Automating Deep Neural Network Model Selection for Edge Inference

Bingqian Lu
UC Riverside

Jianyi Yang
UC Riverside

Lydia Y. Chen
TU Delft

Shaolei Ren
UC Riverside

***Abstract*—The ever increasing size of deep neural network (DNN) models once implied that they were only limited to cloud data centers for runtime inference. Nonetheless, the recent plethora of DNN model compression techniques have successfully overcome this limit, turning into a reality that DNN-based inference can be run on numerous resource-constrained edge devices including mobile phones, drones, robots, medical devices, wearables, Internet of Things devices, among many others. Naturally, edge devices are highly heterogeneous in terms of hardware specification and usage scenarios. On the other hand, compressed DNN models are so diverse that they exhibit different tradeoffs in a multi-dimension space, and not a single model can achieve optimality in terms of all important metrics such as accuracy, latency and energy consumption. Consequently, how to automatically select a compressed DNN model for an edge device to run inference with optimal quality of experience (QoE) arises as a new challenge. The state-of-the-art approaches either choose a common model for all/most devices, which is optimal for a small fraction of edge devices at best, or apply device-specific DNN model compression, which is not scalable. In this paper, by leveraging the predictive power of machine learning and keeping end users in the loop, we envision an automated device-level DNN model selection engine for QoE-optimal edge inference. To concretize our vision, we formulate the DNN model selection problem into a contextual multi-armed bandit framework, where features of edge devices and DNN models are contexts and pre-trained DNN models are arms selected online based on the history of actions and users' QoE feedback. We develop an efficient online learning algorithm to balance exploration and exploitation. Our preliminary simulation results validate our algorithm and highlight the potential of machine learning for automating DNN model selection to achieve QoE-optimal edge inference.**

I. Introduction

Machine learning models, especially *deep neural networks* (DNNs), have recently enabled unprecedented levels of intelligence on numerous systems and found successful applications in a broad spectrum of domains, including computing vision, healthcare, autonomous driving, machine translation, among many others [1], [2]. To achieve accuracies comparable to or even exceeding human levels, today's machine learning models are becoming increasingly more complex, as evidenced by DNNs that can contain millions of or even more parameters [1], [3], [4]. Consequently, model training is typically run in cloud-scale data centers with vast computational resources. For a concrete example, at Facebook, DNN training for all applications, such as news ranking, content understanding, object tracking for virtual reality is exclusively run in its geographically distributed data centers [2].

While data centers remain as the preferred platforms for training most machine learning models, running machine learning inference (i.e., applying pre-trained models to new data) as close to end users as possible has emerged as a growing trend. In fact, there is a constant push to bring inference all the way down to end devices at the Internet edge, such as mobile phones, wearables, drones, medical devices and robots, which we collectively refer to as *edge* devices. For example, Instagram uses deep learning for real-time image processing upon a image capture on mobile devices, and a key goal of Facebook is to provide the best inference experience on its mobile app for more than 2 billion active users [2]. By running inference on edge devices (a.k.a. *edge inference*), users receive improved experiences with reduced latency and overall inference time, are less dependent on network connections, and can keep private data locally without transferring it to the cloud or other edge computing platforms [2], [5]–[8].

Along with advances in embedded and mobile hardware, the recent breakthroughs on DNN model compression (e.g., network pruning and weight quantization) have significantly reduced model sizes by orders of magnitude with an acceptable accuracy loss, successfully turning edge inference into a reality [4], [6]. Naturally, to run inference on resource-constrained edge devices with a satisfactory user experience (which we refer to as quality of experience, or QoE), inference accuracy is not the sole metric to optimize [2], [5]; instead, the employed DNN model architecture must be, in an *automated* manner, tailored to specific edge device hardware and strike an optimal balance among various important metrics such as accuracy, latency and energy consumption. For example, when a device has a small battery capacity, reducing energy consumption may be more important and result in a better QoE for the user than improving inference latency performance. Nonetheless, as detailed in Section II-A, optimizing QoE for edge inference is challenged by the extremely high degree of heterogeneity in terms of the underlying device hardware, usage scenarios and DNN models. First, there are thousands of unique systems on a chip (SoCs) running on more than ten thousand different types of smart phones and tablets, and the top 30 SoCs cover only 51% of the whole market in total. Second, DNN models running on different edge devices can be

The first two authors, Bingqian Lu and Jianyi Yang, make equal contributions to this work and are listed in an alphabetical order by last name. This work is supported in part by the U.S. NSF under grants CNS-1551661, ECCS-1610471, and CNS-1910208.

978-1-7281-6737-4/19/$31.00 ©2019 IEEE
DOI 10.1109/CogMI48466.2019.00035

exposed to significantly different usage scenarios, such as very different locations and illumination conditions, thus resulting in drastically different distributions of users' input data sets and hence also different inference accuracies even on the same DNN model [3], [8]. Last but not least, even with the same training data set but given different compression techniques or optimization parameters, hundreds of or even more different DNN models can be generated using neural architecture search techniques [9], [10], among which no single model is a clear winner in terms of all important metrics such as accuracy, latency and energy.

Consequently, in view of the extreme heterogeneity of edge devices, usage scenarios and DNN models, a new challenge arises: *how to automatically select DNN models for edge inference with optimal user experience?* To address this challenge, research efforts on optimal DNN model selection for edge inference have been quickly proliferating [3], [6]–[8], [11]–[14]. A straightforward approach is to benchmark on a set of edge devices and select a DNN model that is reasonably accurate and meets constraints for *most* devices. Nonetheless, this "one for most" approach can at best cover only a small set of devices under limited usage scenarios, thus lacking wide applicability as edge devices and usage scenarios are extremely diverse. On the other hand, state-of-the-art AutoML and neural architecture search techniques have been employed to identify the optimal DNN model for a *specific* target edge device [4], [5], [9], [10], [13], [15]–[17]. This "one for one" approach can result in an optimal or near optimal DNN model for a specific device under given a set of optimization parameters, but it suffers from poor scalability: the same procedure needs to be executed, whenever an unseen new device comes. Furthermore, the way that DNN models are currently selected for edge inference is typically to apply optimization based on pre-determined parameters and hence "open loop", without incorporating users' QoE feedback into a closed loop [4], [5], [13], [16]. As a result, the optimized metrics may not necessarily translate into improved QoE of end users.

In this paper, we envision an automated DNN model selection engine for QoE-optimal edge inference. The key ideas we exploit are two-fold. *First*, we leverage the predictive power of online learning based on history data. Specifically, the DNN model selection engine needs to serve a huge number of users and also continuously update its decisions at runtime. Thus, as time goes on, more knowledge regarding how good DNN model selection decisions are for given edge devices is accumulated, which can be exploited to continiously improve future DNN model selections. *Second*, it is crucial to keep users in a closed loop when designing a DNN model selection engine. Naturally, optimized objective metrics (e.g., accuracy, latency, and energy) may not necessarily translate into improved QoE; instead, it is ultimately users themselves who decide the actual QoE. Thus, the design of a DNN model selection engine must incorporate feedback from users regarding their QoE, forming a closed loop with users in the middle.

As a preliminary investigation, we demonstrate our vision by proposing an automated DNN model selection engine powered by online learning (illustrated in Fig. 2), which leverages contextual multi-armed bandit (MAB) learning to automatically select DNN models for QoE-optimal edge inference. Specifically, given features of available DNN models and device features of each incoming edge device, our engine employs an online QoE predictor that estimates the resulting QoE for DNN model selection decisions. Then, based on predicted QoE, our engine outputs a selected DNN model for each edge device with the goal of optimizing the user's QoE while ensuring adequate exploration to avoid being trapped in a local optimum. After DNN models are installed and used for some time, users' QoE feedback is requested for updating the online QoE predictor and improving future DNN model selections, thus forming a closed loop.

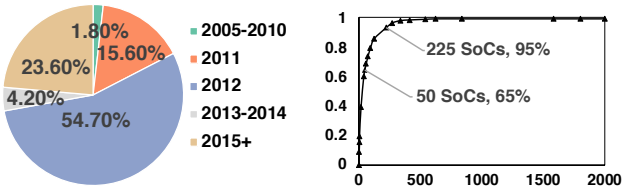
More concretely, our online QoE predictor assumes for simplicity a linear relation between the context information and the resulting QoE (a.k.a., reward in the MAB literature) to capture the first-order impacts. To balance exploration and exploitation at runtime, the model selector in our automated DNN model selection engine uses an LinUCB-style estimate of the QoE in each round, building on the linear minimum mean square error (LMMSE) estimator given by the Bayesian Gauss-Markov theorem [18], [19]. We run a set of simulations to validate our solution. Importantly, our DNN model selection engine addresses the following two practical challenges.

Delayed feedback: Users' QoE feedback may not be observed immediately since a user will unlikely give feedback until it has used the deployed DNN model for a while. In the worse case, a user might not be willing to share its QoE feedback at all. Thus, the QoE feedback signals are delayed for the DNN model selection engine.

Noisy feedback: In practice, depending on system statuses of the edge device (e.g., the number of concurrent processes), latency for edge inference can vary significantly over time, which affects a user's runtime QoE [2]. Thus, even when a user is honest and willing to share its QoE feedback, its true long-term QoE can only become more accurate as it uses the deployed DNN model more times to average out runtime QoE fluctuations.

The key novelty of our research is to make an early step towards automated DNN model selection for edge inference in a scalable manner, incorporating users' QoE feedback into a closed-loop design. The core of our automated DNN model selection engine is to accumulate more refined knowledge based on history decisions and users' QoE feedback, while striking a balance between exploration and exploitation at runtime. In addition to the considered linear contextual MAB, our engine can also incorporate other more sophisticated machine learning techniques such as DNN-based bandits.

The rest of this paper is organized as follows. The challenges of DNN model selection for edge inference and limitations of state-of-the-art solutions are presented in Section II. Section III shows the framework of our proposed DNN model selection engine, along with the design principles and technical



(a) Breakdown of mobile CPU years. (b) CDF of the SoC market share.

Fig. 1: Device statistics for Facebook mobile users [2]. (a) The most common mobile CPUs were designed more than 6 years ago, and only 25% smartphones have CPUs designed in 2013 or later. (b) There exist no standard or typical mobile SoCs: the top 50 SoCs account for only 65% of the smartphone market.

challenges. In Section IV, we present a concrete design based on contextual MAB, including its problem formulation, online learning algorithm, and simulation results. Finally, related works are reviewed in Section V and our conclusion is provided in Section VI.

II. Challenges and Limitations of State of the Art

In this section, we first present the key challenges for automating DNN model selection to achieve QoE-optimal edge inference, and then highlight the key limitations of state-of-the-art DNN model selection approaches.

A. Challenges for QoE-optimal Edge Inference

Automating DNN model selection for QoE-optimal edge inference is challenged by the extremely high degree of heterogeneity for edge inference in terms of the underlying device hardware, usage scenarios and DNN models.

1) Edge device heterogeneity: Take edge inference on mobile platforms alone as an example. Mobile devices have extremely diverse computing and memory capabilities: some high-end devices have state-of-the-art CPUs along with dedicated graphic processing units (GPUs) and even purpose-built accelerators to speed up inference, while many others are powered by CPUs of several years old [2], [12]. As illustrated in Fig. 1, Facebook's 2018 statistics show that nearly 75% of smart phones have CPUs designed more than 6 years ago. Further, mobile SoC vendors typically design their own customized components with blocks licensed from third parties, further contributing to device heterogeneity especially on the open-source Android platform. In fact, there are thousands of unique SoCs running on more than ten thousand different types of smart phones and tablets. Only 30 SoCs can each account for more than 1% of the market share, and these top 30 SoCs cover only 51% of the whole market in total. Consequently, assuming a universal availability of uniform (high-end) hardware for all edge devices is problematic, and the edge device heterogeneity results in a huge variability in QoE: e.g., even with fine-tuned DNN models, inference latency varies by a factor of 10+ among Facebook mobile users [2].

2) Usage scenario heterogeneity: DNN models run on different edge devices can be exposed to significantly different usage scenarios: very different locations, illumination conditions, temperature, etc. These all account to drastically different distributions of users' input data sets, which can thus result in very different inference accuracies even on the same DNN model [3], [8]. Moreover, inference latency differs significantly across even the same generation of devices with the same software configuration, because of runtime environment factors and internal systems statuses (e.g., number of concurrent processes running, battery conditions, etc). Last but not least, users have different preferences towards different metrics, further complicating the scenario: some users are more energy-sensitive due to limited battery capacities, whereas others like to trade energy consumption for latency. Therefore, for different usage scenarios, even the same DNN model on the same type of device may not result in the same QoE.

3) DNN model heterogeneity: The recent studies have proposed various DNN model compression techniques, such as network pruning, weight quantization, low-rank matrix approximation, and knowledge distillation [4], [11], [14], [20]–[26]. As a consequence, even with the same training data set but given different compression techniques or optimization parameters, hundreds of or even more different DNN models can be generated using neural architecture search techniques [9], [10]. While many of the resulting lightweight DNN models can be deployed on a target edge device, they can exhibit very different tradeoffs in a multi-dimension space of important metrics (e.g., accuracy vs. latency vs. energy), and no single model can achieve optimality in all the dimensions [4], [5], [7], [13]. Thus, along with the large set of diverse compressed DNN models comes the model selection difficulty for QoE-optimal edge inference.

B. Limitations of State of the Art

To achieve efficient edge inference, research efforts on optimal DNN model selection for edge devices have been quickly proliferating [3], [6]–[8], [11]–[14]. Nonetheless, they exhibit *one or more of* the following limitations.

1) Lack of wide applicability: A common approach used by the industry is to benchmark on a set of edge devices (e.g., in pilot test or lab settings) and conservatively select a fine-tuned DNN model that is small, computationally cheaper, and meets performance objective for *most* devices. Nonetheless, even for large companies like Facebook, this "one for most" approach can at best cover only a small set of devices under limited usage scenarios, thus lacking wide applicability as edge devices and usage scenarios are extremely diverse. For example, inference latencies for Facebook mobile app users can vary by a factor of 10+, raising serious QoE issues [2].

2) Poor scalability: Many research studies have leveraged AutoML and neural architecture search techniques to identify the optimal DNN model for a *specific* target edge device [4], [5], [9], [10], [13], [15]–[17]. While this "one for one" approach can result in an optimal or near optimal DNN

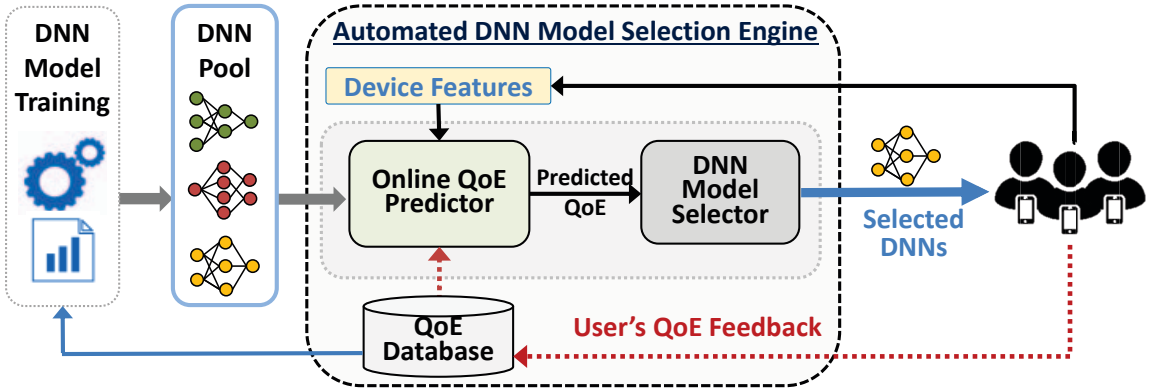


Fig. 2: Overview of our proposed automated DNN model selection engine. For each edge device that requests a DNN model, the selection engine takes as inputs the device features and DNN model features, and outputs a selected DNN model.

model for a specific device under a given set of optimization parameters, it suffers from poor scalability: the same and sometimes lengthy procedure needs to be executed, whenever an unseen new device comes. In parallel, some recent studies have proposed "once for all" training such that one pre-trained DNN model can be converted into many different lightweight versions without re-training, but they can only coarsely select a model for a given application scenario (say, mobile vs. smart home) [14]; in other words, given a particular scenario, how to automatically select a DNN model for an edge device out of the many possibilities in a scalable manner still remains unknown.

3) Without QoE awareness: While commonly-used metrics, such as accuracy, latency and energy, are all important factors, it is ultimately users themselves who decide the actual QoE. Users often have different and possibly dynamic preferences towards different metrics depending on the usage scenario (e.g., energy is not a major concern when devices have large battery capacities and/or are plugged into external power sources). Nonetheless, the way that DNN models are selected for edge inference today is typically "open loop" and assuming a set of pre-determined optimization parameters, such as weights of latency vs. accuracy [4], [5], [13], [16], without incorporating user feedback into a closed loop. Consequently, the optimized metrics may not necessarily translate into improved QoE.

III. Automated DNN Model Selection Engine

The goal of this paper is to envision a new research direction on exploiting machine learning techniques to automatically select pre-trained DNN models for QoE-optimal edge inference in a scalable manner, incorporating actual user experience into a closed loop. Next, we show an overview of our proposed DNN model selection engine, outline the key design principles, and present a few key technical challenges.

A. Overview

The overview of our cloud-hosted automated DNN model selection engine is shown in Fig. 2. Given a large set of DNN models pre-trained in the cloud using state-of-the-art AutoML and neural architecture search techniques [9], [10], [13], [15], our DNN model selection engine constructs a QoE model based on users' QoE feedback and selects appropriate DNN models for edge devices, balancing exploration and exploitation. Specifically, for each incoming edge device,[1] our engine takes as inputs the device features and DNN model features, and outputs a selected DNN model with the goal of optimizing the user's QoE. After DNN models are installed and used for some time, users' QoE feedback is requested for improving future DNN model selections, thus forming a closed loop. In practice, QoE feedback can be obtained by asking users to provide ratings on their experience with our selected DNN models, which is commonly used by many of today's mobile apps.

Note that our research focuses on how to select pre-trained DNN models for edge inference, and hence model training is orthogonal. In practice, the DNN models can be trained offline based on a selected set of testing edge devices such that they are more likely to result in better QoE when initially deployed on target edge devices. Moreover, given users' QoE feedback history, new DNN models may need to be trained and added to expand the model pool, if a new type of edge devices emerge and/or no DNN models in the existing pool can offer a satisfactory QoE.

B. Design Principles

An efficient automated DNN model selection engine cannot be accomplished without the following key design principles.

1) Leveraging predictive power of online learning: The DNN model selection engine needs to serve a huge number of users (e.g., Facebook needs to select DNN models for billions of its mobile app users) [2] and also be continuously updated, thus generating a large amount of history data. Nonetheless, the poor scalability of existing "one for one"

[1]An edge device that requests DNN model updating is considered as a new device, although the previous DNN model selected for this device can be exploited as prior information by our engine.

approaches (i.e., optimize DNN model selection for a specific target device) [4], [5], [9], [10], [13], [16], [17] stems from the fact that they disregard the *similarities* between previous model selections and new ones without exploiting the history data. Intuitively, if certain DNN models have been deployed on certain edge devices with a good QoE, then the same DNN models may also likely fit well into new but similar edge devices with similar usage scenarios (which is similar in spirit to the core idea of recommendation system design based on collaborative filtering [27]). Inspired by this fact, we propose an automated DNN model selection engine based on online learning: as time goes on, more knowledge regarding how good DNN model selection decisions are for given edge devices is being accumulated, thus gradually facilitating future DNN model selections. Note that the "one for most" approach (i.e., selecting a single DNN model for most edge devices) [2] essentially assumes that the optimal mapping of a single DNN model into many edge devices is known through offline profiling in advance, but this is not possible given the vast heterogeneity of edge devices, let alone the even more diverse usage scenarios and user preferences. In sharp contrast, our engine gradually improves DNN model selection by learning the optimal decisions online at runtime.

While our proposed online learning-based approach may not outperform the "one for one" approaches in terms of specific objective metrics [4], [5], [9], [10], [13], [15]–[17], our approach is more scalable to a huge number of heterogeneous edge devices. More importantly, we aim at improving users' QoE, whereas the "one for one" approaches focus solely on optimization of objective metrics such as accuracy and latency.

2) Keeping users in a closed loop: We envision that it is crucial to keep users in a closed loop when designing an automated DNN model selection engine. Naturally, optimized objective metrics (e.g., accuracy, latency, and energy) may not necessarily translate into improved QoE; instead, it is ultimately users themselves who decide the actual QoE. In practice, users can have different and possibly even dynamic preferences towards different metrics depending on the usage scenario (e.g., energy is not a major concern when devices have large battery capacities and/or are plugged into external power sources). There, as illustrated in Fig. 2, based on feedback from users regarding their actual QoE, our DNN model selection engine continuously updates the online QoE predictor as well as the model selector, forming a closed loop with users in the middle.

C. Technical Challenges

To design an automated DNN model selection engine for QoE-optimal edge inference, we outline a few major challenges as follows.

• First, there are many factors that may affect various important metrics for edge inference and affect the users' QoE. For example, factors such as hyperparameters and architecture of the selected DNN model, edge device's hardware configuration, software library, background application on the same device, and environmental contexts (e.g., location, ambient temperature, illumination condition, etc.) can all affect the accuracy, latency, and energy consumption. While taking more factors into account can potentially improve the modeling accuracy of users' QoE, it also complicates the QoE model learning at runtime with likely increased learning time. Moreover, some of these factors, like runtime location and ambient temperature, are not exposed to our DNN model selection engine prior to actual model deployment. Thus, our DNN model selection engine must carefully select a minimum set of key features that are yet representative enough to capture the users' QoE.

• Second, unlike standard recommendation systems where feedback signals are often quickly available (e.g., clicking on a news article indicates the user's interest in the content) [27], the online QoE prediction module in our DNN model selection engine must deal with delayed, even missing, and noisy feedback regarding users' actual QoE. Specifically, users may not provide their accurate QoE feedback right after they install our selected DNN models. Moreover, in view of a possibly large set of pre-trained DNN models, the online learning module needs to strike a balance between exploitation and exploration, in order to gain high QoE while avoiding being trapped in a local optimum.

• Last but not least, the DNN model selection decision may be combinatorial: multiple DNN models need to be selected out of a large pre-trained model sets in order to enhance inference robustness at runtime [8]. Nonetheless, compared to the single DNN model case, selecting multiple DNN models can exponentially increase the complexity of online learning, which thus mandates more efficient online learning techniques in practice.

IV. Contextual MAB for DNN Model Selection

In this section, we present a concrete design of our proposed automated DNN model selection engine based on contextual MAB, including its problem formulation, online learning algorithm, and evaluation. For simplicity, the online QoE predictor in our DNN model selection engine assumes a linear relation between the context information and the reward (i.e., user's QoE in our work) to capture the first-order impacts. Then, to balance exploration and exploitation, the model selector in our engine uses an LinUCB-style estimate of the QoE in each round, taking into consideration users' delayed and noisy QoE feedback. Finally, we validate our design by conducting a simulation study, highlighting the potential of machine learning for automated DNN model selection to achieve QoE-optimal edge inference.

A. Problem Formulation

We formulate the problem of automated DNN model selection into the contextual MAB framework.

1) Contextual MAB: Our automated DNN model selection engine chooses proper DNN models from a set of pre-trained models for incoming edge devices to achieve QoE-optimal edge inference. Thus, by viewing each DNN model as an arm, our problem can be modeled as a contextual MAB problem

that sequentially chooses arms out of a candidate arm set for maximizing some cumulative rewards.

Consider DNN model selection for edge devices over a time horizon of T rounds. Denote the set of pre-trained candidate DNN models to be selected for an edge device at round t by $\mathcal{A}_t = \mathcal{A}$, with $|\mathcal{A}| = A$ where $|\cdot|$ denotes cardinality. Note that our analysis can be extended to the case where the set of DNN models are volatile, since the set of candidate models $\mathcal{A}_t$ can change over time t (e.g., only certain DNN models can be installed on an edge device due to the hardware constraints). For each candidate DNN model $a \in \mathcal{A}_t$, an associated context vector $x_{t,a} \in \mathbb{R}^d$ is observed at round t, where d is the dimension of context vector $x_{t,a}$. The context vector $x_{t,a}$ captures all the available side information, including selected features of both the DNN model a and the incoming edge device such as the DNN model size, architecture, device's CPU, and RAM capacity. Let $r_{t,a}$ be the reward (i.e., QoE in our study) of the DNN model a selected at round t.

Since $r_{t,a}$ is a random variable relying on context $x_{t,a}$, we model it as

$$r_{t,a} = f(x_{t,a}) + \eta_t \tag{1}$$

where $f(x_{t,a})$ is a deterministic function in terms of $x_{t,a}$ and $\eta_t \sim \mathcal{N}(0, \lambda^\eta)$ is the independent and identically distributed (i.i.d.) model mismatch noise. Thus, $f(x_{t,a})$ actually models the expected QoE for a DNN model selection decision on an edge device. For simplicity, we assume that the expected QoE follows a linear model as follows

$$\mathbb{E}[r_{t,a}] = f(x_{t,a}) = x_{t,a}^T \theta \tag{2}$$

where $\theta \in \mathbb{R}^d$, $\|\theta\| \leq 1$, is an unknown weight vector representing the QoE model parameter to be learnt online. Albeit simple, such linear models have been applied in other real applications (e.g., news article recommendation) with satisfactory outcomes [18]. Nonetheless, more complicated relationships between the expected QoE and available contextual information can also be considered by our DNN model selection engine and will be part of our future research.

As in a standard contextual MAB setting, the goal of our DNN model selection engine is to minimize the cumulative *regret* defined as

$$R_T = \mathbb{E}\left[\sum_{t=1}^{T} \left(r_{t,a_t^*} - r_{t,a_t}\right)\right] \tag{3}$$

where $a_t^* = \arg\max_{a \in \mathcal{A}_t} \mathbb{E}[r_{t,a}]$ is the optimal DNN model with respect to the expected QoE and a_t is the DNN model actually selected by our engine at round t.

2) Delayed and Noisy Feedback: Unlike standard recommendation systems where feedback signals are often quickly available [27], there is a delay for a user's QoE feedback to be obtained by our DNN model selection engine for updating the online QoE predictor and improving future decisions. Moreover, because of DNN model latency variations at runtime [2], users may not provide their accurate QoE feedback right after they install our selected DNN models. Here, assume that the user's QoE feedback for the DNN model selected at round t is received after d_t rounds. Then, our engine receives the QoE feedback at $t + d_t$, which is denoted as

$$y_{t+d_t,a_t} = r_{t,a_t} + v_{d_t} = x_{t,a}^T \theta + \eta_t + v_{d_t} \tag{4}$$

where $v_{d_t} \sim \mathcal{N}\left(0, \lambda_{d_t}^v\right)$ is the independent observation noise due to users' runtime QoE variations. In general, the longer time a user uses a selected DNN model, the more accurate QoE it has about the DNN model. Thus, we assume that the variance of the observation noise v_{d_t} decreases monotonically with delay d_t. Note that η_t is the error between our linear expected QoE model and the true QoE, and hence is independent of v_{d_t}. Thus, $\eta_t + v_{d_t} \sim \mathcal{N}(0, \lambda_t)$ where $\lambda_t = \lambda^\eta + \lambda_{d_t}^v$ is the total variance.

TABLE I: Notations

Notation	Description
$\mathcal{A}_t$	Set of candidate DNN models at round t
T	Total number of rounds played
$x_{t,a}$	d-dimensional context vector of DNN model a observed at round t
d_t	Delay of the QoE feedback to DNN model selected at round t
t_c	Longest delay for user's QoE feedback
$y_{t+d_t,a}$	Delayed QoE feedback for DNN model selected at round t
η_t	Model mismatch noise at round t
λ^η	Variance of η_t
v_t	User's QoE observation noise for DNN model selected at round t
$\lambda_{d_t}^v$	Time-varying variance of v_t
$\mathcal{T}_t$	Set of time rounds for which delayed QoE feedback is received by round t
$\mathbf{X}_t$	Horizontal concatenation of context vectors x_{s,a_s} for $s \in \mathcal{T}_t$ and a_s being the corresponding DNN model selected at round s
$\boldsymbol{y}_t$	$\lvert\mathcal{T}_t\rvert$-dimensional vector, with i-th element being the delayed QoE feedback of DNN model associated with i-th column in $\mathbf{X}_t$
$\mathbf{C}_{n,t}$	Covariance matrix of λ_t for all $t \in \mathcal{T}_t$
$\mathbf{C}_{\epsilon,t}$	Covariance of estimation error of $\hat{\theta}_t$
$p_{t,a}$	UCB of estimated QoE for selecting DNN model a in round t

For the readers' understanding, the mathematical notations we use in this paper are listed in Table I.

B. Online Learning Algorithm

Based on history actions and users' QoE feedback, we can build an online QoE predictor to estimate the expected QoE for a given DNN model selection decision on an edge device. Then, one may want to simply make a decision to maximize the expected QoE. Without sufficient exploration, however, this exploitation-based strategy can result in an arbitrarily bad outcome [28], [29]: our DNN model selection engine can be trapped in a local optimum and continuously select sub-optimal DNN models without exploring better solutions. Therefore, the DNN model selector must carefully balance exploration and exploitation, which is a crucial part of online learning and addressed as follows.

Specifically, we design a learning algorithm to make online DNN model selection decisions with the goal of achieving

a sub-linear regret compared to an oracle benchmark that knows the expected QoE perfectly. In our algorithm, a QoE predictor exploiting the delayed feedback is designed and the idea of Upper Confidence Bound (UCB) [18], [30], [31] is employed to balance the exploitation and exploration for online decisions.

The prediction of expected QoE relies on an accurate estimation of the unknown coefficient θ in Eqn. 2. The QoE feedback delay and noise should both be considered in the estimation method. Denote the set of time rounds whose delayed QoE is received by round t as $\mathcal{T}_t$. Namely, for each $s \in \mathcal{T}_t$, the QoE feedback of action made at round s is received at round $s+d_s$ that satisfies $s+d_s \leq t$. Define a matrix $\mathbf{X}_t$ of size $d \times |\mathcal{T}_t|$, with each column being the context vector x_{s,a_s} for $s \in \mathcal{T}_t$. Then, we define a $|\mathcal{T}_t|$-dimensional vector $\boldsymbol{y}_t$, with each element being the corresponding observed QoE feedback y_{s+d_s,a_s}, for $s \in \mathcal{T}_t$. We also define the $|\mathcal{T}_t|$-dimensional noise vector $\boldsymbol{n}_t$ with each element being the corresponding combined noise $\eta_s + v_{d_s}$, for $s \in \mathcal{T}_t$. As explained in Section IV-A2, the variance of $\eta_s + v_{d_s}$ is $\lambda_s = \lambda^\eta + \lambda^v_{d_s}$, which decreases monotonically with d_s. Now, the received rewards at round t can be expressed as

$$\boldsymbol{y}_t = \mathbf{X}_t^T \theta + \boldsymbol{n}_t, \tag{5}$$

where $\boldsymbol{n}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{C}_{n,t})$ and $\mathbf{C}_{n,t}$, which is a diagonal matrix with diagonal elements as λ_s, for $s \in \mathcal{T}_t$, is the covariance matrix of $\boldsymbol{n}_t$.

We use an LMMSE estimator to estimate θ based on the received QoE feedback. Specifically, by applying the Bayesian Gauss-Markov Theorem [19], the estimated coefficient $\hat{\theta}_t$ at round t can be derived as

$$\hat{\theta}_t = (\delta I_{d\times d} + \mathbf{X}_t \mathbf{C}_{n,t}^{-1} \mathbf{X}_t^T)^{-1} \mathbf{X}_t \mathbf{C}_{n,t}^{-1} \boldsymbol{y}_t, \tag{6}$$

where $I_{d\times d}$ is an identity matrix of size $d \times d$ and δ, which relies on the distribution of θ, is a tunable parameter. The covariance of the estimation error above is

$$\mathbf{C}_{\epsilon,t} = (\delta I_{d\times d} + \mathbf{X}_t \mathbf{C}_{n,t}^{-1} \mathbf{X}_t^T)^{-1}. \tag{7}$$

With the estimated coefficient $\hat{\theta}_t$, the predicted QoE for selecting DNN model a is thus $\hat{r}_{a,t} = x_{t,a}^T \hat{\theta}_t$ and its estimation variance is $x_{t,a}^T \mathbf{C}_{\epsilon,t} x_{t,a}$. Accordingly, the UCB of the estimated QoE for selecting DNN model a at round t can be computed as

$$p_{t,a} = x_{t,a}^T \hat{\theta}_t + \alpha \sqrt{x_{t,a}^T \mathbf{C}_{\epsilon,t} x_{t,a}}, \tag{8}$$

where α is a hyperparameter to balance exploration and exploitation: the larger α, the more emphasis on exploration, and vice versa. Then, the DNN model selection rule at round t can be written as

$$a_t = \arg\max_{a \in \mathcal{A}_t} p_{t,a}. \tag{9}$$

Finally, we describe the flow of our online learning for DNN model selection in Algorithm 1.

Algorithm 1 Online Learning for DNN Model Selection with Delayed and Noisy Feedback

Initialize $\mathcal{T}_0 = \emptyset$, $\boldsymbol{X}_t, \boldsymbol{y}_t = [\ ]$;
for $a \in \mathcal{A}_0$ **do**
  Initialize $p_{t,a} = 0$;
end for
for $t = 1, 2, ..., T$ **do**
  if No QoE feedback has been received yet **then**
    Randomly choose a DNN model a_t;
    Continue;
  else
    Observe if any new QoE feedback is received at round t;
    if new QoE feedback received **then**
      Update $\mathcal{T}_t, \mathbf{X}_t, \boldsymbol{y}_t, \mathbf{C}_{n,t}$;
      Compute $\hat{\theta}_t$ and $\mathbf{C}_{\epsilon,t}$ according to Eqns. (6) and (7), respectively;
    else
      $\hat{\theta}_t = \hat{\theta}_{t-1}$, $\mathbf{C}_{\epsilon,t} = \mathbf{C}_{\epsilon,t-1}$;
    end if
    for $a \in \mathcal{A}_t$ **do**
      Compute $p_{t,a} = x_{t,a}^T \hat{\theta}_t + \alpha \sqrt{x_{t,a}^T \mathbf{C}_{\epsilon,t} x_{t,a}}$;
    end for
    Select DNN model a_t according to $a_t = \arg\max_{a \in \mathcal{A}_t} p_{t,a}$;
  end if
end for

C. Simulation Study

We run simulations using a synthetic dataset to preliminarily validate our proposed online learning algorithm, demonstrating its potential for automating DNN model selection to achieve QoE-optimal edge inference. Note that a more rigorous evaluation of our engine, both theoretically and empirically, is left as our future work.

1) Settings: We utilize a synthetic dataset to simulate online learning for automated DNN model selection. The key parameters in our simulation are summarized in Table II.

QoE model. The expected QoE in Eqn. 2 is modeled as a linear function paratermized by θ in terms of contextual information, which is available side information to capture features of both DNN models and edge devices. In our simulation, for each DNN model and edge device, the contextual information is four-dimensional (i.e., $d = 4$), including the numbers of nodes N_{node} and layers N_{layer} of the DNN model and the edge device's CPU capacity and RAM capacity. Additional features, such as the device's battery capacity, can also be incorporated if available.

A user's QoE of edge inference can be affected by multiple metrics, which in our simulation are the accuracy, latency and energy consumption for edge inference. For simplicity, we assume a linear function for the DNN model accuracy $accuracy = g_1(N_{node}, N_{layer})$, although other features such as the DNN model architecture can

TABLE II: Simulation Setup

Parameter	Value
T	2000
A	10
d	4
t_c	40
d_t	$[1, 40]$
α	1.5
δ	0.1
λ^{η}	0.4
$\lambda^{v}_{d_t}$	$\frac{5}{t^2}$
θ	[-0.5, 0.37, -0.36, 0.69]

also be factored in. Additionally, inference latency and energy consumption is affected by the CPU and RAM capacity of an edge device. We denote these two metrics as $latency = g_2(N_{node}, N_{layer}, CPU, RAM)$ and $energy = g_3(N_{node}, N_{layer}, CPU, RAM)$, which are also assumed to be linear functions to capture first-order effects. While users' preferences towards these metrics can vary significantly, we assume that the expected QoE is a linear function of these three metrics. Thus, combined altogether, the expected QoE can be modeled as a linear function in terms of N_{node}, N_{layer}, CPU and RAM, with an unknown parameter θ to be learnt online.

Dataset and parameters. The key parameters in our simulations are listed in Table II. We simulate the DNN model selection process for $T = 2000$ rounds, which we will show is enough for our online learning algorithm to learn the optimal DNN model selection. During each round, there is an edge device that requests a DNN model from our DNN model selection engine. In practice, there can be quite a few DNN models in the pool, but not all of them are suitable for edge devices. Here, we set the number of pre-trained candidate DNN models as $A = 10$, which can be a set of models that are most likely to be suitable for target edge devices based on offline profiling/testing. We generate a uniformly distributed vector $\mathbf{X}$ of size $T \times A \times d$ to represent the available time-varying contexts. For each context vector $x_{t,a}$ of DNN model a at round t, we normalize it to $[0, 1]^d$ without loss of generality. We also generate a d-dimensional vector θ, with $\|\theta\| = 1$, as the ground-truth QoE model parameter to be learnt by our DNN model selection engine. Then, the product of $\mathbf{X}$ and θ is the ground-truth expected QoE, based on which our engine will receive delayed and noisy QoE feedback corresponding to its DNN model selection.

The model mismatch noise η_t and user's observation noise v_{d_t} are both generated subject to Gaussian distribution with zero means. For the ground truth, the maximum expected QoE in our synthetic dataset is 0.76. Thus, we set the variance of the model mismatch noise η_t as 0.4, which is almost 50% of the maximum expected QoE. As for the time-varying variance of user's observation noise v_{d_t}, we set it as $\frac{5}{t^2}$ to model the fact that the longer a DNN model is used by a user, the more accurate QoE the user observes and hence the smaller variance.

In practice, the delay of a user's QoE feedback is uncertain and may be long. Here, we set the cut-off time t_c as 40 (rounds), and uniformly generate a delay d_t between 1 and 40 for the DNN model selected at round t. Thus, the ratio of the maximum delay to the total simulated rounds is 2%: if scaled up, for each learning period of 100 days, we give a 2-day window for a user to provide QoE feedback, which is reasonable in practice.

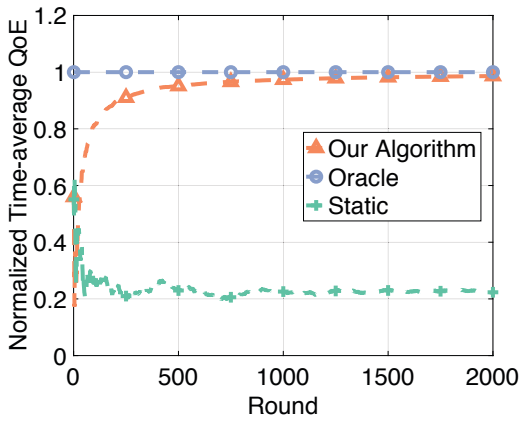


Fig. 3: Normalized time-average QoE vs. number of rounds played. Our online learning algorithm quickly approaches the optimal oracle and significantly outperforms the static selection strategy in terms of time-average QoE.

Baselines. We compare our proposed DNN model selection based on online learning against two baselines: Static and Oracle. As a common practice used by the industry [2], Static is "one for all" and chooses a single DNN model for all the edge devices to maximize the total expected QoE in T rounds. On the other hands, Oracle represents a "one for one" approach (similar to those proposed by recent studies [4], [5], [9], [10], [13], [15]–[17]) and chooses the best DNN model for each individual edge device to maximize the expected QoE, under the assumption that the parameter θ in the QoE model is perfectly known a priori. In practice, this is not possible unless large-scale user trials are conducted in advance, and thus we propose to learn the parameter θ online to gradually improve DNN model selection.

2) Results: We show the simulation results in terms of time-average QoE in Fig. 3. The values are normalized by setting the Oracle's QoE as 1 and minimum as 0. It can be seen that Static yields the lowest QoE, because it does not adapt its DNN model selection to an incoming edge device's contextual information that can significantly vary over time. While Oracle expectedly achieves the highest QoE, our proposed online learning algorithm can quickly approach Oracle in terms of the time-average QoE, with a small gap. The reason is that by exploiting the history DNN model selections and users' QoE feedback (albeit delayed and noisy), our engine can learn the QoE model parameter θ with improved accuracy over time, which is helpful for future DNN model selection. For example, at the end of our simulation, the parameter learnt by our engine is $\hat{\theta}$ = [-0.45, 0.36, -0.31, 0.67], which is very close to the actual ground true of θ shown in Table II.

To sum up, our preliminary evaluation demonstrates that our proposed online learning algorithm can effectively learn the QoE model parameter online even in the presence of delayed and noisy QoE feedback, thus gradually improving the future DNN model selection and approaching the oracle solution in terms of the resulting QoE.

V. Related Work

To enable DNN deployment for inference on resource-constraint edge devices, various model compression methods have been proposed, including network and weight pruning [20], [22], [32], [33], weight quantization [34], [35], low-rank matrix approximation [36], [37], knowledge distillation [21], and/or a combination of basic compression techniques [38]–[40]. To achieve structured pruning, a common approach is channel pruning [22]–[26]. A layer-wise compensate filer pruning algorithm is proposed in [41], and some works are pruning certain structures in DNN [42], [43]. These studies result in different compressed DNN models for the same application, but not a single DNN model can outperform all the others in terms of all performance metrics that are important for users' QoE, thus rasing a follow-up challenge: how to match these DNN models with edge devices for QoE optimization in an automated manner?

Another line of research is running DNN-based inference on edge computing servers/platforms [44]–[47]; conversely, DNNs have been applied to optimize system performances of edge platforms [48]–[50]. Being orthogonal to these studies, our proposed DNN model selection engine focuses on the emerging edge inference that can improve inference performance, is less dependent on network connections, and protect users' privacy without transferring sensitive data to offsite clouds or edge servers.

The existing research on DNN model selection for edge inference typically focuses on a specific target device and optimizes the DNN model based on a set of pre-determined parameters [4], [5], [9], [10], [13], [15]–[17], thus lacking scalability, especially in large systems (e.g., DNN selection for Facebook's mobile users). Conversely, another approach is to fine tune a single DNN model for all or most the edge devices, but this can result significant performance variations on different devices [2]. Most importantly, the existing approaches are "open loop" and optimize for objective metrics such as accuracy and latency, which does not keep users in a closed loop and hence may not lead to the optimal QoE. By contrast, our research makes an early step towards automated DNN model selection for edge inference in a scalable manner, incorporating users' QoE feedback into a closed-loop design.

VI. Conclusion

Motivated by the surging demand for DNN-based edge inference, we study and call for attention to the crucial problem of automated DNN model selection to achieve QoE-optimal inference on heterogenous edge devices. We discuss the challenges of edge inference as well as the limitations of state of the art. Then, we present our vision of exploiting the predictive power of machine learning for automating DNN model selection to achieve QoE-optimal edge inference with users in the loop. We provide an overview of our proposed DNN model selection engine, its design principles as well as research challenges. We also provide a concrete design by formulating the DNN model selection problem into the contextual MAB framework, and propose an online learning algorithm to balance exploitation and exploration in the presence of delayed and noise QoE feedback from users. Finally, we run simulation studies based on synthetic datasets to validate our online learning algorithm. While our current research is preliminary and more sophisticated learning can be leveraged by our engine, the key goal of this paper is to highlight the under-explored potential of machine learning for automating DNN model selection, laying a stepping stone for QoE-optimal edge inference in the future.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[2] C.-J. Wu, D. Brooks, K. Chen, D. Chen, S. Choudhury, M. Dukhan, K. Hazelwood, E. Isaac, Y. Jia, B. Jia, T. Leyvand, H. Lu, Y. Lu, L. Qiao, B. Reagen, J. Spisak, F. Sun, A. Tulloch, P. Vajda, X. Wang, Y. Wang, B. Wasti, Y. Wu, R. Xian, S. Yoo, and P. Zhang, "Machine learning at Facebook: Understanding inference at the edge," in *HPCA*, 2019.

[3] G. Gobieski, B. Lucia, and N. Beckmann, "Intelligence beyond the edge: Inference on intermittent embedded systems," in *ASPLOS*, 2019.

[4] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "Model compression and acceleration for deep neural networks: The principles, progress, and challenges," *IEEE Signal Processing Magazine*, vol. 35, pp. 126–136, Jan 2018.

[5] 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 Electronics*, vol. 1, no. 4, p. 216, 2018.

[6] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep learning for iot big data and streaming analytics: A survey," *IEEE Communications Surveys Tutorials*, vol. 20, pp. 2923–2960, Fourthquarter 2018.

[7] E. Li, Z. Zhou, and X. Chen, "Edge intelligence: On-demand deep learning model co-inference with device-edge synergy," in *MECOMM*, 2018.

[8] B. Taylor, V. S. Marco, W. Wolff, Y. Elkhatib, and Z. Wang, "Adaptive deep learning model selection on embedded systems," *SIGPLAN Not.*, vol. 53, pp. 31–43, June 2018.

[9] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey.," *Journal of Machine Learning Research*, vol. 20, no. 55, pp. 1–21, 2019.

[10] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2017.

[11] A. Ren, T. Zhang, S. Ye, J. Li, W. Xu, X. Qian, X. Lin, and Y. Wang, "Admm-nn: An algorithm-hardware co-design framework of dnns using alternating direction methods of multipliers," in *ASPLOS*, 2019.

[12] Google, "Edge TensorFlow processing unit," in *https://cloud.google.com/edge-tpu/*, 2019.

[13] W. Jiang, X. Zhang, E. H.-M. Sha, L. Yang, Q. Zhuge, Y. Shi, and J. Hu, "Accuracy vs. efficiency: Achieving both through fpga-implementation aware neural architecture search," in *DAC*, 2019.

[14] H. Cai, C. Gan, and S. Han, "Once for all: Train one network and specialize it for efficient deployment," in *arXiv*, 2019, https://arxiv.org/abs/1908.09791.

[15] Q. Lu, W. Jiang, X. Xu, Y. Shi, and J. Hu, "On neural architecture search for resource-constrained hardware platforms," in *ICCAD*, 2019.

[16] S. Yao, Y. Zhao, H. Shao, S. Liu, D. Liu, L. Su, and T. Abdelzaher, "Fastdeepiot: Towards understanding and optimizing neural network execution time on mobile and embedded devices," in *SenSys*, 2018.

[17] W. Jiang, E. H.-M. Sha, X. Zhang, L. Yang, Q. Zhuge, Y. Shi, and J. Hu, "Achieving super-linear speedup across multi-fpga for real-time dnn inference," *ACM Trans. Embed. Comput. Syst.*, vol. 18, pp. 67:1–67:23, Oct. 2019.

[18] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *WWW*, 2010.
[19] S. M. Kay, *Fundamentals of statistical signal processing*. Prentice Hall PTR, 1993.
[20] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *ICLR*, 2016.
[21] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.
[22] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *arXiv preprint arXiv:1608.08710*, 2016.
[23] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang, "Network trimming: A data-driven neuron pruning approach towards efficient deep architectures," *arXiv preprint arXiv:1607.03250*, 2016.
[24] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *ICCV*, 2017.
[25] J. Ye, X. Lu, Z. Lin, and J. Z. Wang, "Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers," *arXiv preprint arXiv:1802.00124*, 2018.
[26] Z. Huang and N. Wang, "Data-driven sparse structure selection for deep neural networks," in *ECCV*, 2018.
[27] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *SIGKDD*, 2015.
[28] A. Slivkins, "Contextual bandits with similarity information," *J. Mach. Learn. Res.*, vol. 15, pp. 2533–2568, Jan. 2014.
[29] T. Zahavy and S. Mannor, "Deep neural linear bandits: Overcoming catastrophic forgetting through likelihood matching," in *arXiv*, 2019, https://arxiv.org/abs/1901.08612.
[30] S. Bubeck, N. Cesa-Bianchi, *et al.*, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *Foundations and Trends® in Machine Learning*, vol. 5, no. 1, pp. 1–122, 2012.
[31] Y. Abbasi-Yadkori, D. Pál, and C. Szepesvári, "Improved algorithms for linear stochastic bandits," in *NeurIPS*, 2011.
[32] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *NeurIPS*, 2015.
[33] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *NeurIPS*, 1990.
[34] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to + 1 or -1," *arXiv preprint arXiv:1602.02830*, 2016.
[35] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *ECCV*, 2016.
[36] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *NeurIPS*, 2014.
[37] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned cp-decomposition," *arXiv preprint arXiv:1412.6553*, 2014.
[38] S. Srinivas and R. V. Babu, "Data-free parameter pruning for deep neural networks," *arXiv preprint arXiv:1507.06149*, 2015.
[39] D. Molchanov, A. Ashukha, and D. Vetrov, "Variational dropout sparsifies deep neural networks," in *ICML*, 2017.
[40] C. Louizos, M. Welling, and D. P. Kingma, "Learning sparse neural networks through l_0 regularization," *arXiv preprint arXiv:1712.01312*, 2017.
[41] T.-W. Chin, C. Zhang, and D. Marculescu, "Layer-compensated pruning for resource-constrained convolutional neural networks," *arXiv preprint arXiv:1810.00518*, 2018.
[42] J. Lin, Y. Rao, J. Lu, and J. Zhou, "Runtime neural pruning," in *NeurIPS*, 2017.
[43] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and J. E. Gonzalez, "Skipnet: Learning dynamic routing in convolutional networks," in *ECCV*, 2018.
[44] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "Mobile edge computing: Survey and research outlook," *arXiv preprint arXiv:1701.01090*, 2017.
[45] P. Liu, B. Qi, and S. Banerjee, "EdgeEye: An edge service framework for real-time intelligent video analytics," in *Proceedings of the 1st International Workshop on Edge Systems, Analytics and Networking*, 2018.
[46] J. Chen, K. Li, Q. Deng, K. Li, and S. Y. Philip, "Distributed deep learning model for intelligent video surveillance systems with edge computing," *IEEE Transactions on Industrial Informatics*, 2019.
[47] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *ICDCS*, 2017.
[48] Y. Han, X. Wang, V. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *arXiv preprint arXiv:1907.08349*, 2019.
[49] X. Li, Y. Qin, H. Zhou, Y. Cheng, Z. Zhang, and Z. Ai, "Intelligent rapid adaptive offloading algorithm for computational services in dynamic internet of things system," *Sensors*, vol. 19, no. 15, p. 3423, 2019.
[50] L. Huang, X. Feng, A. Feng, Y. Huang, and L. P. Qian, "Distributed deep learning-based offloading for mobile edge computing networks," *Mobile Networks and Applications*, pp. 1–8, Nov. 2018.