REVAMP^{2}T: Real-time Edge Video Analytics for Multi-camera Privacy-aware Pedestrian Tracking

Christopher Neff*, *Student Member, IEEE,* Matías Mendieta*, *Student Member, IEEE,* Shrey Mohan, *Student Member, IEEE,* Mohammadreza Baharani, *Student Member, IEEE,* Samuel Rogers, *Student Member, IEEE,* Hamed Tabkhi, *Member, IEEE*

***Abstract*—This article presents REVAMP^{2}T, Real-time Edge Video Analytics for Multi-camera Privacy-aware Pedestrian Tracking, as an integrated end-to-end IoT system for privacy-built-in decentralized situational awareness. REVAMP^{2}T presents novel algorithmic and system constructs to push deep learning and video analytics next to IoT devices (i.e. video cameras). On the algorithm side, REVAMP^{2}T proposes a unified integrated computer vision pipeline for detection, re-identification, and tracking across multiple cameras without the need for storing the streaming data. At the same time, it avoids facial recognition, and tracks and re-identifies pedestrians based on their key features at runtime. On the IoT system side, REVAMP^{2}T provides infrastructure to maximize hardware utilization on the edge, orchestrates global communications, and provides system-wide re-identification, without the use of personally identifiable information, for a distributed IoT network. For the results and evaluation, this article also proposes a new metric, Accuracy·Efficiency (Æ), for holistic evaluation of IoT systems for real-time video analytics based on accuracy, performance, and power efficiency. REVAMP^{2}T outperforms current state-of-the-art by as much as thirteen-fold Æ improvement.**

***Index Terms*—Edge Computing, Video Analytics, Deep Learning, Re-identification, Pedestrian Tracking, Privacy.**

I. Introduction

The emerging wave of Internet of Things (IoTs), scarcity of bandwidth resources, and tight latency awareness is pushing system designers to extend cloud computing to the edge of the network. Edge computing (and also fog computing [1]) refers to a group of technologies allowing cooperative computation at the edge of the network [2, 3]. Ambient computer vision and real-time video analytics are the major classes of applications that requires edge computing for human-like vision processing over a large geographic area [3–6].

Recent advances in machine learning, particularly deep learning, have driven the development of more advanced video analytics and surveillance technologies. This includes everything from simple license plate scanners that search for stolen vehicles, to facial recognition and pedestrian tracking.

The authors are with the Electrical and Computer Engineering Department, The University of North Carolina at Charlotte, Charlotte, NC, 28223 USA (e-mail: cneff1@uncc.edu, mmendiet@uncc.edu, smohan7@uncc.edu, mbaharan@uncc.edu, sroger48@uncc.edu, htabkhiv@uncc.edu).

* Corresponding authors have equal contribution.

DOI: https://doi.org/10.1109/JIOT.2019.2954804

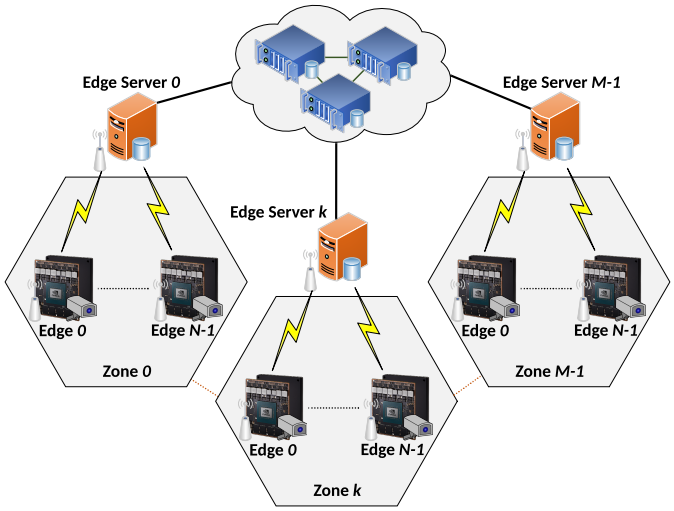

Fig. 1: Hierarchical System Overview

These applications often rely on a cloud computing paradigm for mass video collection and processing on a centralized computing server. The cloud computing paradigm introduces significant technical and social/ethical concerns for such applications. On the technical side, cloud computing leads to mass recording and storage of raw video data, which result in significant costs and limits scalability. At the same time, cloud computing is not applicable to many inherently real-time and time-sensitive video analytics.

On the social perspective, the broad net cast by typical surveillance approaches means that large amounts of personal information are incidentally collected and stored. This has led to significant push-back by privacy advocates against any expansions to video surveillance systems As an example, multiple cities in the US have imposed bans on all deployment of facial recognition and tracking technologies [7]. European Union regulators are also considering new restrictions on AI-driven surveillance [8]. To address both technical and ethical concerns, novel approaches are required to address both IoT systems design and privacy challenges in a holistic manner across entire computing stack from algorithm design to computation mapping, communication, and system-level synchronization.

This paper introduces novel Real-time Edge Video Analytics for Multi-camera Privacy-aware Pedestrian Tracking, or REVAMP^{2}T. REVAMP^{2}T is able to track pedestrians across multiple cameras without ever transferring raw video or other

forms of personally identifiable information. Fig.1 presents our proposed REVAMP^{2}T IoT system. Each IoT device (edge nodes) contains cameras equipped with the NVIDIA AGX Xavier [9] embedded platform, running a deep learning based video analytics pipeline, for real-time pedestrian detection and tracking over streaming pixels. In keeping with the concept of the "right to be forgotten" that was recently enshrined in EU law, our system does not rely on a static identity database. Instead, unique identities are generated when pedestrians first enter the view of a camera in our system and forgotten when those individuals are no longer being actively tracked by any part of our system.

Overall, REVAMP^{2}T achieves a pedestrian re-identification accuracy of 74.8% (only 4.3% below the current state-of-the-art [10]) on the DukeMTMC dataset [11], while achieving more than two times the real-time FPS and consuming 1/5th of the power compared to [10]. A balance was struck between algorithmic Accuracy and system Efficiency, measured by Accuracy•Efficiency (Æ). Our system has high scalability potential in a multi-camera IoT environment while never sacrificing personal privacy.

The rest of this article is organized as follows: Section II briefly reviews the other works related to the various components of our system. Section III describes the privacy threat models. Next, Section IV details the design and implementation of our edge tracking algorithms. In Section V, we describe our edge-to-network infrastructure for enabling multi-camera identity tracking. Evaluation of our complete system is detailed in Section VI. Finally in Section VII we discuss our final conclusions as well as areas for future work.

II. Related Work

A. Pedestrian Detection, Re-Identification, and Tracking

With the rapid advancements made in deep learning, a plethora of work has been published on pedestrian detection. Such models include region proposal networks like Faster-RCNN [12], single shot detectors like SSD [13] and YOLO [14], as well as pose-estimation models like DeeperCut [15] and OpenPose [16]. When analyzing these algorithms in light of edge-capable real-time performance, MobileNet-SSD, TinyYOLOv3, and OpenPose show promising results.

The heart of pedestrian tracking is consistent re-identification (ReID) of those pedestrians throughout the frames of videos across multiple cameras. Similarly, on the re-identification side, recent methods leverage CNNs to extract unique features among persons [17–24]. The work in [25] learns the spatial and temporal behavior of objects by translating the feature map of the Region of Interest (RoI) into an adaptive body-action unit. [26] uses bidirectional Long-Short-Term-Memory (LSTM) neural networks to learn the spatial and temporal behavior of people throughout the video. Triplet loss [27–29] is another promising technique to train the network with the goal of clustering classes in a way that IDs with the same type have minimum distance among each other, while examples from different categories are separated by a large margin.

Pedestrian tracking systems often rely on prediction models to create insight on the changes in movement over time and empowers object re-identification. Object tracking has been tried using spatial masking and Kalman filter techniques for single and multiple object tracking [30–32]. In contrast, there is an interest in leveraging LSTM networks for prediction and tracking. One pronounced example is ROLO [33], which uses YOLOv1 as its feature extractor, combined with LSTMs. Similarly, [34] uses VGG-16 for feature extraction and inputs the 500x1 feature vector into an LSTM. LSTM networks have been shown to provide lower Mean Squared Error in single object and fewer ID switches in multi-object tasks. However, the approaches in [34] and [33] often show very low accuracy as they are not customized for human objects.

Overall, the current state of pedestrian tracking algorithms struggle with limited focus and lack of privacy preservation. First, they look at the problem of pedestrian tracking in isolation, whether solely by detection, only re-identification using image crops, or just tracking with trajectories. However, these approaches do not analyze the problem in a holistic manner, which would require designing a pipeline to understand, integrate, and correlate these three functions into a single intelligent system. Second, the idea of privacy preservation and online functionality are lost with this narrowly focused approach. The previous works typically rely on the storage of large time segments of video data or image crops, degrading privacy preservation. Similarly, many works propose facial recognition techniques [35–39], which also gravely compromises the privacy of tracked persons, requiring the pre-loaded and long-term storage of personally identifiable information like a facial database. At the same time, existing approaches typically analyzes the data offline with the ability to move forward and backward in time to maximize their algorithm accuracy scores, making edge deployable operation of these approaches impractical. In contrast to existing work, this article proposes a shift to non-personal and data private pedestrian tracking, improving upon our previous work in re-identification [40] and LSTM tracking [41] for a holistic algorithm pipeline and fully edge capable design.

B. IoT Systems for Edge Video Analytics

The concept and motivation behind edge computing has been described in a number of recent publications [1, 3–6, 42–45]. However, there are very few works that present a distributed IoT system for video analytics and real-time tracking. [46] proposes a basic system framework for vehicle detection and tracking across multiple cameras. The approach uses positional matching for re-identification, relying on GPS coordinates, known distances, and time synchronization between cameras. The Gabriel project from CMU [47] is a wearable cognitive assistance system where the images captured by a mobile device are processed by the edge node to analyze what the user is seeing, and provide the user with cues as to what is in the scene (for example, recognizing a person). In the VisFlow project from Microsoft, Lu et al. [48] describe a system that can analyze feeds from multiple cameras. In particular, they describe a dataflow platform for vision queries

that is built on top of the SCOPE dataflow engine, that offers general SQL syntax, and supports added user-defined operators such as extractors, processors, reducers, and combiners. [49] proposed a method for single camera multi-target tracking in terms of the Binary Integer Program, and can incur online, real-time results on hardware. However, the system does not scale to multi-camera systems. The current state-of-the-art in Multi-Target Multi-Camera (MTMC) systems is DeepCC [10]. This approach uses OpenPose for detections, a deep learning triplet loss ReID network for visual data association, and trajectory tracklets.

In contrast to Gabriel, REVAMP^{2}T targets machine vision at the edge involving multiple cameras distributed across a geographical area. Unlike VisFlow, where all processing is done at the cloud, REVAMP^{2}T performs a considerable amount of the processing at the edge nodes (next to the camera) by custom-designed deep learning based vision engines, thereby decreasing bandwidth requirements. This also allows REVAMP^{2}T to protect the privacy of the tracked individuals, On the other hand, cloud-based systems must transfer personally identifiable information across large, interconnected networks and store that information in the cloud, where it is all too vulnerable. Additionally, the proposed edge vision system scales easily to a large number of cameras distributed over a wide geographic region.

REVAMP^{2}T accomplishes all these tasks online, in real-time, on low-power edge devices.

III. Privacy Requirements and Threat Modeling

This section describes the privacy threat models which REVAMP^{2}T is designed to address. In safeguarding privacy, we wish to protect the identities and Personally Identifiable Information (PII) of the individuals being viewed by our system. This is most commonly in the form of raw image data, but can also refer to meta-data that can be used to determine the race, gender, nationality, or even identity of an individual. There are three main threats to this that we attempt to address:

- The external threat of someone getting unintended access to network communications and retrieving image data or Personally Identifiable Information (PII).
- The internal threat of someone with authorized access to the system viewing image data or PII - even someone who is supposed to have access to the system should not be able to discern the identities of individuals or have access to their personal information.
- The physical threat of someone getting physical access to the edge device.

To safeguard against these threats, we impose two major policies for designing REVAMP^{2}T:

(1) REVAMP^{2}T will not store any image data or transfer it across the network. As soon as the image is processed on the edge node, it is destroyed. This protects any PII in the images from being viewed by anyone with access to the system. Even with direct access to the edge node, image data never touches non-volatile memory, so accessing it is impossible without fundamentally changing the semantics of our system.

(2) REVAMP^{2}T re-identification algorithm will work on an encoded feature representation of an individual (without using facial recognition algorithms). These features represent the visual and structural attributes of an individual, but can not be interpreted by humans and has zero meaning outside the constraints of our system. This means that even if a person gains access to this feature representation, intended or otherwise, they can not learn anything about the appearance or identity of the individual it was derived from. By utilizing these feature representations, REVAMP^{2}T is able to focus on *differentiation* between people rather than personal *identification*. This is in contrast to common methods that rely on facial recognition or other PII [35–39].

We design REVAMP^{2}T, with respect to defined privacy protection policies. Section IV and Section V present algorithmic constructs and IoT system design of REVAMP^{2}T.

IV. REVAMP^{2}T: Algorithmic Constructs

This section presents algorithmic constructs to enable real-time pedestrian re-identification and tracking while satisfying our privacy model detailed in Section III.

Fig. 2 outlines the full algorithmic pipeline. The pipeline consists of three primary phases: (1) Detection, (2) Re-identification, and (3) Tracking and Prediction. For the detection part, we chose OpenPose [16] from the CMU Perceptual Computing Lab. OpenPose is a pose prediction framework that uses part affinity fields to understand the image input and provide person detections with marked keypoint locations. In addition, it provides keypoints that reveal the motion of the human body, making them useful for motion prediction and action recognition. In the re-identification portion, discriminative features are generated for detection comparison and matching in the Local Database. Once the re-identification has been completed, an LSTM network is applied to predict future positions of known detections. The rest of this section discusses the technical details of our proposed re-identification, tracking, and integration.

A. Feature Extractor Network

The core of the re-identification is the feature extraction network to extract discriminative features from each detection, represented by the Ft-Ext. box in Fig. 2. For this task, a deep convolution network had to be developed for accurate, real-time performance. Most deep convolution networks have a massive number of parameters and operations, which makes them computationally expensive for use in mobile and embedded platforms. MobileNet-V1 [50] and MobileNet-V2 [51] are two developed light-weight deep convolution networks which effectively break down a standard convolution into a depth-wise and point-wise convolution to decrease the network parameters and operations. MobileNet-V2 further improved the MobileNet-V1 architecture by adding linear bottleneck layers and inverted residual connections.

In this article, we use the MobileNet-V2 model and change the fully connected layer to a 2D average pooling with the kernel size of (8, 4) in order to make the output of the network a 1x1280 vector as the embedded appearance

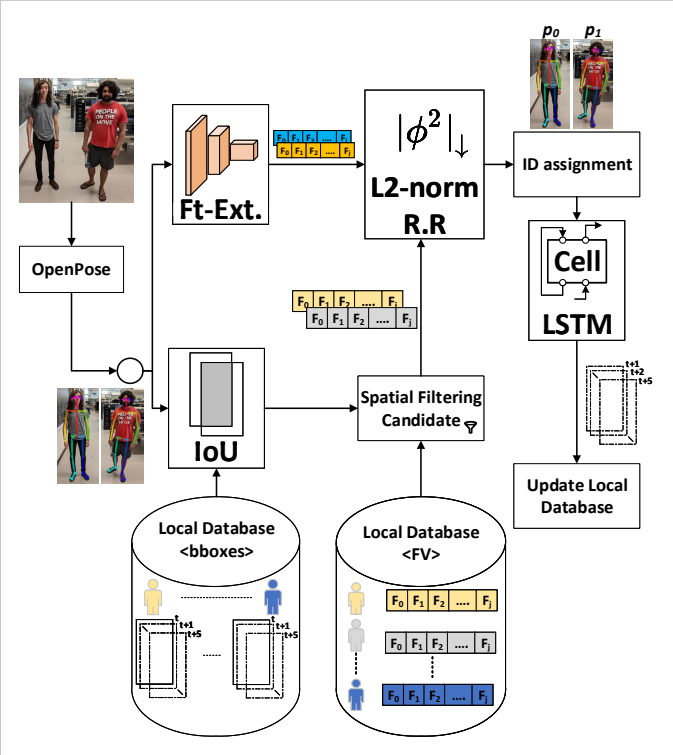

Fig. 2: Algorithm Pipeline on the Edge

features. We also use the triplet loss function [29] to train the MobileNet-V2 for extraction of discriminative features based on person appearance. The underlying architecture of a triplet loss network is consisted of three identical networks which transform the cropped RoI into embedding on a lower dimensional space. One RoI is the anchor image, the second is a positive sample of the anchor and third is a negative sample. The basic concept here is to minimize the distance between the anchor and the positive samples and maximize the distance between the anchor and the negative samples in the lower dimensional embedding space. To facilitate such learning, a suitable loss function is used after the embeddings are extracted from the RoIs:

$$Loss = \sum_{i=1}^{n} \left[\alpha + \|f_i^a - f_i^p\|^2 - \|f_i^a - f_i^n\|^2 \right]_+, \quad (1)$$

where α is margin, f^a, f^p, and f^n are embedded appearance features of the anchor, positive, and negative samples for the class i, respectively. Minimizing $Loss$ function will force all samples of class i to be inside of hypersphere of radius α. The dimension of the hypersphere is equal to the size of the network output (1280 for MobileNet-V2). To further improve the performance of MobileNet-V2 , we have assigned error-friendly operations, such as convolution and General Matrix Multiply (GeMM) operations, to half precision which is 16-bit floating point accuracy and applied mixed precision training [52] to minimize the error caused by half precision operations.

B. Pedestrians Tracking

After the re-identification process, we send the current detections to the LSTM network to get their respective bounding box predictions for the coming five frames. In this way, we can handle miss-detections that the detection network might suffer as it is running at the edge (lower detection network resolution). At the same time, we are able to handle short-term occlusions as we know the position of the occluded pedestrian and can ReID them once they are back in the scene.

To efficiently train our LSTM, we chose the DukeMTMC [11] dataset for training the module as it focuses on pedestrian tracking. The dataset involves multiple targets, and we curate single target instances from the dataset so that we do not involve the re-identification pipeline for the training phase. This results in the network being trained on single instances and carrying out inference on multiple targets. Also, using this technique helps us to re-use the same model parameters for multiple pedestrians when predicting their future positions, saving us redundant computation and making our LSTM tracking module scalable. We leverage the sequence learning capabilities of LSTM by providing it with consecutive frame keypoints of a set sequence length and minimizing the mean-squared error between the obtained predictions and the ground-truth positions of the next five frames.

Fig 3 shows our LSTM module in detail. We provide it with the keypoints of three (our sequence length) consecutive frames and send the last step output to the fully-connected layer which encodes these keypoints to the bounding box position of the pedestrian for the next five frames. The size of our trained LSTM model is under 1.5 MB, making it suitable to run on edge devices.

C. Integration of Video Analytic Pipeline

In order for the entire re-identification task to be accomplished on the edge, these modules must be integrated seamlessly together. Referring back to Fig. 2, a frame is inputted from the camera feed directly into the detection network. The resulting detections are received, scaled to the appropriate size and aspect ratio, and batched through the feature extractor (FT-Ext. box, Fig. 2). The output of this network provides the encoded 1x1280 Feature Vector for each detection. In parallel,

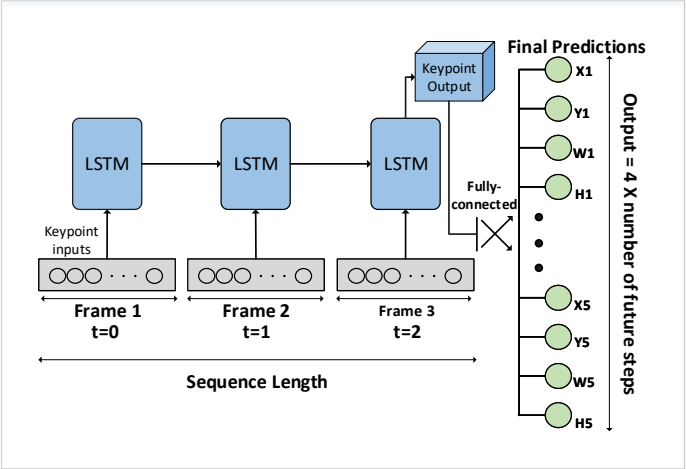

Fig. 3: LSTM training: Feeding a sequence of frames and getting the predicted bounding box predictions

spatial filtering is being done on the Local Database (southern portion of Fig. 2). For each detection, a subset of candidate matches are chosen based on IoU with last known or predicted bounding boxes from previous Local Database entries. The intuition behind this filtering mechanism is to ensure that detections are matched with entries that not only match in the embedded space (Feature Vector), but also in location and trajectory. Because the entire pipeline is running many times per second, the likelihood of a pedestrian traversing a substantial amount of distance or drastically changing trajectory between processed frames is low. Therefore, we avoid including entries that do not make sense from a positional standpoint in the candidate pool for ReID on a new detection.

Within the subset of candidates, the L2-norm operation can be done between Feature Vectors to differentiate between entries, and make final matching decisions using a re-ranking approach to ensure optimal ID assignment (L2-norm R.R box in Fig. 2). As described above, the feature extraction network was trained to maximize the euclidean distance between Feature Vectors of different pedestrians, and minimize the distance between vectors of the same pedestrian. This training and inference methodology provides a privacy-aware approach to ReID, as per our threat models. Rather than using specialized, personally identifiable blocks of information to continually re-identify a pedestrian, our model simply encodes the current visual features of a detection to an abstract representation, and focuses on *differentiation* between entries rather than *personal* identification. Once all detections in the scene are assigned, the LSTM described previously takes in the detection keypoints and generates predicted bounding boxes for the next five time intervals. Finally, the Local Database is updated with assigned labels, keypoints, Feature Vectors, and generated predictions from the processed frame.

V. REVAMP^{2}T: System Constructs

Creating algorithms that can effectively solve issues while running on low-power devices is of vital importance to enable inference on the edge. However, there are many system-level considerations that must be taken into account when developing a robust end-to-end system. How data flows between algorithms, when and how to utilize said algorithms, how to handle communications between the edge node and edge server, and how to map and optimize processes to and for the underlying hardware available on the edge. All of these are system-level design decisions that greatly impact the efficiency and viability of the end-to-end system. With REVAMP^{2}T's focus on privacy, it was important that the IoT system was designed around never storing any personally identifiable information. Algorithmic selections and the design of the system's processing flow hinged around that constraint.

A. System Hyperparameters and Processing Flow

Fig. 4 shows the logical processing flow of one frame of data on the edge, beginning at when the image is extracted from the camera to when the final output is displayed on the edge device. First, the image is run through the keypoint

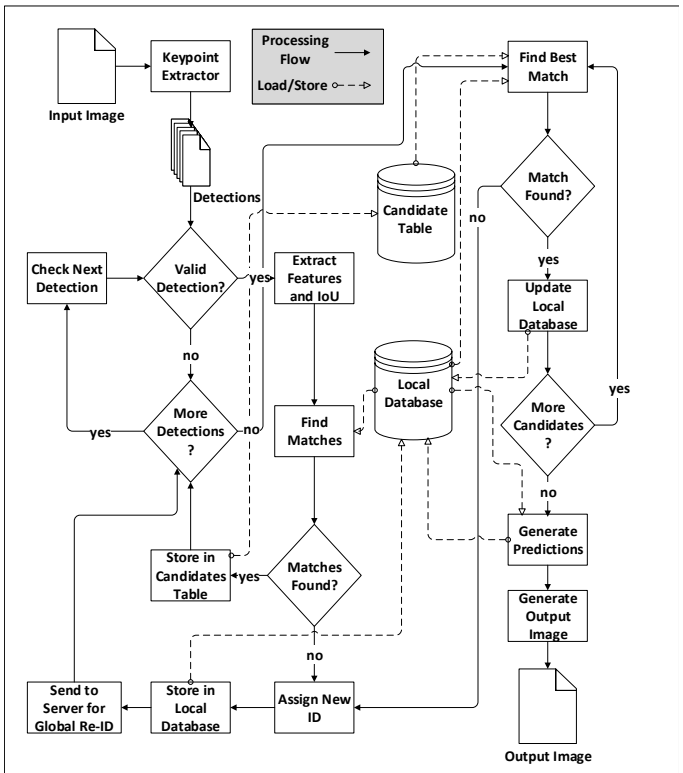

Fig. 4: Processing Flow of the Edge

extractor, which outputs a vector of detections. To remove false detections, each detected pedestrian should have a minimum number of keypoints equal to θ_{key}, and each of those keypoints a confidence value of at least θ_{conf}.

Table I presents the system configurable hyperparameters. For every valid detection, all possible matches for ReID are gathered from the Local Database, as discussed in Section IV-C. When a potential match is found, the detection, database entry index, and the Euclidean Distance between the two are stored in the Candidate Table. If no potential matches were found for a detection, it is considered to be a new person, assigned a new Local ID, stored in the Local Database, and sent to the server for Global ReID.

After all detections have been processed and the Candidate Table filled, the ReID processing is completed and IDs assigned, as shown in Algorithm 2. The lowest Euclidean Distance score in the Candidate Table is found, the detection assigned the ID it was matched to, and the Local Database updated accordingly. Then all entries in the Candidate Table corresponding to that detection and Local Table entry are removed. This process is repeated until there are no suitable

TABLE I: System Parameters

Parameter	Description
θ_{key}	Minimum keypoints for valid detection
θ_{conf}	Confidence threshold for valid keypoint
θ_{euc}	Euclidean threshold with IoU
β_{euc}	Euclidean threshold without IoU
θ_{IoU}	IoU threshold for updating Feature Vector
$\bar{D}$	Detection from keypoint extractor
$\bar{V}$	Valid Detections
$\widehat{DB}$	Local database
$\widehat{C}$	Candidate matrix
ζ	Global variable to keep track of new ID

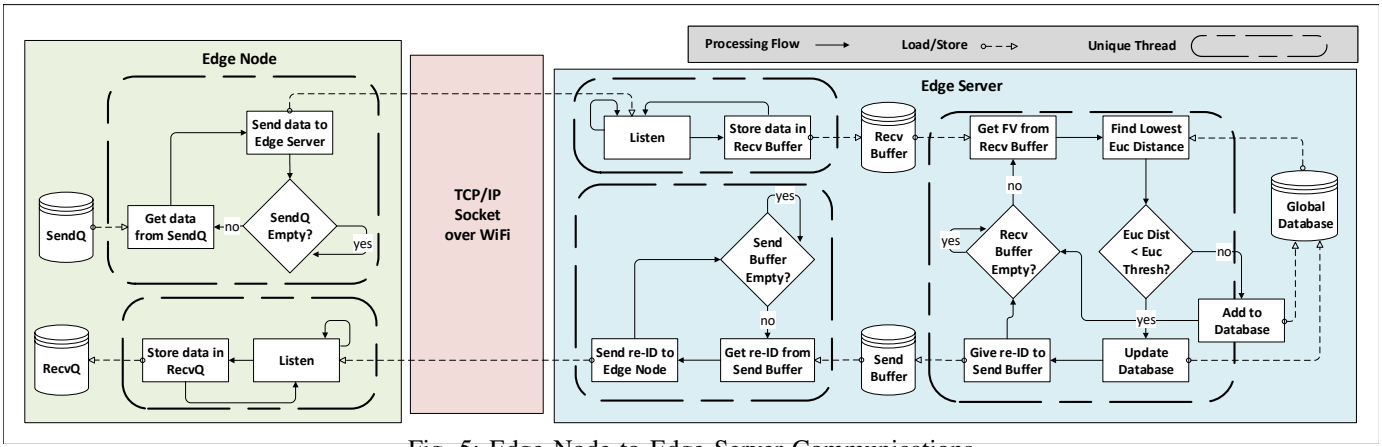

Fig. 5: Edge Node to Edge Server Communications

matches in the Candidate Table, after which all remaining detections are assigned new IDs.

For updating the Local Database on a successful ReID, the system always updates the spatial location of the person (bbox coordinates). However, it only updates the Feature Vector if the IoU score is less than θ_{IoU} and the new Feature Vector is better representative of the object (meaning obtained with more keypoints than previously had). Whenever the Local Database is updated, a message is sent to the server to update its contents accordingly. Once ReID is complete, the system uses the LSTM to generate predictions on all applicable detections, as detailed in Section IV-B and Section IV-C.

B. Databases

On the edge node, a "Local Database" is responsible for storing all pedestrians in the current scene. This database is filled with objects that contain IDs, bounding box coordinates, feature vectors, keypoints, and a parameter called life which keeps track of how many frames it has seen since that pedestrian has been detected. When an object has not been seen by the system after some time (as indicated by life), the object's ID is sent to the edge sever, informing the server of the object's removal from the Local Database. This has two main benefits. Reducing the length of time an object's data is stored on the edge increases the effectiveness of spatial reasoning through IoU, as well as ensuring any single person's data is not stored on the edge when they are not active in the current scene. It also acts as an efficient replacement policy without complex computation.

On the edge server, there is a "Global Database" that functions very similarly to the Local Database. It stores the exact same information, with the addition of knowing which edge node's scene, if any, the object is currently active in. When an object is active in an edge node's scene, that edge node gains ownership of that object, blocking it from being ReID'd by other edge nodes. This ownership is cleared when the server receives notification of the object's removal from the local database of the owning edge node, allowing the object to be included in ReID from all edge nodes. The size of both the Local Databases and the Global Database are easily configurable, allowing for customization to fit the requirements of individual applications.

Algorithm 1 Validating Detections

Input: $\bar{D}, \theta_{conf}, \theta_{key}$
Output: $\bar{V}$
$\bar{V} \leftarrow \emptyset$
for d in $\bar{D}$ **do**
  $numKeyPoints$ = findValidKeyPoints(d, θ_{conf})
  if $numKeyPoints \geqslant \theta_{key}$ **then**
    $\bar{V} \leftarrow \bar{V} \cup \{d\}$
  end if
end for

Algorithm 2 Finding Best Matches

Input: $\widehat{C}$
Output: $\widehat{newID}, \widehat{reID}$
$\widehat{newID} \leftarrow \Phi, \widehat{reID} \leftarrow \Phi$
$\tilde{C}$ = sortBasedOnL2Norm($\widehat{C}$)
for (v, e, ϕ) in $\tilde{C}$ **do**
  if $e \neq null$ **then**
    $\widehat{reID} \leftarrow \widehat{reID} \cup \{e.\text{id}\}$
    removeValidEntryAndCandidate(v,e, $\tilde{C}$)
  end if
end for
while thereIsCandidate($\tilde{C}$) **do**
  $\zeta = \zeta + 1$
  $\widehat{newID} \leftarrow \widehat{newID} \cup \{\zeta\}$
  removeValidEntryAndCandidate(v,$\emptyset$, $\tilde{C}$)
end while

C. System Communication / Synchronization

A vital aspect of REVAMP^{2}T's communication design hinges around exactly what data is sent to the server, according to our privacy threat models described in Section III. The only data transmitted across the network is encoded Feature Vectors, impersonal IDs, and system metadata. The current iteration of REVAMP^{2}T's communication protocol leverages Wifi, but is adaptable and can be expanded to other communication protocols, such as LTE and 5G. This allows REVAMP^{2}T to keep up with the ever-changing communications landscape, no matter what technologies emerge.

Fig. 5 shows communication synchronization between edge

nodes and edge server. Communications between the edge node and edge server are handled asynchronously. Both the edge node and edge server have separate threads to handle sending and receiving ReID information and storing it in separate buffers. These buffers hold the data until the main threads are ready to work on it. In addition to the main thread and the Global Database, the server has a separate set of these buffers and threads for each node. The main thread of the server processes this data in a round robin fashion and based on the metadata it receives from the edge node. It will either update the Global Database with a new Feature Vector for a global ID, release ownership of a global ID, or check the Global Database for ReID matches for the provided Feature Vector. Communications are only sent back to the edge node when a ReID match is successfully found.

By handling all communications on separate threads unrelated to inference, communications are entirely decoupled from the processing pipeline, eliminating pipeline stalls that would normally result from inline communications. This decoupling also means that edge node throughput is not dependent on network latency; Local ReID will always perform at a constant FPS. Additionally, if the network goes down and communications are completely lost, the buffers will allow a level of data synchronization after communications are restored. The system efficiently utilizes edge resources, which has a monumental impact on inference time. On the server side, the proposed system achieves greater scalability, as edge nodes do not fight over available sockets and communications does not take away from ReID resources.

D. Computation and Optimization

To achieve real-time performance on the edge, we chose Nvidia AGX Xavier SoCs [9]. The Xavier is equipped with many advanced components that are leveraged for REVAMP^{2}T, including eight ARM Core processors, two Nvidia Deep Learning Accelerators (NVDLA), and a Volta GPU with Tensor Cores optimized for FP16 Multiply and Accumulate.

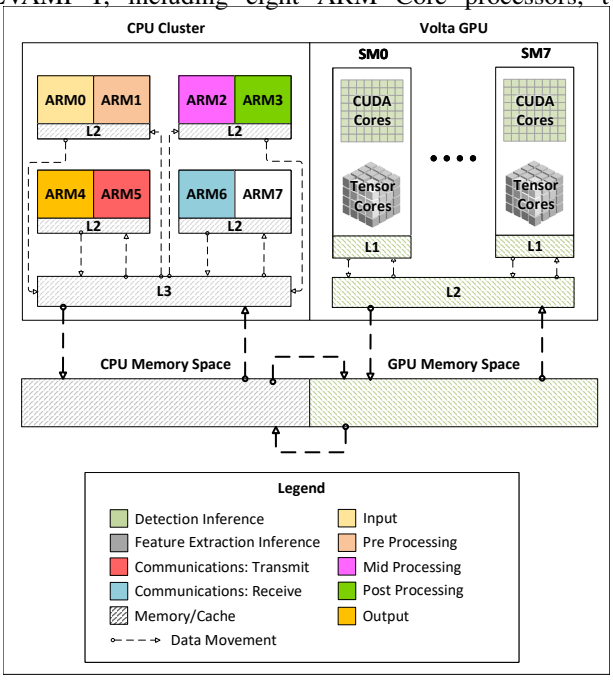

Fig. 6: Mapping of Processes to Edge Resources

Fig. 6 shows the how the different processes in REVAMP^{2}T are mapped to the Xavier resources. Each stage of the detection framework is mapped to a separate ARM Core. The transmit and receive threads are mapped to their own cores as well. This leaves one ARM Core free to handle the OS and any background processes running outside of the system. Detection inference runs on the CUDA Cores of the Volta GPU. ReID inference is run on Tensor Cores. To enable this, the ReID network model was is converted from ONNX to use half precision through TensorRT. Batch normalization layers are also fused into the convolutional layers, reducing data migration. Detections are batched for ReID inference each frame, allowing a ReID throughput above 20 FPS. The NVDLAs were not used for ReID due to a lack of support for the level of group convolution in MobileNet-V2. All code on the edge was developed in C++ for computational efficiency, enhanced execution, and mapping control.

VI. Experimental Results and Evaluation

The experimental setups and results will be split into four subsections: Algorithm, System, Scalability, and Design Flexibility. All project code for simulations and full system implementation is provided on GitHub[1].

A. Algorithm Evaluation

1) Feature Extractor Network: We used DukeMTMC-reID [11, 53], CUHK03 [54], and Market1501 [55] for evaluating the performance of two networks with different training methods. Table II summarizes the hyperparameters of our network. We decreased learning rate exponentially after 150 epochs and used Adam optimizer to train both networks.

TABLE II: Training Parameters

Description	Value	Description	Value
Batch size	128	Input shape (H×W)	(256×128)
IDs per batch	32	Margin	0.3
Instances per ID	4	Epoch	300
Initial learning rate	2×10^{-4}		

We used the baseline ResNet-50 as used in DeepCC for the sake of performance evaluation of MobileNet-V2. We applied Cumulative Match Characteristic (CMC) [56, 57] as a metric to evaluate and compare the identification performance of the two networks. Each dataset consists of a gallery $\mathcal{G}$ as a set of various person images, and a query $\mathcal{Q}$ as a set of various person images that we want to identify. $\mathcal{P}_\mathcal{G}$ is a *probe* set, a subset of $\mathcal{Q}$, and for each of its images there are matches in $\mathcal{G}$. As the gallery embeddings are extracted, they are ranked (sorted) based on the similarity ($L2-Norm$ distance) across the current query image features. Then a set of the matched cases at rank r can be defined as in [57]:

$$C(r) = \{p_j \mid rank(p_j) \leqslant r\} \; \forall p_j \in \mathcal{P}_\mathcal{G} \tag{2}$$

[1]https://github.com/TeCSAR-UNCC/Edge-Video-Analytic

Based on Eq. 2 CMC at rank r is calculated by following equation:

$$CMC(r) = \frac{|C(r)|}{|\mathcal{P}_{\mathcal{G}}|} \tag{3}$$

It should be noted that the CMC calculation can still be different for each dataset. For example, in Market-1501, $\mathcal{Q}$ and $\mathcal{G}$ can share the same camera view. However, for each individual query image, the individual's samples in $\mathcal{G}$ from the same camera are excluded [55].

Another evaluation metric which gives a representation of network performance over a set of queries $\mathcal{Q}$ is mean Average Precision (mAP), which can be extracted by:

$$mAP = \frac{\sum_{i=1}^{|\mathcal{Q}|} AP(q_i)}{|\mathcal{Q}|}, q_i \in \mathcal{Q} \tag{4}$$

$$AP(q) = \frac{1}{TP_{gt}} \sum_{j}^{|\mathcal{G}|} \frac{TP_{detected}}{j} \tag{5}$$

where TP_{gt} is the number of ground-truth true positives, and $TP_{detected}$ is the number of true positives detected by the network. CMC(1), CMC(5), and mAP are computed and compared side-by-side in Fig. 7 for both ResNet-50 and MobileNet-V2 half precision networks. We can realize that the MobileNet-V2 half precision is 6.1% less than ResNet-50 for mean value of CMC(1) across all three datasets, while reaching an 18.92× model size compression ratio for MobileNet-V2 (5.0MB) over the baseline model (94.6MB).

2) LSTM Prediction Network: A total of 120 single object sequences, 15 from each camera, were used to train the LSTM model for 200 epochs. For testing the model we used 3 sequences per camera. The network takes around 14 hours to train on an Nvidia V100 GPU and was implemented using PyTorch. To evaluate the performance of the network we use the Intersection over Union (IoU) of the predicted bounding boxes with the ground truth bounding boxes and average it for all frames in the sequence, as shown in Fig. 8. This average IoU shows that we maintain performance above the 0.3 IoU detection threshold typically used for evaluation. We do not compare these results with DeepCC because their approach uses tracklets rather than IoU for tracking evaluation.

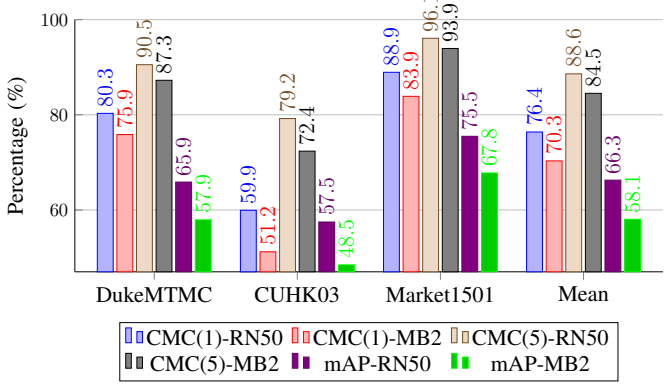

Fig. 7: ResNet-50 single precision and MobileNetV2 half precision accuracy evaluation on three different benchmarks

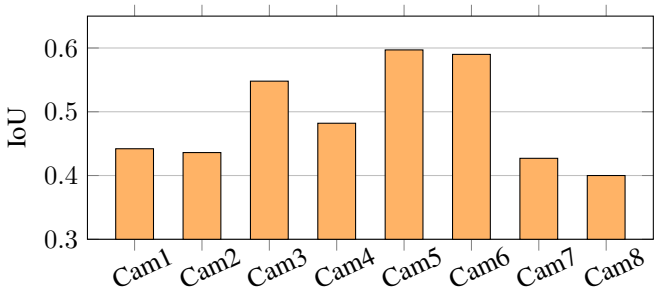

Fig. 8: Average IoU for each camera on the testing sequences

3) Algorithm Pipeline: In order to validate the accuracy of the full algorithm pipeline, the edge algorithms and edge server functionality were ported to MATLAB and compiled into a simulation testbed to gather results. For these experiments, we used the DukeMTMC dataset, which includes 85 minutes of 1080p footage from 8 different cameras on the Duke University campus. Specifically, the trainval_mini frame set was used for validation. For comparison, we also ran the current state-of-the-art in MTMC work, DeepCC, on the same trainval_mini validation set. For all experiments, we measure ID Precision (IDP), ID Recall (IDR), and ID F1 score (IDF1) with truth-to-result matching, as proposed in [58]. Intuitively, IDP measures the percentage of attempted ReIDs that were correct, and IDR measures the percentage of possible ReIDs completed, regardless of number of attempts. IDF1 is simply the harmonic mean of IDP and IDR. Detection misses are computed in accordance to the truth-to-matching method, with the IoU threshold at 0.3. In accordance with Table I, the values for system hyperparameters are as follows: θ_{key} = 5, θ_{conf} = 0.5, θ_{euc} = 5, $\theta_{euc\beta}$ = 2, θ_{IoU} = 0.3.

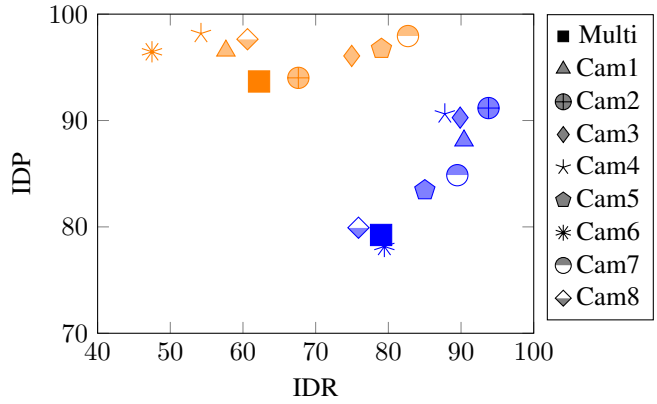

Fig. 9: Precision (IDP) and Recall (IDR) for DeepCC (blue) and REVAMP^{2}T (orange)

Fig. 9 shows IDP versus IDR for REVAMP^{2}T and DeepCC. Analyzing the results, DeepCC maintains groupings around 80% for both IDP and IDR. REVAMP^{2}T maintains high IDP, always above 90%; however, the IDR is less consistent across cameras. The reasons for this problem are two fold. First, because REVAMP^{2}T is an online system, it was designed to ReID within a short temporal window, in accordance with the spontaneous nature of online operation. Second, many of our false negatives are simply the result of missed detections. For the full 8-camera (shown as Multi) scenario, 59% of the

false negatives incurred were from missed detections from the first stage of the pipeline. As mentioned in Section IV, we chose to run the detection network at a relatively low resolution at an attempt to balance reasonable runtime speed and detection accuracy. Nonetheless, despite the challenges of edge-capable algorithmic development, REVAMP^{2}T maintains reasonably close IDF1 in comparison to DeepCC. Fig. 10 shows the IDF1 for each camera individually, as well as the complete 8-camera global system (Multi) for both approaches. Overall, REVAMP^{2}T only drops 4.3% IDF1 in the full multi-camera system compared to the offline DeepCC algorithm, with DeepCC at 79.1% and REVAMP^{2}T at 74.8%.

B. System Evaluation

For all measurements, REVAMP^{2}T is run in real-time. We also compare against DeepCC [10]. For both, 16 detections per frame is assumed. As DeepCC was not built as a real-time system, it would be unfair to include the latency incurred through gallery matching in these comparisons, so we ignore the effect of this on power and latency. Real-time candidate matching is built into REVAMP^{2}T, so it is included in all reported measurements. For measuring the power consumption on the Xavier, the Tegrastats was used.

1) Power Consumption and Computation Efficiency: For power consumption on the Titan V and V100 GPUs, we utilized the NVIDIA System Management Interface. AMD μProf was used to measure CPU idle power for the edge server. For REVAMP^{2}T, 1080p 30 FPS video was pulled directly from a webcam. For DeepCC, 1080p 60 FPS video was read from memory. In both cases, a brief warm up of 20 frames was allowed before power was sampled over 100 frames. Measurements for FPS were taken directly from the OpenPose GUI.

TABLE III: FPS and Power Consump. of Real-Time Inference

System	REVAMP^{2}T	DeepCC	DeepCC	DeepCC
Device	Xavier	Titan V	2xTitan V	V100
FPS ↑	5.7	2.5	4.7	2.7
Power ↓	34.4W	200W	365W	224W
Detailed Xavier Power Consumption				
CPU	GPU	DDR	SOC	Total
4.2W	22.1W	2.75W	5.35W	34.4W

Table III presents the power consumption and FPS for REVAMP^{2}T and DeepCC. Here we can see that for real-time applications, REVAMP^{2}T out performs DeepCC on each GPU setup we tested. Even using two Titan V's, DeepCC is only able to reach 4.7 FPS. Meanwhile, REVAMP^{2}T can reach 5.7 FPS. In addition, REVAMP^{2}T consumes only 17% of the power of DeepCC on a single Titan V, or 9% for the dual Titan setup. Fig. 11 presents computation efficiency, which is FPS processing per watt. DeepCC has an Efficiency between 0.0147 and 0.0161 FPS/Watt in all configurations. In comparison, REVAMP^{2}T has an Efficiency of 0.166 FPS/Watt. When looking at Efficiency, REVAMP^{2}T performs an order of magnitude better than DeepCC for real-time applications. This is because REVAMP^{2}T was built from the ground up to perform in real-time, both algorithmically and systemically.

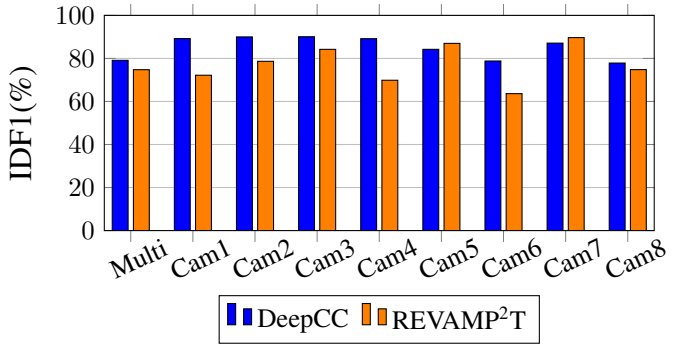

Fig. 10: IDF1 Results for Multi-Camera and Single Camera

2) Accuracy • Efficiency (Æ): To enable real-time AI applications on the edge, we propose a new metric with which to measure edge performance; that is Accuracy • Efficiency (Æ). With Æ, we combine the algorithmic measurement of Accuracy with the systemic measurement of Efficiency to measure how well an application will perform in a real-time edge environment. Æ has two parts: an Æ mark, which is a score measured by the product of Accuracy and Efficiency, and Æ coverage, which is measured in area, as determined by all the components of an Æ mark. The components in Æ coverage, when not already reported as a percentage, are normalize to be so. In the case of power, this normalized value is subtracted from one, as lower power consumption is preferable.

In the case of REVAMP^{2}T, Accuracy would take the form of F1, while Efficiency is measured in FPS/Watt. Fig. 12 shows the Æ Mark for REVAMP^{2}T and DeepCC, while Æ Coverage can be seen in Fig. 13. Here you can see that while DeepCC outperforms REVAMP^{2}T in terms of IDR and IDF1 Accuracy, REVAMP^{2}T has a significantly higher Æ Mark (12.39 vs 1.02) and almost double the total Æ Coverage (81.25% vs 42.75%). This is because our optimizations allow us to operate at twice the framerate, 17% of the power, and we only lose by 4.3% in F1 accuracy.

C. Scalability

To measure scalability, we compare REVAMP^{2}T, which pushes all local processing to the edge nodes, against two other configurations: (1) "Server Processing" which streams the edge video frames to the edge server to handle all the local processing, and (2) "Split Processing" which splits local processing between the edge node and the edge server.

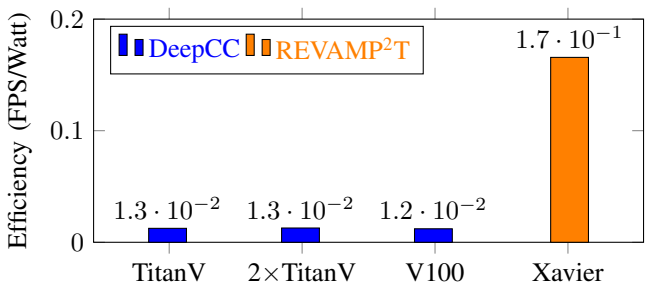

Fig. 11: Efficiency of each test case.

TABLE IV: Scalability Evaluation Results

	Server Processing					Split Processing				Edge Processing			
Nodes	GPUs	Cost	Power (W)	End-to-end Latency (ms)	Network Latency (ms)	Cost	Power (W)	End-to-end Latency (ms)	Network Latency (ms)	Cost	Power (W)	End-to-end Latency (ms)	Network Latency (ms)
1	1	$8,300	201	**83.2**	21	$8,900	122	506.6	29	**$5,700**	**34**	540.6	17
2	1	$8,400	202	**105.0**	35	$9,600	203	571.6	53	**$6,400**	**69**	549.6	26
4	2	$11,800	359	**168.8**	51	$11,000	306	678.6	82	**$7,800**	**138**	559.7	36
8	3	$15,400	518	**223.8**	107	$13,800	455	941.6	175	**$10,600**	**275**	601.9	78
16	6	$25,000	946	**374.8**	258	$19,400	742	1547.1	414	**$16,200**	**550**	700.3	175
32	11	$48,400	1827	972.8	856	$30,600	1301	2723.2	858	**$27,400**	**1101**	**866.0**	338
64	22	$91,800	3609	1202.8	1086	$53,000	2406	6020.9	2074	**$49,800**	**2202**	**1095.7**	559

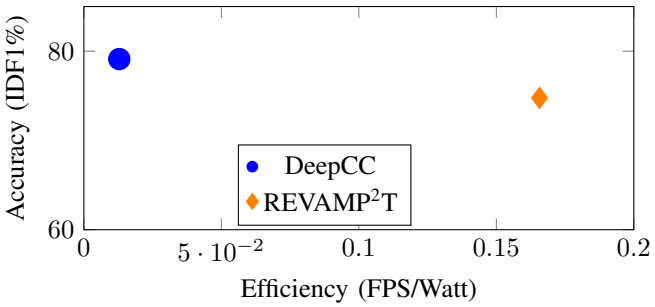

Fig. 12: Æ of DeepCC on Titan V and REVAMP^{2}T on Xavier.

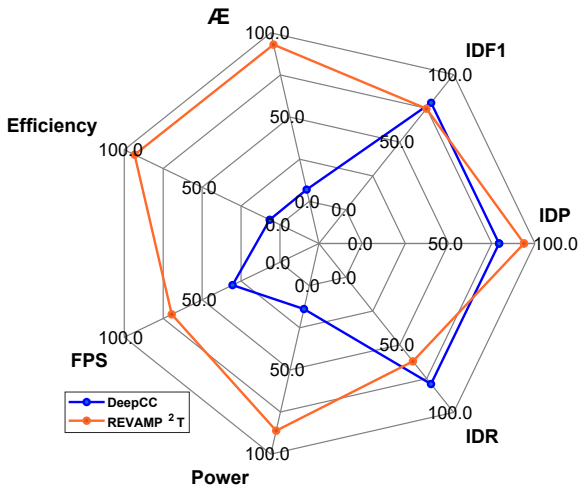

Fig. 13: Æ Coverage

Table IV lists the number of GPUs, cost, power, end-to-end latency, and network latency over increasing number of edge nodes for all three scenarios. Power was measured using the same methods as described previously. The latency for a single frame was measured from when it was grabbed from the camera/video, to right before being displayed (using chrono library in C++), averaged over 100 frames, after a 20 frame warm up. For Server Processing, the edge nodes are Nvidia Jetson Nano SoCs [59], as they only stream video to the server. For the other cases, the Nvidia Xavier was used. In all cases we assume an edge server with a 12 core CPU, at least 32 GB of memory, and the capacity to support up to eight Nvidia Titan-V GPUs. The server processes all data at 30 FPS. We assume video data from separate nodes can be interwoven to allow a single instance of REVAMP^{2}T to support two edge nodes. Network latency was simulated using NS-3 Discrete Network Simulator, using 802.11ac, TCP, and 600Gbps throughput. H.264 compression for video, PNG for images, and 16 detections per frame are assumed.

From Table IV we are able to see that Server Processing is able to achieve the lowest latency for smaller node counts. Past 16 edge nodes, the network latency from streaming video tips the scales latency to favor Edge Processing. However, Edge Processing always wins out in terms of cost and power consumption, making it a promising option particularly for high node counts. At 64 edge nodes, Edge Processing has the best latency while only requiring about half the cost and 60% the power of Server Processing. We expect this trend to continue on to even higher node counts, increasingly favoring Edge Processing. Split Processing fares the worst in the comparison, due mostly to network latency. PNG compression is not nearly as efficient as H.264, meaning far more data is sent across the network, leading to the large latencies seen in the table. Overall, this shows that even for computationally intensive applications, edge processing is the only truly scalable solution for real-time IoT applications.

TABLE V: Design Configuration Analysis

Config.	Power(W) ↓	FPS ↑	Accuracy ↑
P_2	**12.48**	3	71.37%
P_3	15.51	4	73.67%
C_D	34.40	5	74.80%
R_{720}	36.47	3	**74.91%**
R_{256}	30.09	15	73.77%
R_{128}	26.01	**24**	0.97%

D. Design Flexibility and Adaptation

REVAMP^{2}T can further be configured to prioritize accuracy, FPS, or power consumption, as illustrated Table V. The default configuration of REVAMP^{2}T is shown as C_D, with an input resolution of 496x368 for the keypoint extractor, and power consumption as seen in Table III. We analyze five additional design configurations by modifying the input resolution, as well as the power restriction levels on the Xavier device. Configuration R_{720}, R_{256}, and R_{128} are the proposed REVAMP^{2}T with modified input resolutions at 720x544, 256x192, and 128x96. Configuration P_2 and P_3 are the proposed REVAMP^{2}T configured with Power Mode 2 and 3 (C_D uses Power Mode 0) provided by the Xavier device [60].

While R_{720} does provide slightly higher accuracy than other configurations, it does incurs loss in FPS and increased power consumption. The keypoint extraction resolution for R_{128} cannot properly extract keypoints for persons further than ∼30 feet from the camera, resulting in low accuracy for DukeMTMC. R_{256} offers an option for additional throughput at an accuracy loss. While R_{256} performs well on the tested dataset, we found that in real-world testing, this configuration was only able to extract keypoints for persons within ∼50 feet from the camera. Therefore, for robustness to real-world

situations and its balance across all areas, configuration C_D was chosen for the analysis of this report. With deployment in an IoT environment, C_D would likely require PoE Type 3. The P_3 and P_2 configurations show how REVAMP^{2}T could be adapted to the power levels of PoE Type 2 and Type 1 for deployment, with minimal loss in accuracy.

VII. Conclusions

This article proposed REVAMP^{2}T as an integrated end-to-end IoT system to enable decentralized edge cognitive intelligence for situational awareness. For the results and evaluation, this article also proposed a new two-part metric, Accuracy•Efficiency (Æ). REVAMP^{2}T outperforms current state-of-the-art by as much as a thirteen-fold improvement in Æ. Future directions include designing light-weight human feature extractors, as a replacement for OpenPose, further improvement of edge performance by designing application-specific hardware on FPGAs, data encryption to secure communications, and containerizing REVAMP^{2}T using edge Kubernetes for scalable system orchestration and remote programmability across the edge devices.

Acknowledgement

This research is supported by the National Science Foundation (NSF) under Awards No. 1737586 and 1831795.

References

[1] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, *Fog Computing: A Platform for Internet of Things and Analytics*. Cham: Springer International Publishing, 2014, pp. 169–186. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-05029-4_7

[2] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, Oct 2016.

[3] M. Sapienza, E. Guardo, M. Cavallo, G. L. Torre, G. Leombruno, and O. Tomarchio, "Solving critical events through mobile edge computing: An approach for smart cities," in *2016 IEEE International Conference on Smart Computing (SMARTCOMP)*, May 2016, pp. 1–5.

[4] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, and F. Giust, "Mobile-edge computing architecture: The role of mec in the internet of things," *IEEE Consumer Electronics Magazine*, vol. 5, no. 4, pp. 84–91, Oct 2016.

[5] M. Chiang and T. Zhang, "Fog and iot: An overview of research opportunities," *IEEE Internet of Things Journal*, vol. PP, no. 99, pp. 1–1, 2016.

[6] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, Oct 2009.

[7] "https://www.nytimes.com/2019/05/14/us/facial-recognition-ban-san-francisco.html."

[8] "https://www.wsj.com/articles/ai-surveillance-tools-scrutinized-by-european-regulators-11561562155."

[9] M. Ditty, A. Karandikar, and D. Reed, "Nvidia xavier soc," Aug 2018.

[10] E. Ristani and C. Tomasi, "Features for multi-target multi-camera tracking and re-identification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6036–6046.

[11] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," in *European Conference on Computer Vision workshop on Benchmarking Multi-Target Tracking*, 2016.

[12] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015. [Online]. Available: http://arxiv.org/abs/1506.01497

[13] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," *CoRR*, vol. abs/1512.02325, 2015. [Online]. Available: http://arxiv.org/abs/1512.02325

[14] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv*, 2018.

[15] L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. V. Gehler, and B. Schiele, "Deepcut: Joint subset partition and labeling for multi person pose estimation," *CoRR*, vol. abs/1511.06645, 2015. [Online]. Available: http://arxiv.org/abs/1511.06645

[16] Z. Cao, T. Simon, S. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," *CoRR*, vol. abs/1611.08050, 2016. [Online]. Available: http://arxiv.org/abs/1611.08050

[17] X. Zhang, H. Luo, X. Fan, W. Xiang, Y. Sun, Q. Xiao, W. Jiang, C. Zhang, and J. Sun, "Alignedreid: Surpassing human-level performance in person re-identification," *arXiv preprint arXiv:1711.08184*, 2017.

[18] E. Ristani and C. Tomasi, "Features for multi-target multi-camera tracking and re-identification," in *Conference on Computer Vision and Pattern Recognition*, 2018.

[19] K. Zhou, Y. Yang, A. Cavallaro, and T. Xiang, "Omni-scale feature learning for person re-identification," *arXiv preprint arXiv:1905.00953*, 2019.

[20] Y. Shen, H. Li, S. Yi, D. Chen, and X. Wang, "Person re-identification with deep similarity-guided graph neural network," in *The European Conference on Computer Vision (ECCV)*, September 2018.

[21] S. Li, S. Bak, P. Carr, and X. Wang, "Diversity regularized spatiotemporal attention for video-based person re-identification," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 369–378.

[22] M. Li, X. Zhu, and S. Gong, "Unsupervised tracklet person re-identification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2019.

[23] X. Zhu, X. Jing, X. You, X. Zhang, and T. Zhang, "Video-based person re-identification by simultaneously learning intra-video and inter-video distance metrics," *IEEE Transactions on Image Processing*, vol. 27, no. 11, pp. 5683–5695, 2018.

[24] T. Xiao, S. Li, B. Wang, L. Lin, and X. Wang, "Joint detection and identification feature learning for person search," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 3376–3385.

[25] W. Zhang, B. Ma, K. Liu, and R. Huang, "Video-based pedestrian re-identification by adaptive spatio-temporal appearance model," *IEEE Transactions on Image Processing*, vol. 26, no. 4, pp. 2042–2054, 2017.

[26] J. Dai, P. Zhang, D. Wang, H. Lu, and H. Wang, "Video person re-identification by temporal residual learning," *IEEE Transactions on Image Processing*, vol. 28, no. 3, pp. 1366–1377, 2019.

[27] Y. Zhang, Q. Zhong, L. Ma, D. Xie, and S. Pu, "Learning incremental triplet margin for person re-identification," *CoRR*, vol. abs/1812.06576, 2018. [Online]. Available: http://arxiv.org/abs/1812.06576

[28] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *J. Mach. Learn. Res.*, vol. 10, pp. 207–244, Jun. 2009. [Online]. Available: http://dl.acm.org/citation.cfm?id=1577069.1577078

[29] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," *CoRR*, vol. abs/1703.07737, 2017.

[30] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 5, pp. 564–575, 2003.

[31] S.-K. Weng, C.-M. Kuo, and S.-K. Tu, "Video object tracking using adaptive kalman filter," *Journal of Visual Communication and Image Representation*, vol. 17, no. 6, pp. 1190–1208, 2006.

[32] X. Li, K. Wang, W. Wang, and Y. Li, "A multiple object tracking method using kalman filter," in *The 2010 IEEE international conference on information and automation*. IEEE, 2010, pp. 1862–1866.

[33] G. Ning, Z. Zhang, C. Huang, Z. He, X. Ren, and H. Wang, "Spatially supervised recurrent convolutional neural networks for visual object tracking," *CoRR*, vol. abs/1607.05781, 2016. [Online]. Available: http://arxiv.org/abs/1607.05781

[34] A. Sadeghian, A. Alahi, and S. Savarese, "Tracking the untrackable: Learning to track multiple cues with long-term dependencies," *CoRR*, vol. abs/1701.01909, 2017. [Online]. Available: http://arxiv.org/abs/1701.01909

[35] K. Koide, E. Menegatti, M. Carraro, M. Munaro, and J. Miura, "People tracking and re-identification by face recognition for rgb-d camera networks," in *2017 European Conference on Mobile Robots (ECMR)*, Sep. 2017, pp. 1–7.

[36] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[37] H. S. Dadi, G. K. M. Pillutla, and M. L. Makkena, "Face recognition and human tracking using gmm, hog and svm in surveillance videos," *Annals of Data Science*, vol. 5, no. 2, pp. 157–179, Jun 2018. [Online]. Available: https://doi.org/10.1007/s40745-017-0123-2

[38] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 499–515.

[39] "Surveillance solutions." [Online]. Available: https://www.sighthound.com/solutions/surveillance

[40] M. Baharani, S. Mohan, and H. Tabkhi, "Real-time person re-identification at the edge: A mixed precision approach," in *Lecture Notes in Computer Science*. Springer International Publishing, 2019.

[41] P. Kulkarni, S. Mohan, S. Rogers, and H. Tabkhi, "Key-track: A lightweight scalable lstm-based pedestrian tracker for surveillance systems," in *Lecture Notes in Computer Science*. Springer International Publishing, 2019.

[42] E. A. Lee, B. Hartmann, J. Kubiatowicz, T. S. Rosing, J. Wawrzynek, D. Wessel, J. Rabaey, K. Pister, A. Sangiovanni-Vincentelli, S. A. Seshia, D. Blaauw, P. Dutta, K. Fu, C. Guestrin, B. Taskar, R. Jafari, D. Jones, V. Kumar, R. Mangharam, G. J. Pappas, R. M. Murray, and A. Rowe, "The swarm at the edge of the cloud," *IEEE Design Test*, vol. 31, no. 3, pp. 8–20, June 2014.

[43] O. Vermesan, P. Friess, P. Guillemin, and S. Gusmeroli, *Internet of Things Strategic Research Agenda*. River Publishers, 2011.

[44] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, Oct 2016.

[45] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, ser. MCC '12. New York, NY, USA: ACM, 2012, pp. 13–16. [Online]. Available: http://doi.acm.org/10.1145/2342509.2342513

[46] J. Španhel, V. Bartl, R. Juránek, and A. Herout, "Vehicle re-identification and multi-camera tracking in challenging city-scale environment," in *Proc. CVPR Workshops*, 2019.

[47] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, "Towards wearable cognitive assistance," in *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '14. New York, NY, USA: ACM, 2014, pp. 68–81. [Online]. Available: http://doi.acm.org/10.1145/2594368.2594383

[48] Y. Lu, A. Chowdhery, and S. Kandula, "Visflow: A relational platform for efficient large-scale video analytics," Tech. Rep., June 2016. [Online]. Available: https://www.microsoft.com/en-us/research/publication/visflow-relational-platform-efficient-large-scale-video-analytics/

[49] E. Ristani and C. Tomasi, "Tracking multiple people online and in real time," in *Asian Conference on Computer Vision*. Springer, 2014, pp. 444–459.

[50] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017.

[51] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. G. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 2018, pp. 2704–2713.

[52] P. Micikevicius, S. Narang, J. Alben, G. F. Diamos, E. Elsen, D. García, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu, "Mixed precision training," *CoRR*, vol. abs/1710.03740, 2017. [Online]. Available: http://arxiv.org/abs/1710.03740

[53] Z. Zheng, L. Zheng, and Y. Yang, "Unlabeled samples generated by gan improve the person re-identification baseline in vitro," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[54] W. Li, R. Zhao, T. Xiao, and X. Wang, "Deepreid: Deep filter pairing neural network for person re-identification," in *CVPR*, 2014.

[55] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian, "Scalable person re-identification: A benchmark," in *Computer Vision, IEEE International Conference on*, 2015.

[56] H. Moon and P. J. Phillips, "Computational and performance aspects of pca-based face-recognition algorithms," *Perception*, vol. 30, no. 3, pp. 303–321, 2001.

[57] P. Grother, R. J. Micheals, and P. J. Phillips, "Face recognition vendor test 2002 performance metrics," in *International Conference on Audio- and Video-based Biometric Person Authentication*. Springer, 2003, pp. 937–945.

[58] E. Ristani, F. Solera, R. S. Zou, R. Cucchiara, and C. Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," *CoRR*, vol. abs/1609.01775, 2016. [Online]. Available: http://arxiv.org/abs/1609.01775

[59] "Nvidia jetson nano system-on-module data sheet [preliminary]," NVIDIA Corporation, May 2019, rev. 0.7.

[60] Dusty-Nv, "Jetson agx xavier new era autonomous machines," May 2019. [Online]. Available: https://github.com/dusty-nv/jetson-presentations/blob/master/20181004_Jetson_AGX_Xavier_New_Era_Autonomous_Machines.pdf