QoS Aware Video Analysis over Low-Cost Edge-Cluster: A Utility Minimization Approach

Suvadip Batabyal*, Sudip Misra†, Ozgur Ercetin‡
*Department of Computer Science & Engineering, NIT Durgapur, India
†Department of Computer Science & Engineering, IIT Kharagpur, India
‡Faculty of Engineering & Natural Sciences, Sabanci University, Turkey
*sbatabyal.cse@nitdgp.ac.in, †smisra.editor@gmail.com, ‡oercetin@sabanciuniv.edu

***Abstract*—The constrained availability of resources on an edge analytics platform prompted the need for a trade-off between accuracy and latency by selecting suitable deep neural network (DNN) models on-the-fly. Earlier efforts either used a single powerful multi-core edge computing device or a distributed cluster of edge nodes. While the former has a high cost and power consumption, the latter incur a high communication overhead. In this paper, we propose a quality-of-service (QoS) aware video analytics platform using an edge-cluster made of low-cost devices. The edge nodes, that constitute the cluster, host heterogeneous DNN models having different configurations and number of layers. The nodes cooperate among themselves to jointly process a streaming video to achieve an optimal QoS. We formulate an optimization problem using penalty as the utility function to minimize the long-term average penalty (LTAP). We first design a DNN model recommender algorithm to minimize the LTAP and then compare it with an Oracle to show that it can achieve an LTAP with an error of 1.6% and 9.88% for video resolutions of 720p and 2160p, respectively. We also show that the bounds on LTAP are lower and tighter for lower resolution videos compared to the higher resolution videos.**

***Index Terms*—Edge Computing, Distributed Computing, Deep Neural Networks, Video Processing, Quality-of-Service.**

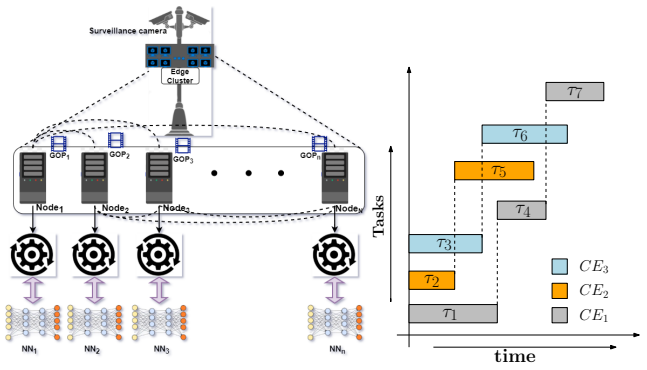
I. Introduction

The exponential growth in surveillance data and the need for its real-time analysis has led to the development of the edge-intelligence paradigm. Edge-intelligence enables near-source data analytics using deep convolutional neural network (DCNN) models. Analyzing video data at or near the source not only reduces the application response time by removing the transmission delay to the cloud, but also saves other network resources. While the edge-intelligence paradigm has several prominent advantages, it often lacks high-end computational resources to analyze large amounts of streaming data in a real-time fashion [1] [2]. This prevents enterprises from deploying state-of-the-art neural network models which demand high-end compute infrastructure for real-time response. Although several attempts have been made towards optimal resource provisioning for real-time video data analytics at the edge, the problem of quality-of-service (QoS) requirements of the application(s) have been overlooked. More importantly, most of these efforts rely on the use of powerful edge devices (such as Jetson Tx2), which demand high and a constant source of power. This prevents an edge-analytics platform from being deployed on mobile and aerial surveillance devices, such as drones, thereby limiting the application to tethered edge devices.

The above challenges of edge-computing paradigm has led to a dichotomous research with the objective to run large DCNN models on low-cost off-the-shelf embedded platforms. On one hand, researchers are pruning/compressing the existing DCNN models to develop new models that fit the memory and computational capabilities of low-cost embedded devices. Few such examples of small DCNN models [3] that can be efficiently run on embedded devices are SqeezeNet, CondenseNeXt, ThiNet [4], etc. The other side of the research is to distribute the analytics task over distributed or localized embedded compute resources to meet the QoS requirements [5] [6]. In this approach, the task of analytics is distributed over a cluster of embedded devices [7] [8], which are then collated to derive the final result. Although such a cluster of embedded devices may be formed as a part of distributed edge-cluster platform, it incurs significant communication overhead. Hence, instead of distributed edge-cluster, researchers are now focusing on the localized edge-cluster [9], where a collection of low-cost off-the-shelf embedded devices are deployed near the data-source to handle the analytics task. A conventional low-cost edge-cluster consists of a collection of low-cost commodity devices (such as laptops), embedded systems (such as a Raspberry Pi running embedded Linux or a google glass), or single-board computers (Panda Boards) closely linked with each other to work as a common system. This not only reduces the transmission and other overheads, but also makes them easier to manage.

An edge-cluster can be best utilized by sub-dividing the tasks and assigning each task to each computational elements (CEs). For example, in a video analytics platform, the streaming video may be split into small segments that is assigned to the computational elements (CEs) such that multiple segments may be independently analyzed in parallel. A popular way to deploy an edge-cluster applications is through containers (such as a Kubernete container [10]). Each CE is capable of hosting multiple containers containing the executable of a DCNN model. Hence, whenever a segment is submitted to a CE for processing, it invokes the container that gives a desirable system performance. For example, a node may contain different (detection and classification) models such as MobileNet, DarkNet, GoogleNet, Resnet, YOLO, etc.; or

ISBN 978-3-903176-73-7 © 2025 IFIP



(a) (b)

Fig. 1: (a) System architecture (b) Task/segment allocation

even different versions of a model such Resnet-18, Resnet-50, YOLOv2, YOLOv3, YOLO-tiny, etc. Therefore, the aim of this paper is to dynamically assign the video analysis task to the *best* DCNN model that results in long-term optimum quality-of-service (QoS). However, the question is *which of these models should be used (or invoked) to process the current task (or video segment)*? In the proposed framework, the QoS is modeled as a utility function that is a product of the penalties incurred with respect to latency and accuracy. Therefore, the objective is to minimize the penalty incurred due to low accuracy and high latency.

While it may be natural to jointly optimize accuracy, latency and resource cost over the entire stream of video, it may not lead to optimal QoS. A surveillance video consists of alternate prosaic sections and important sections. Prosaic sections are those which does not contain any object or event of interest; hence may not require a strong analysis. For example, traffic surveillance may look for vehicles that violate traffic rules. Hence, only the portions of the video after an incident is detected, may need to be processed with a strong DCNN to detect the vehicle with high accuracy. Moreover, the prosaic and important sections have temporal correlation; that is, if a frame does not contain an important object/event, then there is a high probability that the subsequent frame also does not contain important object/event and vice-versa. This temporal correlation leads to better utilization of constrained resources by processing the prosaic portions of the video with a small neural network and important portions of the video with a strong neural network. Therefore, we propose an edge-cluster framework for cooperative processing of streaming video for applications (such as object detection) with the objective to achieve a high QoS. The proposed framework is evaluated using a test-bed that helps in getting some insights on the video resolution to be used, number of nodes required in the cluster, and the number of models required for different QoS and precision values.

II. System Model

Figure 1a shows the architecture of the proposed system. The system consists of a surveillance camera that provides live-feed to the edge-cluster for real-time processing. The cluster is a computational system attached directly to the camera or to a nearby kiosk. The cluster consists of a collection of N low-cost commodity hardware (or computational element (CE)) each with a single CPU. The set of CEs, denoted as $\mathcal{N}$, are connected to each other in a complete mesh fashion. Each CE hosts M neural network models (NNMs), denoted by the set $\mathcal{M}$, and is capable of invoking and running the chosen NNM. The CEs consume the video frames from the ingress buffer and processes the segment using a desirable NNM. This is done under the control of an orchestrator, that keeps track of the segments that are processed by each CE. For simplicity, we assume that the orchestrator assigns the segments in a round-robin fashion to the CEs. For example, if there are three CEs, then segment-1 is allocated to CE-1, segment-2 to CE-2 and segment-3 to CE-3. Once CE-1 completes processing segment-1, segment-4 is allocated to CE-1. It must be noted that the segments have different size (in bytes), and different CEs may use different NNMs to process the segments. Therefore, the processing time of the segments by different CEs are not same, and hence the completion times are not synchronized. This however, does not prevent a CE to start processing the next segment, even if the previous segment is still waiting to be allocated to the designated CE. Figure 1b depicts the allocation of the segments to the CEs. It can be seen that CE-2 incurs less processing time (τ_2) for segment-2 compared to CE-1(τ_1) and CE-3(τ_3), which are processing segment-1 and segment-3, respectively. This however, does not prevent orchestrator from allocating segment-5 to CE-2. Once a GOP is processed, the CEs exchange the instantaneous performance parameters viz., the latency (L_i^j) and the accuracy (A_i^j), directly with each other as they are connected in a fully mesh topology. After receiving the instantaneous parameters, each CE calculates the objective function value, which is then used to decide the NNM to be invoked to process the next GOP in order to achieve the desirable quality of service (QoS).

To maximize the QoS, we employ the method of penalty minimization. Let $\mathcal{C}_i^j$ denote the penalty incurred when the processing time of a GOP i using model j exceeds a certain value. Similarly, let $\mathcal{D}_i^j$ denote the penalty incurred when the (average) accuracy for a GOP i when processed using model j is less than a certain value. In this paper we assume that the minimum accuracy required is 1 (or 100%), which is typically required for mission critical applications. Hence, any value less than 1 will incur a penalty proportional to its deviation from 1. The overall penalty incurred for a GoP i when processed by model j is denoted as ρ_i^j. We use the inverse Cobb-Douglas [11] multivariate utility function to calculate ρ_i^j and denote it as

$$\rho_i^j = (A_i^j)^{-a} \times (\omega L_i^j)^b \tag{1}$$

In (1), ω is a scaling factor to scale the latency between $[1, \infty)$, and $a > 0$ and $b > 0$ are constant exponents that denotes the weights of the accuracy and processing time, respectively. It is easy to see that $\rho(\cdot)$ is a convex function when $a + b \leq 1$. Essentially, $\mathcal{C}_i^j = (\omega L_i^j)^b$ and $\mathcal{D}_i^j = (A_i^j)^{-a}$. The general form of the Cobb-Douglas function is the product of two or more

inputs. In the inverse Cobb-Douglas function, we use a ratio of the two or more inputs to denote the inverse relation of one or more variables with the output variable. In this case, the accuracy holds an inverse relation, while latency holds a direct relation with the penalty (the output variable).

Note that the video is analyzed one GOP at a time (by the CEs) and the objective of the localized edge-cluster platform is the optimal utilization of resources through periodic sharing of performance parameters. Therefore, we try to minimize the long-term average of the penalties (instead of minimizing the instantaneous penalties) incurred for each GOP. The long-term average penalty (LTAP) is calculated as

$$\mathcal{L} = \lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} A_k^{-a} (\omega L_k)^b \tag{2}$$

where $\mathcal{L}$, A_k and, L_k denotes the LTAP, and accuracy and processing time of GOP k under any arbitrary NNM. Therefore, the problem may be formulated as

$$\min_{\mathbf{i,j}} \mathcal{L} \tag{3}$$

$$s.t.\ i \in \mathcal{N}, j \in \mathcal{M} \tag{4}$$

$$\rho(A, L) \leq \sigma \tag{5}$$

Equation (3) denotes the objective to find the desirable node ($i \in \mathcal{N}$) and NNM ($j \in \mathcal{M}$) that minimizes the LTAP over all GoPs under the constraint (5) that the penalty incurred for each GoP is less than a certain value σ. That is, constraint (5) ensures a certain upper bound on the penalty incurred for processing a GoP.

III. Analysis

We now analyze (3), (4) and (5), and find the solution space for it. The live surveillance video is encoded in H.264/SVC format, with a closed GOP structure. Hence, each GOP can be decoded independent of the other GOP, and therefore can be assigned to different CEs for analysis. Let the frame-rate of the video be denoted by f Hz. Therefore, the processing rate (and hence the response time) of any of these models will be greater than $1/f$ seconds per frame. Therefore, $0 \leq L \leq d$, where d is some upper value of processing time for a given model. Similarly, from definition-3, A will lie between $0 \leq A \leq 1$. As mentioned earlier, the processing time of a GOP typically depends on its size (in bytes) and the number of layers in the NNM. In a typical H.264/SVC video, the size of a GOP frame (besides other factors) depends on the number of objects in a key-frame and subsequent predicted frames. In other words, a GoP having more number of objects will typically have a larger size and hence, will consume greater processing time compared to GoPs with smaller size.

To show that there exists an optimal accuracy and latency, we plot the penalty under varying accuracy and latency for $a = 0.6$, $b = 0.4$, and $\omega = 10$ as shown in figure 2a. Figure 2b shows the graph of penalty for different values of a when accuracy (A) is almost zero (0.01). The following points may be noted from figure 2. (i) From figure 2a, we can see that, when the latency is small, the rate of increase in penalty with

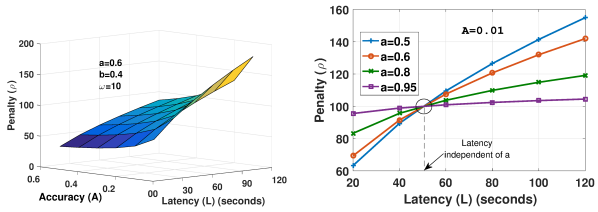
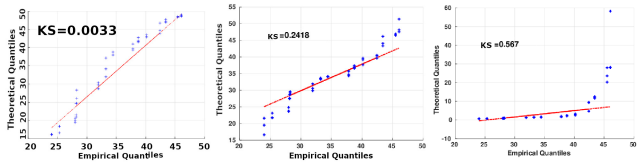


Fig. 2: (a) Penalty under different accuracy and latency for $a = 0.6$, $b = 0.4$ and, $\omega = 10$ (b) Penalty vs latency under varying weight parameters and for accuracy=0.01.

the decrease in accuracy is smaller compared to when the latency is higher. This effect becomes more prominent as we increase ω. This feature signifies that less important sections of a video can be processed by a small DCNN model, such that the penalty incurred is low even if the accuracy is low. (ii) Figure 2b shows the impact of processing time on the penalty by tuning the parameter a (or b) for a given accuracy value. It can be observed that, for a given accuracy, there exists a processing time where the penalty is independent of the weight parameters (viz., a and b). This helps in choosing the right value of the weight parameters depending on the application. For example, mission-critical applications demand low latency and high accuracy. Therefore, a small value of a and a large value for ω will lead to a significantly higher penalty due to high latency or low accuracy.

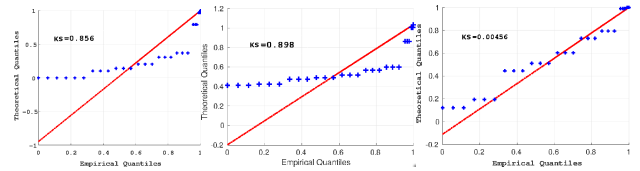
A. Probability distribution of latency and accuracy

The next step is to predict the accuracy and latency of an NNM by utilizing the temporal correlation of the frames in a GOP. For this, we need to find the probability distributions of the accuracy and latency of a video when processed by different DCNN models, and also estimate the parameters of the distributions. We do this by fitting the three (most likely) distributions viz., the uniform, normal, and power-law distributions, and test the goodness-of-fit using the Kolmogorov-Smirnov (KS) test by bootstrapping it with the synthetic data. We also studied the quantile-quantile plot (fig. 3 and fig. 4) comparison between the empirical and the three parametric distributions (mentioned above). All the parametric models are set to take the same median value as the empirical distribution. For this, we first try to find the parameters of each distribution by using the method of curve-fitting. The figures have been plotted for 720p video when processed using Resnet-50+YOLOv3 model. The KS-value shown in the figures, that are obtained from the KS-test, reveal that the uniform distribution is the best fit for the latency and, power-law distribution is the best fit for the accuracy. For the tests, we set the KS-critical value to approximately 0.1. As shown in the QQ-plot of fig. 3, the KS-value for latency is 0.0033 for uniform distribution. Similarly, from the QQ-plot of fig. 4, we can see that the KS-value for accuracy is 0.00456 for the power-law distribution. Since these values are less than the threshold value for both the parameters, we can safely conclude that the latency and accuracy follow uniform and power-law distributions, respectively.



(a) (b) (c)

Fig. 3: QQ-Plot for latency of 720p video using Resnet-50+YOLOv3 model. Distributions are (a) Uniform (b) Normal (c) Power.



(a) (b) (c)

Fig. 4: QQ-Plot for accuracy of 720p video using Resnet-50+YOLOv3 model. Distributions are (a) Uniform (b) Normal (c) Power.

B. Expected value – accuracy and latency

Let $\mathbf{A}$ and $\mathbf{L}$ denote the random variable (r.v.) for accuracy and processing time, respectively. Since accuracy follows a power-law distribution, the pdf of r.v. $\mathbf{A}$ is denoted by $P(\mathbf{A} \leq x) = x^{-\alpha+1}$, where α is the power-law exponent. Similarly, the pdf of r.v. $\mathbf{L}$ has a uniform distribution and is denoted as $\mathbf{L} \sim U(l_1, l_2)$, where l_1 and l_2 denotes the range of the uniform distribution; hence, it also denotes the upper and lower bound of the processing-time. Since, $\mathcal{C}_i^j \propto L^b$, we can define the random variable of the distribution of penalty due to processing time as $\mathbf{Y}_L = \mathbf{L}^b$. Therefore, $f_{\mathbf{Y}_L}(y) = \frac{b}{l_1 - l_2} \times \frac{1}{y^{1-\frac{1}{b}}}$. It can be observed that the processing time is distributed as power-law with an exponent of $\frac{1}{b}$; and since $0 \leq b \leq 1$, $\mathbf{Y}_L$ has a finite mean. Therefore, the mean of $E[\mathbf{Y}_L]$ is given by

$$E[\mathbf{Y}_L] = \frac{l_1^{1+b} - l_2^{1+b}}{(1+b)(l_1 - l_2)} \tag{6}$$

Similarly, $\mathcal{D}_i^j \propto A^{-a}$; and we define a random variable of the distribution of penalty incurred due to accuracy as $\mathbf{Y}_A = \mathbf{A}^{-a}$. Therefore, $f_{\mathbf{Y}_A}(y) = \frac{1-\alpha}{a} y^{\frac{\alpha-1}{a}-1}$. It can be observed that the penalty due to accuracy follows a power-law distribution with a parameter $\alpha - \overline{a+1}$. Therefor, in order to have a finite mean $\alpha > a - 1$.The mean $E[\mathbf{Y}_A]$ is given by

$$E[\mathbf{Y}_A] = \frac{\alpha - 1}{\alpha - 1 + a} \tag{7}$$

C. Proposed Model Selection Algorithm

Algorithm 1 depicts the proposed model selection algorithm to minimize LTAP. We first determine the parameters of the distribution of latency and accuracy for each GOP using a sample surveillance video. The parameters are determined using conventional statistical estimation techniques (described in section 3D). The estimated parameter values are used in the Kalman filter to predict the processing time and the accuracy of the next GOP for each DCNN model. Based on the predicted values, we find the penalty that will be incurred when the GOP is processed by a DCNN model; and, we select the model that gives the minimum penalty. For predicting the accuracy and latency, all the CEs exchange the A_i^j and L_i^j among themselves. The inputs to the algorithm are i, that is, the index of the current GOP to be processed, and a, b, and ω; $\hat{L}_i^j$ is the predicted latency of the i^{th} GOP using the j^{th} model; $\hat{A}_i^j$ is the predicted accuracy of the i^{th} GOP using the j^{th} model.

TABLE I: Summary of estimated parameter values for the fitted distributions.

Parameters	Resnet-50+YOLOv3	Resnet-18+YOLOv2	Resnet-18+YOLOv3(Tiny)
Latency (l_1, l_2)	15.66, 52.33	5.67, 16.75	6.45, 18.7
Accuracy (α)	6.145	15.27	18.33

Algorithm 1 Model Selection(i, a, b, ω)

$s = getSize(i)$ /* get size of GOP i */
for all $j \in M$ **do**
  $\hat{L}_i^j = \text{predict}(i, j, s)$
  $\hat{A}_i^j = \text{predict}(i, j)$
  Calculate $\hat{\rho}_i^j = (\hat{A}_i^j)^{-a} \times (\omega \hat{L}_i^j)^b$
end for
$\min_j \hat{\rho}_i^j$ /* select j having minimum estimated penalty */
Calculate $\mathcal{L}_j$
Exchange $L_i^j, \hat{L}_i^j, A_i^j, \hat{A}_i^j$

D. Estimators

In section 3A, we have shown that latency and accuracy follow a uniform and power-law distribution, respectively. To estimate the parameters of the uniform distribution viz., l_1 and l_2 and the power-law distribution viz., α, we use the method of uniformly minimum variance unbiased estimator (UMVUE). It is well-known that for a uniform distribution $U[\mu - \gamma/2, \mu + \gamma/2]$, the unbiased estimator of μ is $\hat{\mu} = \frac{1}{2}(X_{max} + X_{min})$; and the unbiased estimator of γ is $\hat{\gamma} = \frac{n+1}{n-1}(X_{max} - X_{min})$. Here, $X_{max} = \max\{X_1, X_2, \ldots, X_n\}$, and $X_{min} = \min\{X_1, X_2, \ldots, X_n\}$, where X_i are the observed samples and n is the number of samples. Table I shows the values of the estimated parameters for the fitted distributions for different video resolutions and the three different models used. Once we obtain the estimates of the parameters, we can now apply Kalman filter to predict the values of latency and accuracy.

E. Selecting a, b and ω

Before processing the actual video, we need to determine the weight parameters viz., a, b, and ω for each of the models. The weights are determined offline using a test surveillance video

using the same number of nodes and NNMs as in actual test-bed (or framework). The LTAP is an indirect measure of the trade-off between the average resource utilization and the QoS. As mentioned earlier, the neural network with larger number of hidden layers tend to have higher prediction accuracy, albeit at the cost of higher processing time, and vice-versa. Ideally, the ratio of processing time to accuracy must be same for all the models such that the penalty incurred for a high processing time is compensated by a high accuracy; and the penalty due to low accuracy is compensated by the small processing time. However, this linear relation is not practical and is never observed empirically. Hence, the weights (a, b, ω) are used in order to balance the penalty incurred due to high processing time or low accuracy. Therefore, the values of a, b, and ω should be such that the LTAP for a given data (video) is same for all the models; i.e, $\mathcal{L}_1 = \mathcal{L}_2 = \cdots = \mathcal{L}_m$. For the sake of understanding, let ρ_1^1, ρ_2^1, $\ldots \rho_n^1$ be the penalties incurred when the GOPs $\{1, 2, \ldots, n\}$ are processed using model-1. Similarly, ρ_1^2, ρ_2^2, $\ldots \rho_n^2$ be penalties when the GOPs are processed using model-2. Let the final values of a, b and ω be such, that $\mathcal{L}_1 \approx \mathcal{L}_2$. Now, when the live surveillance video is processed (one GOP at a time by each of the CEs), each time we select the model which is *predicted* to give the least penalty, thereby minimizing the long term average penalty.

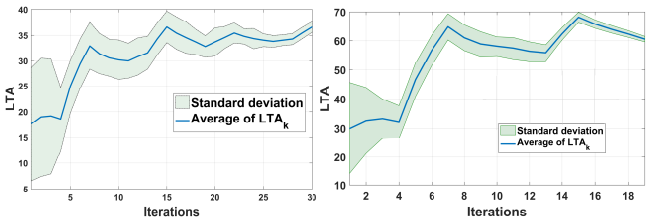
Let a_i, b_i and ω_i denote the weights of the utility function for a given model $i \in M$. Therefore, the initial condition may be written as $(A_i^i)^{-a_i}(\omega_i L_i^i)^{b_i} \leq \sigma, \forall i \in \mathcal{M}$. For the sake of simplicity, let us assume that there are only two models; and let $\epsilon_1 = |(A_1^1)^{-a_1}(\omega_1 L_1^1)^{b_1} - (A_1^2)^{-a_2}(\omega_2 L_1^2)^{b_2}|$ denote the error at the first step (after the first GOP is analyzed by both the models). Therefore, we can arrive at the following recursive equation

$$\epsilon_{k+1} = \epsilon_k + |(A_{k+1}^1)^{-a_1}(\omega_1 L_{k+1}^1)^{b_1} - (A_{k+1}^2)^{-a_2}(\omega_2 L_{k+1}^2)^{b_2}| \tag{8}$$

For the LTAP of both the models to converge to the same value, $|\epsilon_{k+1} - \epsilon_k| \to 0$. This brings us to the fact that if $\rho_i^1 < \rho_i^2$, then ρ_i^1 must be increased and ρ_i^2 must be decreased; and vice-versa. This can be done by incrementing or decrementing the weight parameters as $a_i^{'} = a_i \pm \Delta$, $b_i^{'} = b_i \pm \Delta$, and $\omega_i^{'} = \omega_1 \pm \Delta$, such that the error difference approaches 0.

In (11), at the $k + 1^{th}$ step, we consider only the difference between the penalties to converge the LTAPs of the two models. However in general, when there are more than two models, we use the standard deviation (instead of simple difference) of the instantaneous LTAPs to arrive at the convergence. We start with an initial guess of the weights such that the constraint in (5) is satisfied. It can be observed that the penalty incurred is highest when the accuracy tends to zero and a_i tends to 1; or when processing time tends to infinity, and ω_i tends to infinity and b_i tends to one. Therefore, based on the distribution of the processing time and the accuracy, we initialize the value of the weight parameters such that the penalty constraint is satisfied. In each iteration, we update the values of the weights by incrementing or decrementing them, depending on whether $\rho_i^1 < \rho_i^2$ or $\rho_i^1 > \rho_i^2$. The iteration stops when the error (standard deviation) at the i^{th} step, ϵ_i, is less than a certain threshold δ.

Figure 5 shows the convergence of LTAPs for 720p and 2160p videos, respectively, for the three DNN models. It may be observed that the convergence is faster for 2160p video (18 iterations) compared to the 720p video (30 iterations). This is due to the less variations in model accuracies for higher resolution video. Although the LTAP is higher for higher resolution video, the error is smaller compared to the lower resolution video. For higher resolution video the LTAP is higher due to high processing time. The error is higher for lower resolution video because larger models do not perform well with low resolution videos and have lower accuracies. The same has been experimentally confirmed in section 4A.



(a) (b)

Fig. 5: Convergence of LTA penalty for (a) 720p (b) 2160p videos

The summary of the process flow of the proposed framework is depicted in figure 6. We fist determine the weight parameters (a_i, b_i, and, ω_i) and the distribution parameters (l_1^i, l_2^i and, α^i) for each DCNN model $i \in M$ using a test surveillance video in offline mode. After the system is deployed, the orchestrator assigns the GOPs to the CEs in a round-robin fashion where the model selection algorithm (algorithm-1) is run. If the LTAP variance goes beyond a certain threshold (δ), then the weights and the distribution parameters are calculated again. This may typically happen when there are changes in video characteristics. However, it must be noted that subsequent calculations of the weights are done online and without interrupting the service availability.

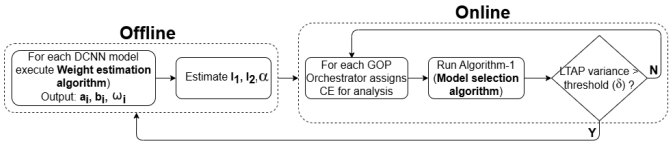


Fig. 6: Summary of the process flow

F. Bounds on LTAP

We shall now find the upper bound on the LTAP usng the method of Markov's inequality. Let $\mathbf{P}$ be the random variable (r.v.) of the penalty. The bound on the penalty may be obtained using the following equation:

$$\begin{aligned} P(\mathbf{P} \geq a) &= P(e^{s\mathbf{P}} > e^{sa}), \ for \ S > 0 \\ &= \int_0^\infty f_{\mathbf{P}}(y) e^{sy} \, dy \end{aligned} \tag{9}$$

TABLE II: Upper bound on LTA penalties for 720p and 2160p video

Resolution	Value of ϵ	Resnet-50 +YOLOv3	Resnet-18 +YOLOv2	Resnet-18 +YOLOv3 (Tiny)
720p	ϵ=0.1	60.5	54.66	51.33
	ϵ=0.01	52.66	49.78	45.50
	ϵ=0.001	46.88	42.33	41.22
2160p	ϵ=0.1	106.7	92.45	87.65
	ϵ=0.01	91.56	84.55	80.22
	ϵ=0.001	86.66	79.88	76.65

TABLE III: Total and average processing time of videos under different models

Resolution	Resnet-50 + YOLOv3	Resnet-18 + YOLOv3 (tiny)	Resnet-18 + YOLOv2
720p	2110s (1.56s/f)	878s (0.65s/f)	592s (0.44s/f)
2160p	1832s (1.36s/f)	532s (0.40s/f)	705s (0.52s/f)

However, it may be noted that the above equation is not bounded and hence we approximate $\mathbf{P}$ with another r.v. $\mathbf{Z}$ as follows

$$Z = \begin{cases} 1 & \text{if } \mathbf{P} \geq a \\ 0 & \text{otherwise} \end{cases}$$

where from Markov's inequality, $P(\mathbf{Y} \geq a) \leq E[\mathbf{Y}]/a$. Let $\mathbf{S} = \sum_i \mathbf{Z}_i$ denote the sum of the r.v. Z_i; and therefore, it denotes the number of times the penalty crosses a certain value η. Therefore, we want to find the probability such that $P(\mathbf{S} = 0) \to 1$. Now, $P(\mathbf{S} = 0)$ is given by $[P(\mathbf{Z} = 0)]^n = [1 - P(\mathbf{Y} \geq \eta)]^n$. From Markov's inequality we know that

$$[1 - P(\mathbf{Z} = 0)]^n \leq e^{-nP(\mathbf{Z}=0)} \tag{10}$$

We want equation (13) to be greater than a certain value $1 - \epsilon$ for $\epsilon \to 0$. Therefore,

$$e^{-nP(\mathbf{Z}=0)} = 1 - \epsilon \tag{11}$$

Therefore, $\eta = \frac{-nE(\mathbf{Y})}{\ln(1-\epsilon)}$

As similar approach may be adhered for finding the upper bound on the accuracy. Table II shows the bounds on the LTAP for different values of ϵ for all the models and video resolutions. It may be observed that as ϵ tend to 0, the bound becomes tighter.

IV. Test-bed Setup and Results

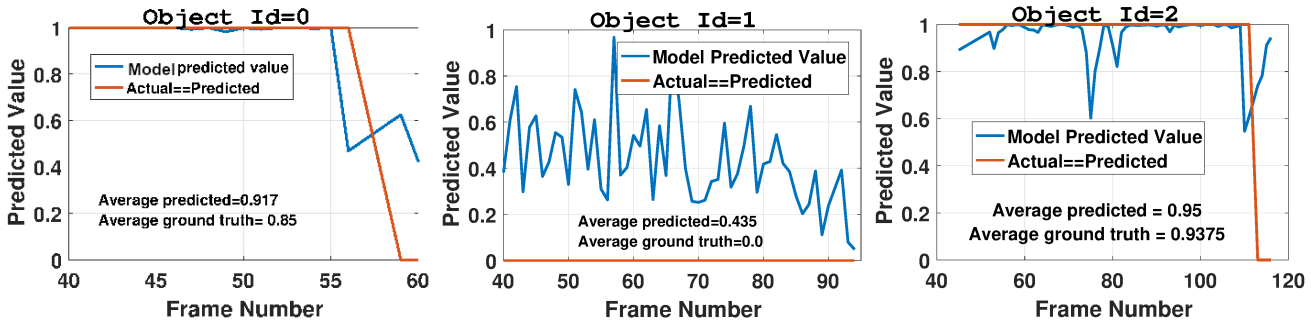
The experiment was conducted on a sample traffic video [12], that captured clear frontal views of cars, thereby facilitating the process of identifying the vehicle make and model easy. The original video of 15 seconds was converted to 45 seconds through repeated appending. The video was encoded in two different resolutions viz., 720p and 2160p with a frame-rate of 30fps. Intra-coded GOP was used with a GOP size of 1 second. The test-bed consists of four low-cost and computationally less powerful single-board systems connected in a fully mesh topology through a common L2-switch that hosted DNN containers. The systems had single-core Intel Celeron processor with a 4GB RAM, frequency of 1GHz, and integrated graphics with a frequency of 350MHz. One system hosted both the orchestrator and the DNN containers. Three DNN models were used that could be run on the containers viz., Resnet-50 with YOLOv3, Resnet-18 with YOLOv2, and Resnet-18 with YOLOv3 (tiny). The total processing time (in seconds) and the average processing time (in seconds/frame) of each of these models for the two different video resolutions are depicted in table III. The process of determining the model accuracies is non-trivial, especially when each frame has multiple objects that have to be detected and classified. In this paper the accuracy and latency is measured in terms of car make and model. Figure 7 shows the predicted and ground-truth values of accuracy of Resnet-50+YOLOv3 model for 720p video for frame numbers from 40 to 120 for three different object, denoted by three object ids. Out of these, object_id-0 and object_id-2 are seen to be predicted correctly with an average (taken over the frames 40–120) value of 0.917 and 0.95, respectively.

A. Experimental Results

The average accuracy of the three models (taken over all the objects) for 720p video was found to be 0.3, 0.3 and 0.25, respectively. Similarly, the average accuracy of the models for 2160p video was found to be 0.32, 0.33, and 0.3, respectively. It may be observed that, while Resnet-18+YOLOv3(Tiny) achieves the maximum accuracy of 0.8 for 720p video, Resnet-50+YOLOv3 achieves the maximum accuracy of 0.9 for 2160p video. Therefore, it may be inferred that models with smaller number of hidden layers tend to have an improved accuracy for low resolution video; and deeper models tend to have better results for high resolution videos. This is essentially because deeper models tend to detect more numbers of car make and models due to overfit for lower resolution videos, which leads to a lower average accuracy.

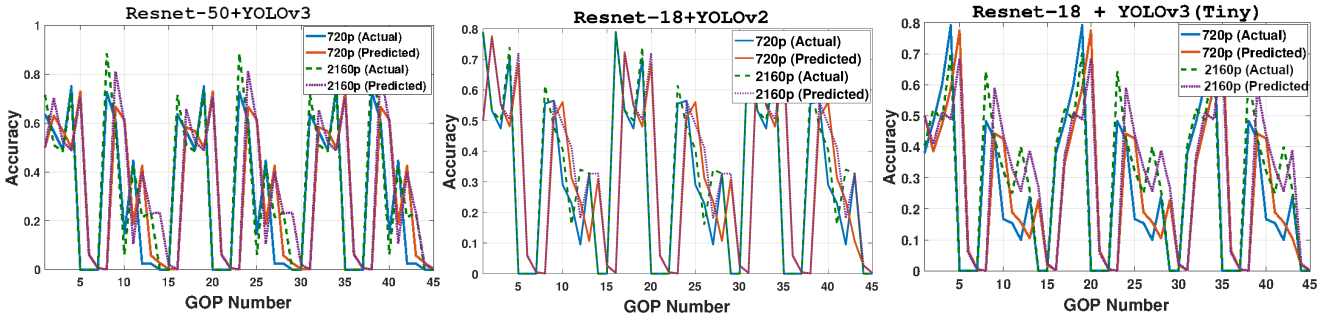
Figure 8 shows the comparison of the actual and the predicted values of the accuracies of each model for different video resolutions viz., 720p and 2160p. In the Kalman filter, for each step, the measured values are obtained from the actual models. The initial estimate of the model, $\hat{p}(0)$, is set to the value such that $P(A \leq a) \geq 0.8$. This is primarily, because the accuracy has power-law distribution. For each of the models, we can observe a few sudden spikes, where the accuracies go beyond a certain value, indicating the power-law distribution of the accuracies. The r.m.s.e between the values obtained from the actual models and the predicted values using the Kalman filter for the 720p videos are 0.35, 0.32, and 0.28, respectively. Similarly, the r.m.s.e for 2160p videos are 0.39, 0.33, and 0.28, respectively. These values show that the predicted values of accuracies closely follow the actual values and provides a reliable predictor.

Figure 9 shows the comparison of the penalties incurred when the video is *exclusively* processed using each model, the Oracle and the penalty incurred using the proposed model selection algorithm (algorithm-1). The σ for the 720p and 2160p video was selected to be 100 and 250, respectively. It



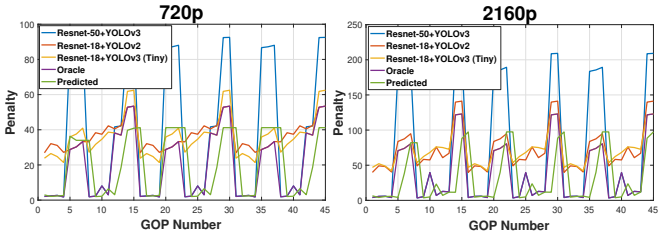
(a) (b) (c)

Fig. 7: Predicted and ground truth values under Resnet-50+YOLOv3 model for 720p video.



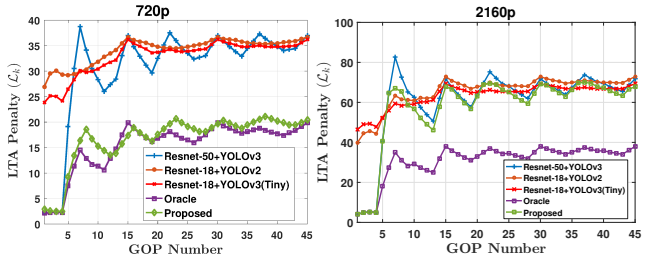
(a) (b) (c)

Fig. 8: Actual and predicted accuracy under different models for (a) Resnet-50+YOLOv3 (b) Resnet-18+YOLOv2 (c) Resnet-18+YOLOv3 (Tiny)



(a) (b)

Fig. 9: Actual and predicted penalty under different models for (a) 720p (b) 2160p video



(a) (b)

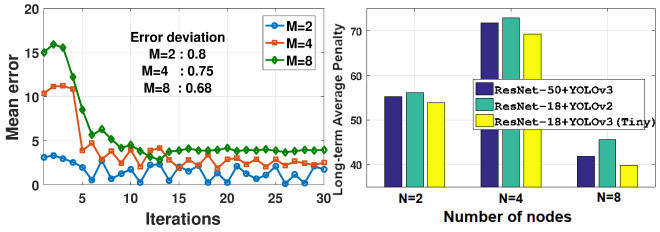
Fig. 10: Actual and predicted LTAP ($\mathcal{L}_k$) under different models for (a) 720p (b) 2160p video

may be observed that the variance in penalty is highest when processed using Resnet-50+YOLOv3 model. This is intuitive since, penalty incurred will be lowest when an object is present and detected correctly; and the penalty will be highest when an object is either not present or not detected accurately. Also the penalty incurred is lower for 720 compared to the 2160p, since the processing time is lower for 720p video. It may be observed that the predicted penalty follows closely with the Oracle with an average deviation of 11.8 and 29.5 for 720p and 2160p videos, respectively. The average deviation is 11.8% and 12%, respectively, of the maximum allowed penalty and therefore, is a good approximation of the Oracle. Moreover, the model selection matches with the Oracle with an accuracy of 73.33% and 71.11%. This also suggests that the proposed algorithm is good approximation of the Oracle.

Finally, in figure 10, we compare the LTAP of the proposed algorithm with the Oracle as well as other models for both the videos. The average deviation is 1.6 and 5.7 for 720p and 2160p videos, respectively. The average deviation is higher for 2160p video because of the selection of a higher maximum penalty.

B. Effect of the number of models

The rate of decrease in mean error of the LTAP for different number of models is shown in fig. 11a. As discussed in section 3E, the rate of decrease in mean error denotes the convergence



(a) (b)

Fig. 11: (a) Convergence of LTAP with different number of models (b) Converged LTAP values for different models with different number of nodes.

rate of LTAP between the different models. From the figure it may be observed that the rate of convergence is longer when the number of models used is small ($M = 2$) compared to when more number of models are used ($M = 8$). It may be observed that while the mean error stabilizes from iteration 15, there is significant deviation in mean-error for M=2. The error deviation for different values of M was found to be 0.8, 0.75 and 0.68, respectively. The reason for slower convergence rate for smaller number of models is the constant change in mean error, leading to a higher deviation. This is because for smaller number of models change in penalty value is higher (especially when the number of hidden layers are significantly different) for smaller changes in weight parameters. However, although the error deviation is higher for smaller number of models, the mean error is higher for higher number of models ($M = 8$). This is due to the significant differences in latency and accuracy values for different models.

C. Effect of number of nodes

The (converged) LTAP for the 2160p video for different values of N for three different models is shown in fig. 11b. It may be observed that the difference is higher when the number of number nodes is more. The maximum difference in LTAP for $N = 2, 4, 8$ is 2.2, 3.56 and, 5.78, respectively. As the number of nodes in the edge-cluster is increased, the variance in distribution of processing time and accuracy is increased. This leads to a higher mean-error between the models when determining the parameter weights, thereby leading to large differences in penalties between the models. The large differences in penalty ultimately leads to higher difference between the LTAPs of each model. It may be observed that the LTAP is highest when $N = 4$. This is due to the particular weight parameters (a_i, b_i, ω_i) decided to calculate the penalty and the error in prediction due to variations in accuracy and latency. We shall investigate this aspect in future and decide on the optimal number of nodes required to achieve a certain LTAP penalty. On the other hand, the LTAP is lowest for $N = 8$, since the average processing time, and hence the penalty, is significantly reduced due to more number of nodes in the cluster.

V. Conclusion

In this paper we propose a framework for QoS aware processing of live streaming video using a localized cluster of low-cost computing elements attached directly to the edge node. Each node has containers containing different deep neural-network models which may be selectively invoked to process each GOP of a video. The QoS is optimized by minimizing the long-term average penalty (LTAP), where penalty is calculated as a utility function. We find that the values for LTAP calculated using different models converge faster for videos with higher resolution compared to videos with smaller resolution. The convergence of LTAP is slower when the number models used are smaller compared to when the number of models used is higher. However, the mean error is smaller for smaller number of models. Increasing the number of nodes/CEs may not reduce the LTAP, and hence may not improve the qos. Increasing the number of nodes may lead to a large variance in penalty values between the models, thereby increasing the overall LTAP. In the future, we intend to investigate this aspect by finding the optimal number of nodes and models to be used which leads to minimal LTAP thereby optimizing the QoS.

References

[1] Y. Jin, B. Huang, Y. Yan, Y. Huan, J. Xu, S. Li, P. Gope, L. Da Xu, Z. Zou, and L. Zheng, "Edge-based collaborative training system for artificial intelligence-of-things," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 10, pp. 7162–7173, 2022.

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

[3] G. Akkad, A. Mansour, and E. Inaty, "Embedded deep learning accelerators: A survey on recent advances," *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 5, pp. 1954–1972, 2024.

[4] J.-H. Luo, J. Wu, and W. Lin, "Thinet: A filter level pruning method for deep neural network compression," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 5068–5076.

[5] M. Hosseinzadeh, A. Wachal, H. Khamfroush, and D. E. Lucani, "Optimal accuracy-time trade-off for deep learning services in edge computing systems," in *ICC 2021 - IEEE International Conference on Communications*, 2021, pp. 1–6.

[6] S. Hu, M. Li, J. Gao, C. Zhou, and X. Shen, "Adaptive device-edge collaboration on dnn inference in aiot: A digital-twin-assisted approach," *IEEE Internet of Things Journal*, vol. 11, no. 7, pp. 12 893–12 908, 2024.

[7] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, "A survey on distributed machine learning," *ACM Comput. Surv.*, vol. 53, no. 2, mar 2020. [Online]. Available: https://doi.org/10.1145/3377454

[8] F. N. Iandola, M. W. Moskewicz, K. Ashraf, and K. Keutzer, "Firecaffe: Near-linear acceleration of deep neural network training on compute clusters," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2592–2600.

[9] X. Yang, Q. Qi, J. Wang, S. Guo, and J. Liao, "Towards efficient inference: Adaptively cooperate in heterogeneous iot edge cluster," in *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, 2021, pp. 12–23.

[10] Z. Zhong and R. Buyya, "A cost-efficient container orchestration strategy in kubernetes-based cloud computing infrastructures with heterogeneous resources," *ACM Trans. Internet Technol.*, vol. 20, no. 2, apr 2020. [Online]. Available: https://doi.org/10.1145/3378447

[11] C. W. Cobb and P. H. Douglas, "A theory of production," *The American Economic Review*, vol. 18, no. 1, pp. 139–165, 1928. [Online]. Available: http://www.jstor.org/stable/1811556

[12] "Video of cars on freeway · free stock video," accessed 2025-02-09. [Online]. Available: https://www.pexels.com/video/video-of-a-cars-on-freeway-6571483/