Hierarchichal Inference using Online Learning

Puranjay Datta, Guide: Prof. Sharayu Moharir
Master's Thesis

1 Introduction

In this research paper [3], the authors delve into the concept of Hierarchical Inference (HI) within the context of Deep Learning (DL) models and their deployment on edge devices. They introduce the terms "S-ML" and "L-ML" to distinguish between two types of DL models.

- S-ML Model: S-ML models refer to small-sized Deep Learning models that are optimized for deployment on moderately powerful edge devices like mobile phones or microcontroller units. These models are designed to minimize bandwidth, energy, and latency costs associated with remote inference in the cloud. However, due to their smaller size, S-ML models might exhibit lower inference accuracy compared to larger, more powerful models.
- L-ML Model: L-ML models, on the other hand, are larger and more sophisticated DL models that can be deployed on Edge Servers (ESs). These models offer higher inference accuracy compared to S-ML models, but they come with higher memory and computational requirements.

The authors propose the concept of Hierarchical Inference (HI) as a way to combine the strengths of both S-ML and L-ML models while mitigating their limitations. The HI framework suggests that an edge device (ED) with an S-ML model should first perform an inference locally. If the S-ML inference is incorrect, the sample can be offloaded to an L-ML model on an Edge Server (ES) for more accurate inference.The challenge lies in making the decision to offload. To address this, the authors introduced a novel HI meta-learning framework. The framework involves determining a threshold value based on the maximum probability value of the S-ML output. If this value is above the threshold, the local inference is accepted; if below, the sample is offloaded to the L-ML model. The decision-making process incorporates a cost structure to optimize the trade-off between accuracy and latency.

The authors present two scenarios for feedback:

- Full Feedback Scenario: In this scenario, local feedback is available, enabling the ED to determine the correctness of its inference.
- No-Local Feedback Scenario: Here, local feedback is not available, introducing additional challenges in decision-making.

To tackle these challenges, the authors design algorithms: HIL-F (HI Learning with Full Feedback) and HIL-N (HI Learning with No-Local Feedback). These algorithms use dynamic non-uniform discretization and leverage the structural properties of the HI problem to achieve effective decision-making while considering feedback availability.The paper showcases that the proposed algorithms perform well in terms of regret bounds, computational complexity, and practical evaluation using real datasets for image classification applications. Overall, the study suggests a comprehensive approach to leveraging DL models on edge devices while maintaining a balance between accuracy and efficiency.

2 System Model and Problem Statement

Each data sample is assigned an index t that indicates its order of arrival in the sequence. When the S-ML model processes a sample, it generates a probability distribution over $\{0, 1\}$ class. p_t represents the highest probability in this distribution, indicating how confident the S-ML model is in classifying the sample. A binary random variable Y_t is introduced to denote the cost associated with the classification of sample t. If Y_t is 0, it means that S-ML correctly classified the sample, and there's no cost incurred. If Y_t is 1, it implies that the classification was incorrect, resulting in a cost. β represents this cost of offloading a sample to the Edge Server (ES). This cost includes factors like energy spent in transmission and reception.

$$D_t = \begin{cases} \text{Do not offload} & \text{if } p_t \geq \theta_t \\ \text{Offload} & \text{if } p_t < \theta_t \end{cases} \tag{1}$$

Therefore, given p_t, choosing threshold θ_t results in a cost/loss $l(\theta_t, Y_t)$ at step t, given by

$$l_0(\theta_t, Y_t) = \begin{cases} f_n.Y_t & p_t \geq \theta_t, \\ \beta & p_t < \theta_t. \end{cases} \tag{2}$$

$$l_1(\theta_t, Y_t) = \begin{cases} f_p.Y_t & p_t \geq \theta_t, \\ \beta & p_t < \theta_t. \end{cases} \tag{3}$$

where f_n, f_p are constants which represent the costs associated with false negatives and false positives, respectively. A false negative occurs when the S-ML model's output is 0 but the true classification by the L-ML model is 1. On the other hand, a false positive happens when the S-ML output is 1, while the L-ML model's true classification is 0.

$$L(\theta^*, Y) = \sum_{t=1}^{n} l_i(\theta^*, Y_t), \quad i \in \{0, 1\}, \text{ if S-ML output is } \{0, 1\} \text{ at sample } t \text{ respectively.} \tag{4}$$

where θ^* need not necessarily be unique and is given by

$$\theta^* = \arg\min_{\theta \in [0,1]} \sum_{t=1}^{n} l_i(\theta, Y_t). \tag{5}$$

Given a sequence Y, we now define the regret under an arbitrary algorithm π as

$$R_n = E_\pi[L(\boldsymbol{\theta}, Y)] - L(\boldsymbol{\theta}^*, Y) \tag{6}$$

3 Background and Preliminary Analysis

3.1 Hedge / Prediction with Expert Advice(PEA) [1]

Algorithm 1 Hedge Algorithm

$W_1(1) \leftarrow 1$
for $t = 1$ to T **do**
  for $i = 1$ to k **do**
    $l_t(i) \leftarrow$ loss of expert i
  end for
  $i_t \sim$ Expert index selected by drawing from $p_t(i) = \frac{W_t(i)}{\sum_{j=1}^{k} W_t(j)}$
  $l_t(i_t) \leftarrow$ Loss incurred at time t
  for $i = 1$ to k **do**
    $W_{t+1}(i) \leftarrow W_t(i) \cdot e^{-\varepsilon l_t(i)}$ ▷ Weight update
  end for
end for

The following theorem provides a guarantee of the performance of the Hedge algorithm. Below, we use the notation $[K] := \{1, 2, \cdots, K\}$

Theorem 3.1. *(Hedge Algorithm Performance). The hedge algorithm achieves performance described by:*

$$\mathrm{E}\left[\sum_{t=1}^{T} \ell_t\left(i_t\right)\right] \leq \underbrace{\min_i \sum_{t=1}^{T} \ell_t(i)}_{(a)} + \underbrace{\epsilon \cdot \sum_{t=1}^{T} \mathrm{E}\left[l_t^2\left(i_t\right)\right]}_{(b)} + \underbrace{\frac{\log N}{\epsilon}}_{(c)}.$$

- **(a)** denotes the minimum loss incurred if a fixed arm were repeatedly pulled throughout the time horizon $[T]$.
- **(b)** describes the sum of the second moments of the losses. It can be trivially upper-bounded by T if the losses $l_t(i)$ are uniformly bounded. When more structure regarding $l_t(i)$ is known, the second moment can be more explicitly computed, resulting in a tighter upper bound for this term. This idea will appear in the upcoming discussion of the Exp3 algorithm.
- **(c)** This term describes how, as the number of experts, N, increases, it becomes more difficult to achieve performance close to that of the best expert. However, this term scales only sub-linearly in N, and thus will not deteriorate the algorithm's performance too much. This term, together with the "trivial" bound for the term in **(b)**, yields a performance of $\Theta(\sqrt{T}\log N)$ for an appropriately chosen $\varepsilon > 0$.

3.2 HIL-F using a single expert [3]

To determine a suitable threshold θ_t in round t, we draw inspiration from the discrete PEA, where an expert's weight is calculated using the exponential of scaled cumulative losses incurred for potentially selecting that expert. We expand on this idea and define a continuous weight function $w_t(\theta)$ as follows:

$$w_{t+1}(\theta) = e^{-\eta \sum_{\tau=1}^{t} l(\theta, Y_t)} \tag{7}$$

$$= e^{-\eta \sum_{\tau=1}^{t} l(\theta, Y_t)} e^{-\eta l(\theta, Y_t)} \tag{8}$$

$$= w_t(\theta) e^{-\eta l(\theta, Y_t)}, \tag{9}$$

where $\eta > 0$ is the learning rate. At each round t, normalized weights yield the probability distribution for selecting the next threshold θ_{t+1}, aiding in learning the system. However, two challenges arise: (i) finding thresholds that adhere to this distribution, and (ii) computing the integral. While direct numerical methods can address these challenges, they come with high computational costs. Instead, we leverage the observation that the final decision (to offload or not) depends only on the relative positions of θ_t and p_t, but not directly on θ_t. Thus, we define q_t as the probability of not offloading, i.e., the probability that θ_t is less than p_t, using the distribution given by the normalized weights:

$$q_t = \frac{\int_0^{p_t} w_t(x)dx}{\int_0^1 w_t(x)dx}. \tag{10}$$

Having addressed the first challenge, we seek efficient methods for computing the integral in q_t. Note that $\sum_{\tau=1}^{t} l(\theta_\tau, Y_\tau)$ can take on three different values (0, 1, or β) in each step, without a necessary pattern, precluding direct analytical integration. To overcome this, we use the result of Lemma 4.1, transforming the integral into a summation by discretizing the domain [0, 1] into a finite set of non-uniform intervals.

The non-uniform discretization proposed by this lemma is incremental, adding new intervals in each round. After n rounds, the structure of the weight function is as follows. Let $p_0 = 0$ and $p_N = 1$, where N is the number of intervals formed in $[0, 1]$ by the sequence of probabilities $\mathbf{p}_n$. Here, we have $N \leq n+1$ due to repeated probabilities that do not result in new intervals. Denote these intervals as $B_i = (p[i-1], p[i])$, $1 \leq i \leq N$, where $p[i]$ is the i-th smallest distinct probability in $\mathbf{p}_n$. Let m_i, $1 \leq i \leq N$, be the number of times $p[i]$ is repeated in $\mathbf{p}_n$. For instance, $N = n + 1$ and $p[i] = p_i$ if $m_i = 1$ for all i. Finally, let $Y[i]$, $1 \leq i \leq n$, be the i-th element in the ordered set of local inference costs, sorted according to the increasing values of the corresponding probability p_i. Note that i in $Y[i]$ ranges up to n, while i in $p[i]$ ranges only up to N, as any two local inference costs Y_j and Y_k associated with repeated probability values $p_j = p_k$ are two distinct but independent and identically distributed random variables.

In this section, we consider the full-feedback scenario, where the algorithm receives the ground truth Y_t for all the samples, including those that are not offloaded by accepting the S-ML inference. For this scenario, we present the HIL-F algorithm in Algorithm 1. Algorithmic rules for the parameter updates are provided in Section 7. When given p_t, we compute q_t, the probability of not offloading. After the decision is made using q_t, costs are received, and the weights are updated using (4) and (5). For simplicity, we denote the expected cost received by HIL-F in round t as $\bar{l}(\theta_t, Y_t)$, which is given by

$$\bar{l}(\theta_t, Y_t) = E_{Q_t}[l(\theta_t, Y_t)] \tag{11}$$

$$= Y_t q_t + \beta(1 - q_t). \tag{12}$$

Here, the expectation is with respect to the probability distribution determined by q_t. Additionally, let $L(\theta, \bar{Y})$ denote the total expected cost after n rounds. In Theorem 5.1, we provide a regret bound for HIL-F.

Theorem 3.2. *For $\eta > 0$, HIL-F achieves the following regret bound:*

$$R_n = \bar{L}(\theta^*) - L(\theta, \bar{Y}) \leq \frac{1}{\eta} \ln \frac{1}{\lambda_{min}} + \frac{n\eta}{8},$$

where $\lambda_{\min}$ is the minimum interval length and can be approximated by $\frac{1}{n+1}$.

3.3 Hedge Analysis for Constant Learning Rate [1]

In the context of the hedge setting, the task of the learner involves making decisions in each round $t = 1, 2, \ldots$ with respect to a weight vector $w_t = (w_{t,1}, \ldots, w_{t,K})$ across K "experts." Consequently, Nature reveals a K-dimensional vector containing the losses of the experts $\ell_t = (\ell_{t,1}, \ldots, \ell_{t,K}) \in R^K$. The learner's loss is represented by the dot product $h_t = w_t \cdot \ell_t$, which captures the expected loss if the learner employs a mixed strategy that selects expert k with probability $w_{t,k}$. We denote cumulative versions using capital letters, and vectors are denoted in bold. Hence, $L_{T,k} = \sum_{t=1}^{T} \ell_{t,k}$ represents the cumulative loss of expert k up to round T, and $H_T = \sum_{t=1}^{T} h_t$ stands for the learner's cumulative loss, often referred to as the "Hedge loss."

The performance of the learner is evaluated based on her regret, which quantifies the difference between her cumulative loss and the cumulative loss of the best expert:

$$R_T = H_T - L_T^*, \quad \text{where} \quad L_T^* = \min_k L_{T,k}.$$

To analyze the Hedge strategy effectively, a pivotal concept is the mix loss approximation, defined as:

$$m_t = -\frac{1}{\eta} \ln(w_t \cdot e^{-\eta \ell_t}),$$

where m_t accumulates to $M_T = \sum_{t=1}^{T} m_t$. This approximation streamlines the analysis and finds relevance in various proof techniques. When considering Hedge and mix loss for $\eta = \infty$, allowing η to tend to infinity extends their definitions. Specifically, Hedge with $\eta = \infty$ results in a uniform distribution over the experts with the smallest cumulative loss up to time t, while the mix loss m_t converges to $L_t^* - L_{t-1}^*$ as η approaches infinity.

In approximating Hedge loss h_t with mix loss m_t, the discrepancy $h_t - m_t$ is denoted as the mixability gap δ_t. This error constitutes a fundamental notion in the analysis of Hedge-type algorithms and plays a vital role in sequential prediction scenarios.

The objective is to bound the mixability gap δ_t, often achieved using tools such as Hoeffding's bound on the cumulant generating function. The cumulative mixability gap $\Delta_T = \delta_1 + \ldots + \delta_T$ accounts for the error introduced by approximating Hedge loss with mix loss. The regret R_T for the Hedge strategy is then decomposed into the contribution from the mix loss, $M_T - L_T^*$, and the cumulative approximation error, Δ_T:

$$R_T = (M_T - L_T^*) + \Delta_T.$$

The Mix Loss with Constant Learning Rate lemma encapsulates fundamental properties of this analysis:

1. Mix loss is consistently lower than Hedge loss ($m_t \leq h_t$), resulting in $\delta_t \geq 0$. For losses in the range [0, 1], $m_t \geq 0$ and $h_t \leq 1$, implying $\delta_t \leq 1$.
2. Cumulative mix loss M_T is expressed as $M_T = -\frac{1}{\eta} \ln(w_1 \cdot e^{-\eta L_T})$.
3. Cumulative mix loss approximates the loss of the best expert: $L_T^* \leq M_T \leq L_T^* + \frac{\ln K}{\eta}$.
4. Cumulative mix loss M_T decreases monotonically as η increases.

To bound Hedge loss, a well-known upper bound for the mixability gap, derived using Hoeffding's bound on the cumulant generating function, is employed. The relation $m_t \leq h_t$ is leveraged to establish $\delta_t \leq \frac{\eta}{8}$. Consequently, $\Delta_T \leq \frac{T\eta}{8}$. When combined with the bound $M_T - L_T^* \leq \frac{\ln(K)}{\eta}$ from mix loss property 3, it results in the following overall regret bound:

$$R_T = (M_T - L_T^*) + \Delta_T \leq \frac{\ln(K)}{\eta} + \frac{T\eta}{8}.$$

The optimal learning rate $\eta^* = \sqrt{\frac{8\ln(K)}{T}}$ balances the two terms and yields a regret bound of $\sqrt{T\ln(K)/2}$, matching the worst-case regret lower bound established in the literature. This optimal rate can be employed when the time horizon T is known in advance. In scenarios where T is unknown, the doubling trick can be utilized, albeit it may result in a larger constant factor in the leading term of the regret bound.

3.4 Adaptive Hedge [2]

Previous Section Recap and Adaptive Learning Rate: In the previous section, we divided the regret for the Hedge algorithm into two components: $M_T - L_T^*$ and Δ_T. We then derived bounds for both components. While other Hedge approaches typically balance these components by considering an upper bound on Δ_T and tuning the learning rate η accordingly, AdaHedge takes a different approach. Instead of relying on an upper bound for Δ_T, AdaHedge aims to equalize Δ_T and $\ln(K)/\eta$ directly. Since the cumulative mixability gap Δ_T is monotonically increasing and can be tracked online, it's possible to adapt the learning rate directly based on Δ_T. The doubling trick is a simple method to achieve this balance: in each subsequent block, the learning rate is halved, and a new block begins when the observed cumulative mixability gap Δ_T surpasses the mix loss bound $\ln(K)/\eta$. This ensures that these two quantities are equal at the end of each block. An earlier version of AdaHedge employed this approach (Van Erven et al., 2011). However, a more elegant method involves adjusting the learning rate over time using the following formula:

$$\eta_{\text{ah}}^t = \frac{\ln K}{\Delta_{\text{ah}}^{t-1}} \tag{13}$$

(Here, note that $\eta_{\text{ah}}^1 = \infty$.) The definitions of weights and the mix loss (equations (14) and (15)) are modified to incorporate this variable learning rate:

$$w_{\text{ah}}^{t,k} = w_{\text{ah}}^{1,k} e^{-\eta_{\text{ah}}^t L_{t-1,k}} \tag{14}$$

$$w_{\text{ah}}^1 = \left(\frac{1}{K}, \ldots, \frac{1}{K}\right) \tag{15}$$

$$m_{\text{ah}}^t = -\frac{1}{\eta_{\text{ah}}^t} \ln\left(w_{\text{ah}}^t e^{-\eta_{\text{ah}}^t \ell_t}\right) \tag{16}$$

Lemma 3.3. *Learning Rate Strategy Comparison:Consider any strategy for selecting learning rates, denoted as "dec", where $\eta_1 \geq \eta_2 \geq \ldots$ holds. Then, the cumulative mix loss for the "dec" strategy does not surpass the cumulative mix loss incurred by the strategy employing the final learning rate η_T from the beginning:*

$$M_{dec}^T \leq M(\eta_T)^T$$

Lemma 3.4. *AdaHedge Regret: The regret of AdaHedge is denoted as R_{ah}^T, and it can be expressed as follows:*

$$R_{ah}^T = M_{ah}^T - L_T^* + \Delta_{ah}^T \leq 2\Delta_{ah}^T.$$

Lemma 3.5. *(Bernstein's Bound):Let $\eta_t = \eta_{alg,t} \in (0, \infty)$ denote the finite learning rate chosen for round t by any algorithm denoted as alg. For losses in the range $[0, 1]$, the mixability gap $\delta_{alg,t}$ satisfies:*

$$\delta_{alg,t} \leq e^{\eta_t} - \eta_t - 1 \leq \frac{v_{alg,t}}{\eta_t}$$

Furthermore, $v_{alg,t} \leq \hat{v}_{alg,t}(1 - \hat{v}_{alg,t}) \leq \frac{1}{4}$.

Lemma 3.6. *For losses in the range $[0, 1]$, the cumulative mixability gap of AdaHedge satisfies:*

$$(\Delta_{ah,T})^2 \leq V_{ah,T} \ln K + \left(1 + \frac{2}{3}\ln K\right)\Delta_{ah,T}.$$

Theorem 3.7. *For losses in the range* $[0,1]$*, the regret of AdaHedge is bounded by:*

$$R_{ah,T} \leq 2\sqrt{V_{ah,T} \ln K} + \frac{4}{3} \ln K + 2.$$

Lemma 3.8. *Suppose* $H_{ah,T} \geq L_{*,T}$*. For losses in the range* $[0,1]$*, the cumulative loss variance for AdaHedge satisfies:*

$$V_{ah,T} \leq L_{*,T} \left(T - L_{*,T}\right) + 2\Delta_{ah,T}.$$

Theorem 3.9. *Theorem 8:For losses in the range* $[0,1]$*, AdaHedge's regret is bounded by:*

$$R_{ah,T} \leq 2\sqrt{L_{*,T}(T - L_{*,T})}\sqrt{T \ln K} + \frac{16}{3} \ln K + 2.$$

3.5 Another variant of ada hedge-doubling trick

The proofs are simpler and bounds are less strong compared to the above ada hedge.

Algorithm 2 AdaHedge Algorithm

Require: $\phi > 1$
  $\eta \leftarrow \phi$
  for $t = 1, 2, \ldots$ **do**
    if $t = 1$ or $\Delta \geq b$ **then**
      Start a new segment
      $\eta \leftarrow \eta/\phi$
      $b \leftarrow \left(\frac{1}{e-1} + \frac{1}{\eta}\right) \ln(K)$
      $\Delta \leftarrow 0$
      $w = (w_1, \ldots, w_K) = \left(\frac{1}{K}, \ldots, \frac{1}{K}\right)$
    end if
    Make a decision
    Output probabilities w for round t
    Actions receive losses ℓ_t
    Prepare for the next round
    $\Delta \leftarrow \Delta + w \cdot \ell_{t+1} + \frac{1}{\eta} \ln(w \cdot e^{-\eta \ell_t})$
    $w \leftarrow \frac{w_1 \cdot e^{-\eta \ell_1 t}, \ldots, w_K \cdot e^{-\eta \ell_K t}}{w \cdot e^{-\eta \ell_t}}$
  end for

4 HIL-F using separate set of experts

Algorithm 3 The HIL-F Algorithm for Full Feedback

function HIL-F($data, \beta, CLASSID, \eta, repeats$)
  Set $w_1(\theta) = 1$ for all $\theta \in [0, 1]$
  Set $v_1(\theta) = 1$ for all $\theta \in [0, 1]$
  Set $N = 1$
  for every sample in round $t = 1, 2, \ldots$ **do**
    S-ML outputs p_t and $classMax\_id$
    if $classMax\_id = 1$ **then**
      Compute q_t using (10) and generate Bernoulli random variable Q_t with $P(Q_t = 1) = q_t$
      $f = f_p$
    else
      Compute q_t using (11) and generate Bernoulli random variable Q_t with $P(Q_t = 1) = q_t$
      $f = f_n$
    end if
    if $Q_t = 1$ **then**
      Accept the S-ML inference and receive cost $f.Y_t$
    else
      Offload the sample and receive cost β
    end if
    Find the loss function using (7)
    if p_t is not a repetition **then**
      Update the intervals by splitting the interval containing p_t at p_t. Increment N by 1
    end if
    Update the weights for all intervals using (4), based on the interval positions with respect to p_t
  end for
end function

4.1 Data Generation

A synthetic dataset is generated to emulate both large and small-scale machine learning models. This dataset comprises three columns: the mode probability p_t, the S-ML output, and the corresponding ground truths.

The initial step involves determining the ratio of 1's and 0's in the ground truth, denoted as p_ground_truth. Subsequently, mode probabilities are assigned to samples with ground truths as 1 using the parameter x, which governs the proportion of samples where $p_t > \theta_1^*$. Similarly, another parameter y controls the ratio of samples where $p_t > \theta_0^*$ for samples with ground truths as 0.

In the subsequent analysis, we focus solely on samples with a ground truth value of 1. For these samples, the determination of the S-ML output involves assessing whether to flip the ground truth based on the assigned p_t value. If $p_t > \theta_1^*$, the S-ML output is assigned as 1 with a high fixed probability. Conversely, if p_t is below θ_1^*, the output is assigned as 0 with a fixed high probability. This process is repeated for samples with a ground truth of 0, based on a similar rationale.

4.2 Effect of the false postive cost f_p

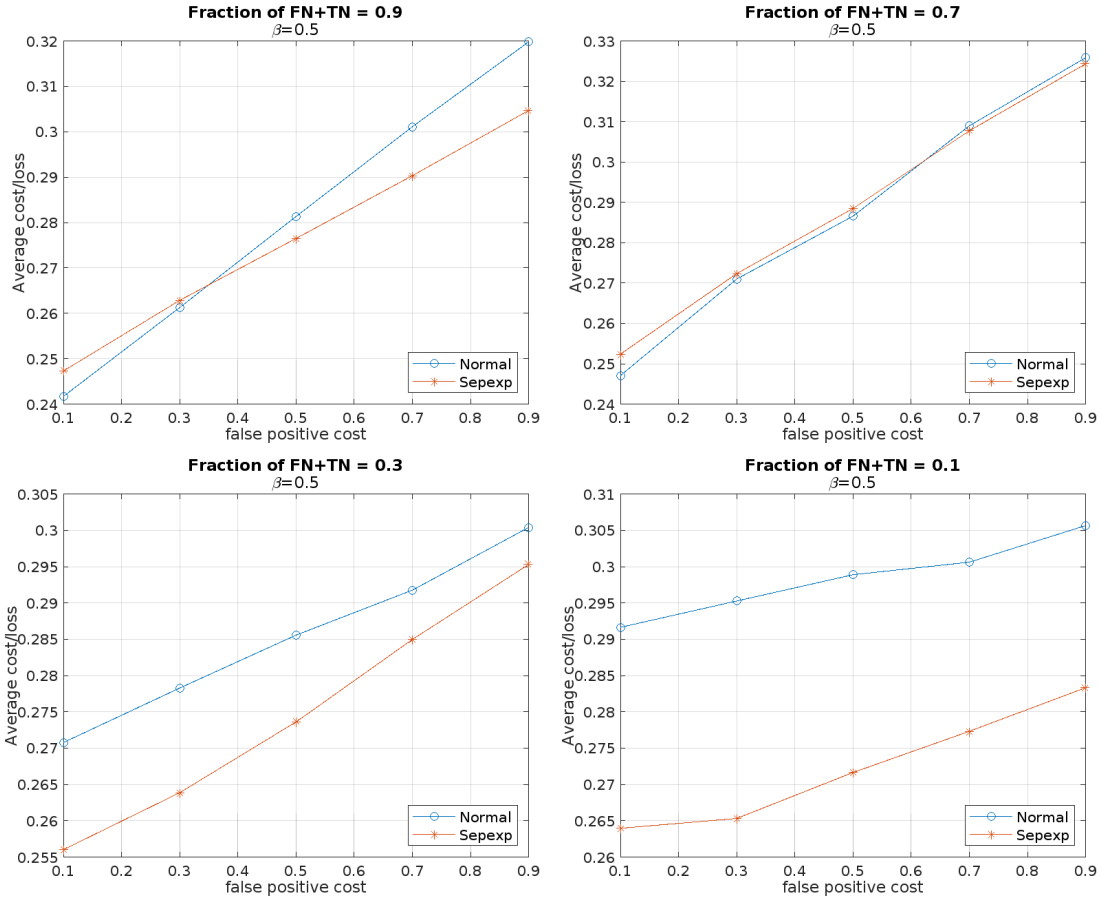

Figure 1: Average cost vs false postive cost f_p

The provided graphs have been created by maintaining a constant value of $fn = 1$ and $\beta = 0.5$, while adjusting the parameters $fn + tn$ and fp. y co-ordinate of each graph represents an average of costs obtained from various combinations of x and y as defined in the data generation process.x defines the distribution of f_p, t_p. From the analysis of the graphs, we observe that as $fp + tp$ increases, the loss incurred by the standard experts experiences a noticeable degradation compared to the separate experts approach. Conversely, for scenarios characterized by lower false positive costs and higher ratios of $fn + tn$, the standard experts tend to outperform the separate experts strategy.

4.3 Effect of the ratio of number of $0's$ in ground truth

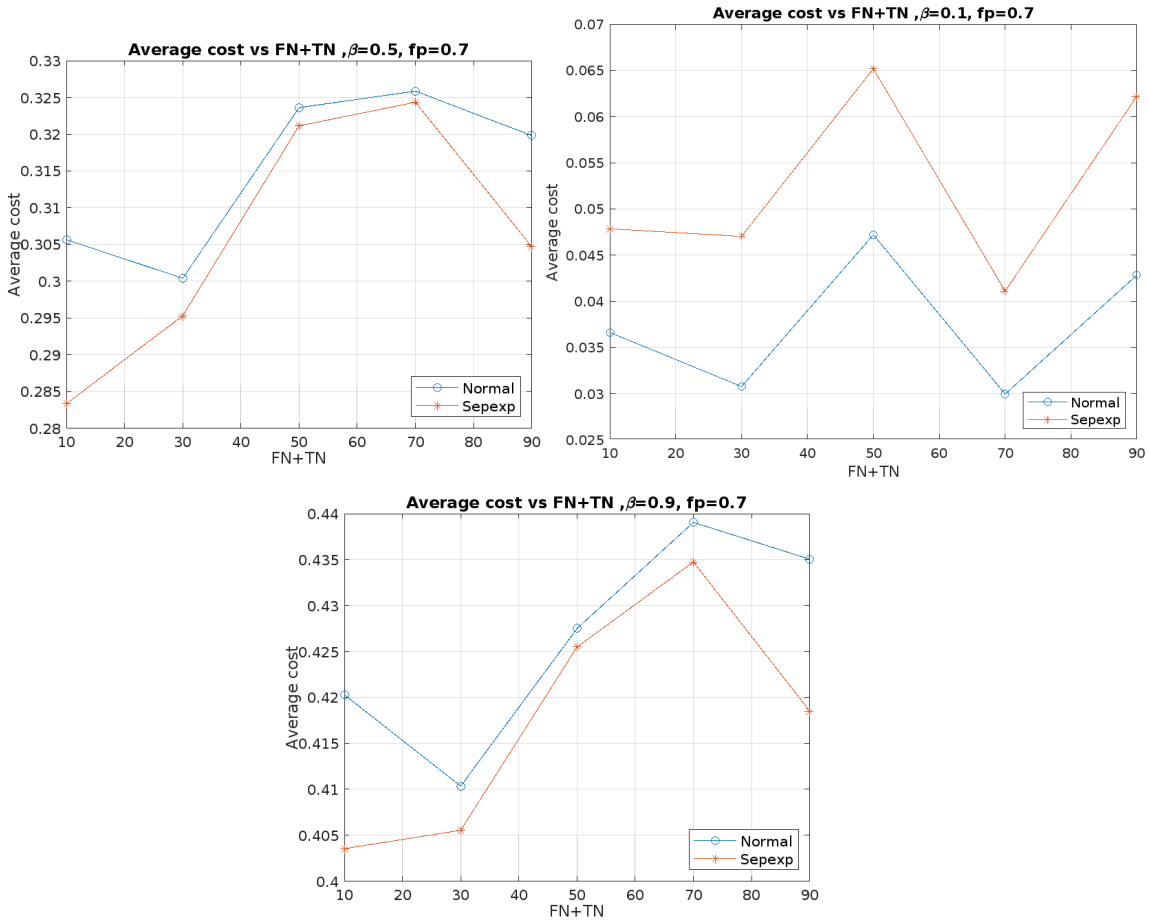

Figure 2: Average cost vs number of $0's$ in ground truth

The graphs have been generated by taking avaerage of costs across different combinations of x, y keeping other parameters as defined. For $\beta = 0$ the separate experts' performace is worse than normal experts for all $f_n + t_n$. But as $\beta = 0.5, 0.9$ increaseses the performace starts improving and is better than normal experts. Another observation is the difference in performance is more as $f_n + t_n$ increases to 1 or $f_n + t_n$ decreases to 0.

4.4 Effect of Offloading cost β

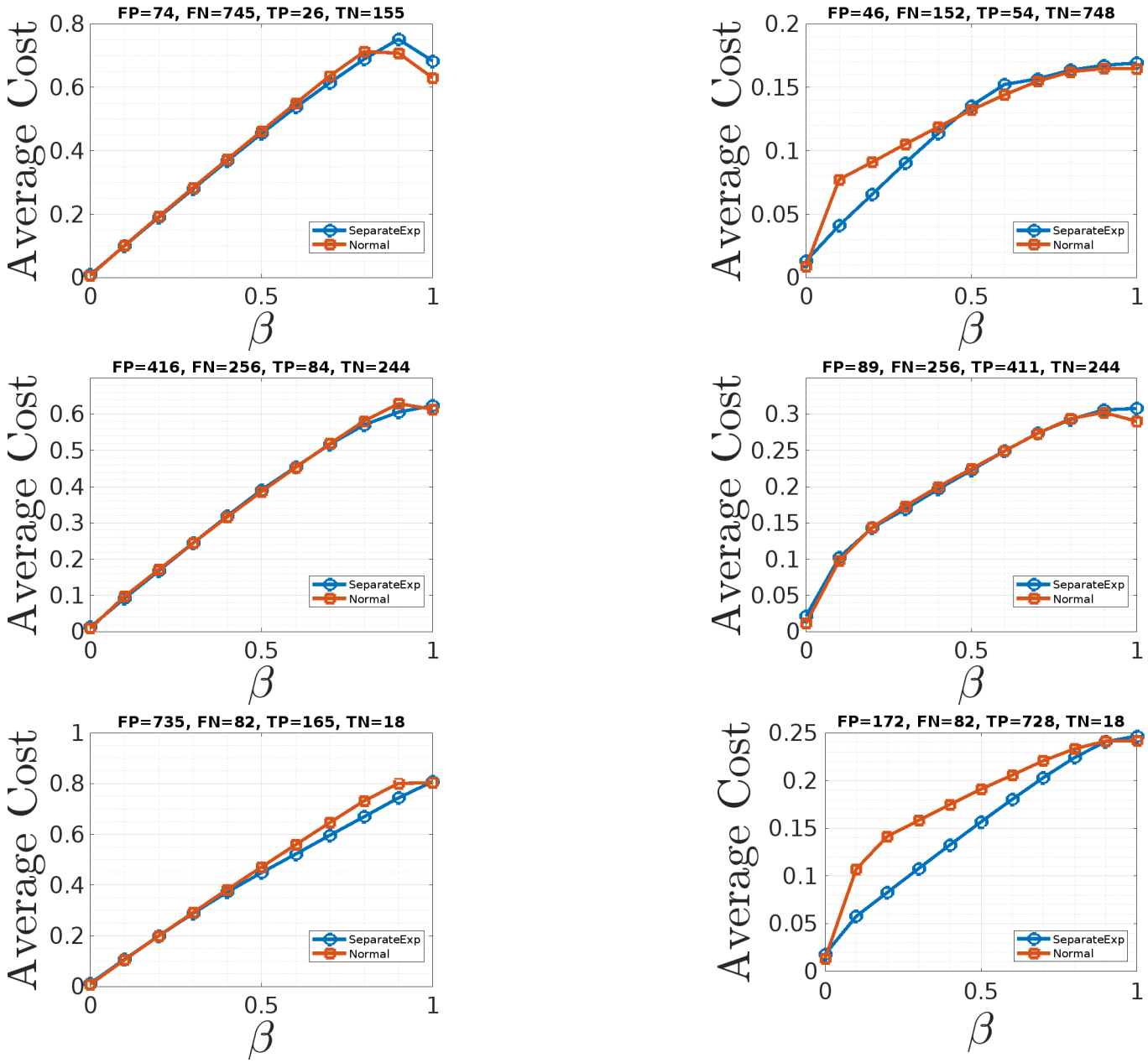

Figure 3: Comparison of Graphs

References

[1] Steven De Rooij, Tim Van Erven, Peter D Grünwald, and Wouter M Koolen. Follow the leader if you can, hedge if you must. *The Journal of Machine Learning Research*, 15(1):1281–1316, 2014.

[2] Tim Erven, Wouter M Koolen, Steven Rooij, and Peter Grünwald. Adaptive hedge. *Advances in Neural Information Processing Systems*, 24, 2011.

[3] Vishnu Narayanan Moothedath, Jaya Prakash Champati, and James Gross. Online algorithms for hierarchical inference in deep learning applications at the edge. *arXiv preprint arXiv:2304.00891*, 2023.