemier, J. Cong, Y. Hu, and Y. Shi, "Scaling for edge inference of deep neural networks," *Nature Electron.*, vol. 1, no. 4, pp. 216–222, 2018.

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vision Pattern Recog. (CVPR)*, Boston, MA, USA, Jun., 2015.

[13] 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.

[14] Z. Zhuang, M. Tan, B. Zhuang, J. Liu, Y. Guo, Q. Wu, J. Huang, and J. Zhu, "Discrimination-aware channel pruning for deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Montreal, Canada, Dec. 3-8, 2018.

[15] 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.

[16] G. Saon and M. Padmanabhan, "Minimum bayes error feature selection for continuous speech recognition," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Denver, CO, USA, Jan. 1, 2000.

[17] 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.

[18] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data mining, Inference, and Prediction*. New York, NY, USA: Springer Science & Business Media, 2009.

[19] R. A. Gopinath, "Maximum likelihood modeling with gaussian distributions for classification," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Seattle, WA, USA, May 1998.

[20] 3GPP, "Study on traffic characteristics and performance requirements for ai/ml model transfer in 5gs,," 3GPP, Tech. Rep. TR 22.874, 2021.

[21] S. Weber, X. Yang, J. Andrews, and G. de Veciana, "Transmission capacity of wireless ad hoc networks with outage constraints," *IEEE Trans. Inf. Theory*, vol. 51, no. 12, pp. 4091–4102, 2005.

[22] 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.

[23] D. Liu, G. Zhu, J. Zhang, and K. Huang, "Data-importance aware user scheduling for communication-efficient edge machine learning," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 1, pp. 265–278, 2021.

[24] M. R. Sikonja and I. Kononenko, "Theoretical and empirical analysis of Relief and ReliefF," *Mach. Learn.*, vol. 53, p. 23–69, 2003.

[25] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York, NY, USA: Wiley, 2006.

[26] Y. Chow, H. Robbins, and D. Siegmund, *Great expectations: The theory of optimal stopping*. Boston, MA, USA: Houghton Mifflin, 1971.

[27] K. Murota, "Discrete convex analysis," *Math. Program.*, vol. 83, no. 1, pp. 313–371, 1998.

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

[29] J. Gao, Q. Wang, and Y. Yuan, "SCAR: Spatial-/channel-wise attention regression networks for crowd counting," *Neurocomputing*, vol. 363, pp. 1–8, 2019.

[30] O. Olabiyi and A. Annamalai, "Invertible exponential-type approximations for the Gaussian probability integral Q(x) with applications," *IEEE Wireless Commun. Lett.*, vol. 1, no. 5, pp. 544–547, 2012.

[31] M. Chiani, D. Dardari, and M. Simon, "New exponential bounds and approximations for the computation of error probability in fading channels," *IEEE Trans. Wireless Commun.*, vol. 2, no. 4, pp. 840–845, 2003.