TRAINABLE POLICY FOR THE ON-DEMAND OFFLOADING ON EDGE-CLOUD COLLABORATIVE SYSTEM

Cheng-Lin Hsieh* Yi-Cheng Lo† An-Yeu (Andy) Wu†

*Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan
†Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan

ABSTRACT

An on-demand offloading framework is a practical solution for resource-limited Internet of Things scenarios. However, an ineffective offloading policy can lead to wasteful transmission costs. Prior works have designed policies based solely on edge information, neglecting the role of the cloud and potentially degrading overall performance. In this paper, we propose two methods to address this issue. First, we modify the training process to incorporate information from both the edge and the cloud, achieving joint edge-cloud optimization in our trainable offloading policy. Second, we leverage structured feature representations to enhance our policy’s efficiency and reduce the cost of ineffective offloading. Our experimental results show that our methods outperform existing approaches on ResNet152 and VGG16, reducing the offloading ratio by 17.74%/23.57% and increasing the offloading efficiency by 4.47%/5.52%, respectively.

Index Terms— Collaborative intelligence, model confidence, computation offloading, metric learning, deep learning.

1. INTRODUCTION

The combination of the Internet of Things (IoT) and convolutional neural networks (CNN) has seen a rise in applications in various fields, such as healthcare monitoring, autonomous vehicles, and smart homes. Nevertheless, edge devices often have limited resources, which has led to increased attention toward edge-cloud collaborative systems [1]. For such systems, the on-demand offloading framework is a practical method, where edge devices perform initial computation on inputs and only transmit necessary information to the cloud. Regarding identifying the necessities, a confidence score is calculated to determine the need for offloading. If the score is high, indicating a well-handled task, the edge avoids transmission; otherwise, indicating a challenging task, the edge on-demand transmits the hyper-features to the cloud.

In the on-demand offloading framework, an efficient offloading policy that judges confidence at the edge is essential. Effective policies are more likely to distinguish between easy and complex tasks, ensuring that each offloaded task is relevant and the available bandwidth is fully utilized. Some previous works use statistical methods such as the softmax value or the entropy of the prediction as their policy [2,3], and others employ neural networks as the estimators [4]. While these previous works achieve favorable reductions in bandwidth, two issues have remained open. First, these works do not consider edge-cloud joint optimization and focus only on estimating the confidence of the edge. However, some tasks have consistent results at both ends and offloading these tasks is unnecessary. Second, the prior works formulate the policy as a pure regression problem [4], regardless of the relationship between the confidence value and the features’ distribution.

In this paper, we propose OffloadNet, a trainable policy network, for the on-demand offloading framework with consideration of edge-cloud joint optimization and the distribution of the features. As shown in Fig. 1, our approach can more accurately offload the necessary tasks and mitigate the transmission cost. In contrast, previous works may waste bandwidth transmitting redundant tasks to the cloud. Our contributions are two-fold:

- Trainable policy with edge-cloud joint optimization: We consider the predictions of not only the edge but also the

![Fig. 1. Comparison between different offloading policies.](image-url)
cloud. Therefore, our OffloadNet can estimate the performance of both edge and cloud to some extent, which can identify the necessary tasks for offloading more accurately and significantly improve the system’s efficiency.

• Leveraging the structure of the embedding features:
We suggest that the confidence relates to the distribution of the embedding features. Since structured features may reveal information about the uncertainty level, we adopt the concept of metric learning to cluster the features, enabling our OffloadNet to estimate the confidence value more accurately.

We validate our method on the CIFAR-100 dataset [5], with ResNet152 [6] and VGG16 [7] as our backbone networks. When, respectively, holding the system’s accuracy at 81.0% and 75.2%, our method significantly reduces the offloading ratio by 17.74% and 23.57% compared to previous methods. Additionally, we increase the offloading efficiency by 4.47% and 5.52% for each experimental setup.

2. BACKGROUND AND RELATED WORKS

2.1. Edge-Cloud Collaborative System with On-demand Offloading Technique

Fig. 2 shows the on-demand offloading edge-cloud collaborative framework, consisting of the edge classifier presented on-device and the cloud classifier at the server [2]. Typically, researchers perform joint training on the edge and cloud classifiers. The loss of training at this stage can be formulated as follows:

\[ L_{\text{class}} = \text{loss}_{CE}(\hat{y}_e, y) + \text{loss}_{CE}(\hat{y}_c, y), \quad (1) \]

where \( \text{loss}_{CE}(\cdot) \) denotes the cross-entropy loss, \( \hat{y}_e \) and \( \hat{y}_c \) denote the output vector of the edge and cloud classifiers respectively, and \( y \) denotes the ground truth label vector for classification.

At inference, the two classifiers solve tasks collaboratively with an offloading policy. The edge classifier first classifies an upcoming task with low computation overhead. Meanwhile, the policy analyzes the output on the classification layer to determine the confidence level of retaining the edge classifier’s result. If a high confidence score occurs, the framework keeps the result of the edge; otherwise, the framework offloads the current task to obtain a more robust classification result from the cloud classifier. Conventional policies rely on the comparison between the user-set threshold and the confidence score, which is calculated from the edge’s classification result. The general form of the offloading policies is detailed as follows:

\[ \hat{y}_f = \begin{cases} \arg \max \hat{y}_e, & c > \lambda, \\ \arg \max \hat{y}_c, & \text{otherwise}. \end{cases} \quad (2) \]

\( \hat{y}_f \) denotes the classification result of the entire framework, \( c \) is the confidence score, and \( \lambda \) is an adjustable threshold set by the users according to the desired offloading ratio.

2.2. Related Works of On-demand Offloading Policy

An effective offloading policy is crucial in the on-demand offloading framework. It can better distinguish between easy and complex tasks, ensuring the relevance of each offloaded task and the full utilization of available bandwidth. On the other hand, a poor policy may lead to incorrect decisions, resulting in the transmission of easy tasks and wasted bandwidth. To the best of our knowledge, previous research on offloading policy can be broadly categorized into two approaches: statistical methods and neural network methods.

Statistical methods perform mathematical calculations on the classifier’s output to estimate the confidence score \( c \), as illustrated in Fig. 2. For instance, Teerapittayanon et al. [2] computed the entropy of the output vector as the confidence score, while Hendricks et al. [3] proposed the maximum class probability (MCP) approach, which uses the maximum softmax value of the output. These approaches leverage the physical characteristics of classifiers and have been shown to achieve fine performance. However, they are susceptible to the overconfident problem, where they present high confidence levels for self-assured predictions, regardless of the accuracy of the classification [4].

To address the issue of overconfidence, neural network methods use an additional policy network to predict the confidence score. These methods focus on the training label for the policy networks. A typical example is the true class probability (TCP) method proposed by Corbière et al. [4]. In this approach, the policy network predicts the softmax value of the true class, or the golden label, instead of using the maximum
softmax value as the confidence score. The ability to predict the true class probability by the policy network has enabled the TCP approach to demonstrate advanced efficiency.

Despite the achievements of the previous methods, we have observed that they do not fully utilize the on-demand offloading mechanism. Previous works have mainly focused on predicting the correctness of edge classification without considering joint optimization with the cloud. Moreover, these works have formulated the policy as a regression problem and ignored valuable information that structural features might contain. As a result, these approaches often lead to unnecessary offloading, limiting the system's full potential.

3. PROPOSED METHOD

We propose a trainable policy realized by our confidence-generating network, OffloadNet, for the on-demand offloading framework. Our main objective is to minimize the communication overhead between the edge and the cloud while maintaining the system’s accuracy. We achieve the goal by two means: 1) We label the training confidence score with consideration for joint optimization of edge-cloud collaboration, enabling more accurate identification of offloading-worthy tasks. 2) We leverage the representations of the embedding features to enhance our policy’s ability to identify tasks’ characteristics, which helps the OffloadNet to decide the need for offloading accurately.

3.1. Derivation and Construction of OffloadNet

Fig. 3 shows the modified framework with our OffloadNet, an additional policy network at the edge end. OffloadNet predicts the confidence score by leveraging the high-dimensional embedding features that contain much abstract information, in contrast to the previous works that relied simply on the classifier’s output. After generating the confidence score \( \hat{c} \), the framework functions identically to the previous works by the equation in (2).

Our approach focuses on enhancing the OffloadNet to predict the confidence score more accurately. To achieve this objective, we adopt the concept proposed by Charles Corbière et al. [4] and introduce a two-stage training process tailored for the edge-cloud collaborative scenario. First, similarly to (1), we jointly train the edge and cloud classifiers in the first training stage. Second, we freeze the gradient of both classifiers and train OffloadNet.

3.2. Confidence Criterion for On-demand Offloading

To identify the essential tasks that should be offloaded, we categorize all tasks in the on-demand offloading scenarios into four types, as shown in Fig. 4. We recognize that only one type of task takes advantage of the offloading mechanism, which are the tasks misclassified by the edge but correctly classified by the cloud, as shown in Fig. 4(c). Contrarily, including the other types of tasks in the offloading mechanism does not improve the framework’s outcome and may even worsen it. Given the observation, incorporating this information into the training stage of the policy is crucial to create an efficient system.

To this end, we incorporate cloud information into our training stage on the OffloadNet by modifying TCP [4]. To start with, the description of TCP is elaborated as follows:

\[
t = P(\hat{y}_e = y^* | w, x),
\]

where \( t \) denotes the TCP value, \( P(\cdot) \) denotes the probabilistic predictive distribution of the classifier given \( w \) are the model parameters.
parameters and \( x \) is the input vector, and \( y^* \) is the truth class. From the formula, we can state that TCP precisely indicates the confidence level of the outcomes from the edge classifier. Next, we adjust the TCP according to the task types illustrated in Fig. 4. Intuitively, our objective is to reduce the confidence level for the tasks suitable for offloading, thereby increasing the likelihood that the cloud classifier will process these tasks. On the other hand, increasing the confidence level of the other tasks is also critical since we aim to prevent inefficient offloading processes. Thus, we derive the adjustment formula as follows:

\[
t_{off} = \begin{cases} 
(t - \epsilon)/(1 - \epsilon), & \text{offloading tasks} \\
(t + \epsilon)/(1 + \epsilon), & \text{early-exiting tasks}
\end{cases}, \tag{4}
\]

where \( t_{off} \) is the adjusted TCP value containing both edge’s and cloud’s information, and \( \epsilon \) is an adjustable parameter depending on the application. Notice that we add the denominator terms to smoothen the value and keep the final result in the range [0, 1].

With our proposed confidence criterion, we train the OffloadNet by regression technique formulated as follows:

\[
L_{off} = \text{loss}_{MSE}(\hat{c}, t_{off}) , \tag{5}
\]

where \( \text{loss}_{MSE}(\cdot) \) denotes the mean-squared-error loss and \( \hat{c} \) is the scalar output from OffloadNet. Eventually, the OffloadNet accurately predicts the feasibility of offloading, comprising the potential confidence level of the task both at the edge and in the cloud.

### 3.3. Advancing Features on Embedding Layers

We consider that confidence is associated with the distribution of the embedding features. As structured features may disclose information on the level of uncertainty, we apply metric learning methods to group features on the embedding layer with large interclass and small intraclass distances [8]. Through modifications to the loss functions during both training stages, we first formulate embedding features into a clustered structure and then translate the distribution into offloading feasibility. The highly clustered features facilitate the differentiation of task types, thus aiding OffloadNet in estimating uncertainty.

In the first training stage, we utilize the supervised contrastive (SupCon) loss [9] to construct structured embedding features, as illustrated in Fig. 5(a). We advance the classifier’s features by modifying (1) into the following form:

\[
\mathcal{L}_{\text{class}} = \text{loss}_{\text{CE}}(\hat{y}_c, y) + \text{loss}_{\text{CE}}(\hat{y}_e, y) \\
+ \text{loss}_{\text{SC}}(\hat{f}_c, y) , \tag{6}
\]

where \( \text{loss}_{\text{SC}}(\cdot) \) denotes the SupCon loss, \( \hat{f}_c \) denotes the feature layer of the edge classifier. The structured features cause tasks of the same class to cluster while pushing tasks of other classes away. Therefore, we can link the confidence prediction to the distribution of features. If the embedding features are located near the cluster’s center, the confidence would be high since this indicates the input follows the features’ structure distribution. On the other hand, if the embedding features are far from any cluster, the confidence would be low since this implies the model is uncertain of which class the task belongs to and may be an outlier. Thus, we assume that the geometric information provided by the structured features enables OffloadNet to identify uncertain tasks more easily.

In the second training stage, when we train our OffloadNet, we exploit the idea of triplet loss [10] to enhance the offloading suitability. The objective is to translate the structured features into the confidence score. We suggest that the early-exiting task with high confidence score may follow the distribution of the embedding features, while the offloading task with a low confidence score does not and tends to scatter randomly. Concerning this, we apply the modified triplet loss on the OffloadNet’s embedding, which aims to exclusively cluster the early-exiting tasks while separating the offloading tasks from them, as shown in Fig. 5(b). We formulate the modified Triplet loss (\( \text{loss}_{\text{T}} \)) as follows:

\[
\text{loss}_{\text{T}} = \| \hat{f}_a^p - \hat{f}_p \|^2 - \| \hat{f}_a^p - \hat{f}_n \|^2 + m_+ , \tag{7}
\]

where \( \hat{f}_p \) and \( \hat{f}_n \) are the positive and negative tasks, corresponding to the early-exiting ones and the offloading ones, respectively. In addition, \( \hat{f}_a^p \) denote the anchors subject to positive tasks, and \( m \) denotes the desired margin to clearly distinguish the positive tasks from the negative tasks, which...
can be adjusted according to applications. Notice that we only apply clustering on the positive case and neglect the negative case, which differs from the original triplet loss. The reason is that the offloading tasks may significantly diverge from each other; therefore, we cannot anticipate the negative tasks to have resemblance behaviors that facilitate clustering. Lastly, we modify the loss term in (5) as follows:

$$L_{off} = \text{loss}_{MSE}(\hat{c}, t_{off}) + \text{loss}_T(\hat{f}_{off}, y_{off}),$$  

where $\hat{f}_{off}$ is the OffloadNet’s embedding, and $y_{off}$ is the one-hot vector indicating that the task is positive or negative.

4. EXPERIMENTS

4.1. Experimental Setup

We conduct two sets of experiments to demonstrate the validity of our policy, respectively selecting ResNet152 [6] and VGG16 [7] as our backbone networks, which act as the cloud classification model for our study. Considering the number of model parameters, we split the network on the 19th layer for ResNet152 and the 4th layer for VGG16. Then, we extend our edge classifier with lightweight convolution layers at the splitting point. Lastly, we construct our offloading policy network, OffloadNet, using five dense layers connected to the penultimate layer of the edge classifier network.

We conduct experiments on the CIFAR-100 [5] dataset, which we divide into 45,000 training images, 5,000 validation images, and 10,000 test images. The performance of our policy is assessed on the test set compared to alternative policies, the entropy-based method [2], MCP [3], and TCP [4]. Regarding policy assessment, we consider the practical limitations of bandwidth and analyze the system performance under various confidence thresholds for offloading ratio. Furthermore, we define the evaluation metric, offloading efficiency, as the proportion of tasks in Fig. 4(c) that are offloaded among all the offloaded tasks to illustrate the effectiveness of the policies.

4.2. Significance of Joint Optimization for Edge and Cloud Ends

Fig. 6 and Fig. 7 show the comparative results of our method with other statistical and neural network policies. The Pareto fronts of the offloading-accuracy trade-off demonstrate the effectiveness of each policy. As depicted by the blue lines on both figures, our method pushes the Pareto front to a better trade-off. Though the method only employs our confidence criterion, the method outperforms prior works, particularly in high-accuracy requirement scenarios.

For statistical methods, the entropy and MCP policies require the framework to offload more tasks to the cloud to achieve comparable accuracy. Also, the inconsistent results under these statistical methods exemplify the overconfident problem. On the other hand, although TCP policy addresses the issue of overconfidence to some extent, there is still room for improvement in the performance of TCP. The observation confirms the importance of jointly considering insights from edge and cloud ends and validates the efficacy of our confidence criterion.

4.3. Contributions of Advanced Embedding Features

Looking into the blue and red lines in Fig. 6 and Fig. 7, we observe that our approach substantially improves with metric learning methods. Our final result pushes the Pareto front to a more satisfactory trade-off, attaining the best performance in all settings. The result demonstrates the effectiveness of enhancing the embedding features. This phenomenon aligns with our assumption that OffloadNet can better predict of-
Table 1: Comparison on different offloading mechanisms under constrained the systems’ accuracy.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Offloading Ratio ↓</th>
<th>Offloading Efficiency ↑</th>
</tr>
</thead>
</table>
| ResNet152 with system’s accuracy equals 81.0%.
  Entropy [2] | 49.04%             | 20.13%                  |
  MCP [3]     | 48.51% (-0.53%)    | 20.37% (+0.24%)         |
  TCP [4]     | 45.04% (-4.00%)    | 21.69% (+1.56%)         |
  Ours        | 31.30% (-17.74%)   | 24.60% (+4.47%)         |

| VGG16 with system’s accuracy equals 75.2%.
  Entropy [2] | 44.04%             | 19.44%                  |
  MCP [3]     | 29.80% (-14.24%)   | 23.22% (+3.78%)         |
  TCP [4]     | 31.02% (-13.02%)   | 22.60% (+3.16%)         |
  Ours        | 20.47% (-23.57%)   | 24.96% (+5.52%)         |

offloading feasibility by leveraging more distinguishable features between tasks suitable for offloading and those that are not. These feature-advancing methods contribute to structured features, facilitating our OffloadNet to identify the necessity of offloading.

4.4. Performance of Policy under Mission-critical Constraint

To analyze the performance of our policy on mission-critical applications, we set the required accuracy to 81.0% for ResNet152 and 75.2% for VGG16. Table 1 shows the comparative results of the offloading ratio and the offloading efficiency. Regarding offloading ratio, our method saves significant communication bandwidth for comparable accuracy. Our approach respectively reduces 17.74% and 23.57% of offloaded tasks compared to the baseline method. The decrease in bandwidth demand is attributed to the enhancement in accurate task identification. Our approach respectively improves offloading efficiency by 4.47% and 5.52% from the baseline method. The high efficiency indicates that our policy precisely offloads tasks misclassified at the edge but classified correctly at the cloud.

5. CONCLUSION

We propose a trainable policy for on-demand offloading on edge-cloud collaborative frameworks. To address the problem of solely focusing on the performance of the edge, we enrich our confidence criterion by integrating cloud information. Additionally, our approach leverages metric learning techniques, which enhance features to assist our policy in identifying offloading tasks. In the experiments, our method outperforms prior works in terms of offloading ratio and offloading efficiency. Therefore, the effectiveness of our policy can be successfully verified.

6. ACKNOWLEDGMENT

This work was supported in part by grants from the Ministry of Science and Technology of Taiwan under Contracts 110-2221-E-002-184-MY3.

7. REFERENCES