IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 8, 2020, accepted July 14, 2020, date of publication July 20, 2020, date of current version August 5, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3010185

IRON-MAN: An Approach to Perform Temporal Motionless Analysis of Video Using CNN in MPSoC

SOMDIP DEY[1], (Graduate Student Member, IEEE), AMIT KUMAR SINGH[1], (Member, IEEE), DILIP KUMAR PRASAD[2], (Senior Member, IEEE), AND KLAUS DIETER MCDONALD-MAIER[1], (Senior Member, IEEE)

[1]Embedded and Intelligent Systems Laboratory, University of Essex, Colchester CO4 3SQ, U.K.
[2]Department of Computer Science, UiT The Arctic University of Norway, 9019 Tromsø, Norway

Corresponding author: Somdip Dey (somdip.dey@essex.ac.uk)

This work was supported in part by the U.K. Engineering and Physical Sciences Research Council (EPSRC) under Grant EP/R02572X/1 and Grant EP/P017487/1 and in part by Nosh Technologies under Grant nosh/agri-tech-000001.

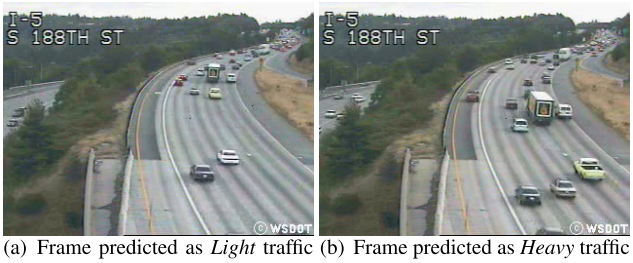
ABSTRACT This paper proposes a novel human-inspired methodology called IRON-MAN (***I****ntegrated* ***R****ati****ON****al prediction and* ***M****otionless* ***AN****alysis*) for mobile multi-processor systems-on-chips (MPSoCs). The methodology integrates analysis of the previous image frames of the video to represent the analysis of the current frame in order to perform Temporal Motionless Analysis of the Video (*TMAV*). This is the first work on *TMAV* using Convolutional Neural Network (CNN) for scene prediction in MPSoCs. Experimental results show that our methodology outperforms state-of-the-art. We also introduce a metric named, Energy Consumption per Training Image (*ECTI*) to assess the suitability of using a CNN model in mobile MPSoCs with a focus on energy consumption and lifespan reliability of the device.

INDEX TERMS Convolutional neural network (CNN), temporal analysis, motionless analysis, video, lifespan, reliability, energy efficiency, embedded device, multiprocessor systems-on-chip (MPSoCs).

I. INTRODUCTION AND MOTIVATION

Recently there has been a huge increase in utilizing Convolutional Neural Networks (CNNs) [1]–[3] to solve several real-life challenges such as human rights violation [4], traffic categorization [5]–[7], pedestrian detection [8]–[10], weather forecasting [11], [12], etc due to its high prediction accuracy/categorization in the aforementioned target applications. One particular case for harnessing the efficacy of visual CNN based prediction model is Intelligent Transportation Systems (ITS), which is becoming an important pillar in the modern ''smart city'' framework.

Given the increase in vehicles on road one of the key challenges in ITS is accurate traffic load categorization. In recent times, there has also been emergence of several methods capable of monitoring and analyzing traffic using motionless analysis of videos [5]–[7], [14], where videos of traffic are broken into frames instead, and the frames are analyzed for further computation or prediction. The main motivation to utilize methodologies consisting of motionless analysis of video is that it is difficult to stream high-frame rate videos gathered by a large network of interconnected cameras due to bandwidth limitation. Hence, streaming low-frame rate videos on these camera networks is very common. In many cases, it is challenging to stream more than 2 frames per second due to the limited bandwidth of the network when these cameras stream over a WIFI network [5], [6]. Moreover, to analyze



(a) Frame predicted as *Light* traffic category (b) Frame predicted as *Heavy* traffic category

FIGURE 1. Frames (Images) from the same *Light* traffic category of UCSD dataset [13] and associated prediction by a trained CNN model [6].

The associate editor coordinating the review of this manuscript and approving it for publication was Mehul S. Raval.

video in real-time without motion features over a WIFI network is difficult due to communication bandwidth constraint and hence, it is better to analyze the image fames on the camera enabled embedded device itself [6] instead of relying on a server system over the WIFI network. Another motivation to develop such approaches in embedded devices is the affordability of such devices, e.g. in developing countries (where an Odroid XU4 [15] mobile platform with a camera sensor costs only 59 USD), instead of employing powerful server systems used for analysis purposes [6]. Comparatively, a powerful server with GPU compute capability (Nvidia Tesla GPU) to perform traditional prediction with CNNs costs more than 8,500 USD [16] and also requires multiple installations of such servers based on Geo-location. Therefore, the approach of analyzing videos without motion features on the embedded device is not just beneficial for categorizing traffic load but could be extended to several computer vision based real-world applications that require analysis of low-frame rate videos. Although motionless analysis of videos has its own benefits, it also come with limitations described with the following observations.

Observation 1: Although several implementations of such methods were able to achieve high prediction accuracy on known dataset [5], [6], [14], in some test cases the prediction/analysis was not accurate at all. The reason for poor prediction/analysis is that in some cases it is difficult to predict the label of an image frame from a video if the ground truth[1] of the image is overlapping with several other categories (labels). For example, in the dataset of traffic released by UCSD [5], [13], which consists of light, heavy and traffic jam categories, two frames (images) belonging to the same category of video are predicted differently by the CNN model [6]. The reason for such behavior is that the CNN predicts the label and probability of it occurring on the instantaneous image frame. In Fig. 1, we notice that a trained CNN model [6] with an overall prediction accuracy of 81.25% predicted the wrong label for a frame, which falls under *Light* category, but was instead predicted as *Heavy*. However, both the frames belong to the same video under the *Light* category. If the ground truth of the two images (Fig. 1 (*a*) & (*b*)) are compared then it is justifiable that the prediction by the CNN is in fact accurate due to the fact that the traffic projected in Fig. 1.(*b*) is more congested than the traffic projected in 1.(*a*). In reality the analysis of each image frame of the video should also portray the overall analysis[2] for the video instead of the image frame itself in order to convey the temporal prediction. Since, each individual image frame of a video could lead to different analysis result (prediction/label), the temporal prediction is the prediction analysis of the video over time. This limitation is due to the fact that the trained CNN model only predicts the label or analyzes the current image frame without taking past image frames into consideration. Therefore, the challenge is to analyze and predict videos just from the image frames without motion features of the video and yet give accurate temporal prediction results for the video as well.

There have been some recent studies, which focused on future predictions of motion in ego-centric videos[3] [17], [18] or action[4] [19], [20] taken by a human being, such as predicting the future position of a person based on the current image frame. However, no study to our best knowledge has tried to predict the scene[5] [21], [22] of a video from image frames taking predictions from immediate previous frames into account to provide a more holistic analysis of the scene over a time period. Hence, we call such an analysis as ***T**emporal **M**otionless **A**nalysis of **V**ideo (TMAV)*. Several target applications of CNN such as traffic categorization require such kind of analysis in comparison with traditional ones [5], [6], [14].

Observation 2: In the study [6], Dey *et al.* proposed a methodology to implement a CNN, which is trained on a configurable embedded device, and it was utilized to categorize traffic on the same device. Although this study gave a novel approach on analyzing traffic using video cameras in low bandwidth network without the need to communicate the image frames over the WIFI network, the study had energy consumption and device lifespan reliability issues (shown later in the section). Due to wide consumer adoption of mobile devices utilizing multi-processor system-on-a-chip (MPSoC) [23]–[31], which implements several different types of processing elements (PEs) such as CPU/GPU on the platform, MPSoCs are perfect candidates for implementing computing resource demanding algorithms such as CNN based methodologies.

When the CNN model, proposed in the study [6], is implemented and trained on a MPSoC such as Odroid XU4 [15], the maximum temperature of the CPU reached 93.72°C on an average and the power consumption peaked at 10.63 Watt on an average during the training period. Reaching a high operating temperature for a long period of time is an important factor in the reduction of lifespan of the device. In some studies [32]–[35], it has been found that an increase in the operating temperature by 10-15° centigrades could reduce the lifespan of the device by 2×. Additionally, an increase in energy consumption on embedded devices is harmful for battery operation itself [36], [37], especially in low-power embedded devices. Therefore, it is important to design an energy-efficient CNN model, which is able to achieve desired analysis of video without motion features, but at the same time does not affect the lifespan reliability of the system adversely.

[1]Ground truth of the image frame in this case is the information gained through empirical evidence as opposed to the inference made by the CNN model.

[2]Here, overall analysis of video means the analysis of the video as a whole as opposed to the analysis of each image frame of the video.

[3]Ego-centric video is a first-person vision technique, which acts as an artificial visual system that perceives the world around camera wearers and assist them to decide their next action.

[4]Action is based on the movement of the human being in consideration in the image frame.

[5]Scene is a place where a human being could navigate or can act within.

In order to overcome the limitations of the existing approaches we propose **IRON-MAN**: **I**ntegrated **R**ati**ON**al prediction and **M**otionless **AN**alysis of videos using CNN, which is capable of performing *TMAV*, in MPSoCs. To this end, this paper is an extended version of [7] and makes the following contributions:

1) An energy efficient scene prediction methodology (*IRON-MAN*) based on CNN, which integrates predictions of previous image frames of a video to predict the current frame, and hence, analyze the video without using motion features.
2) A new metric named *Energy Consumption per Training Image* (*ECTI*), which will enable the choice of suitable CNN model for real-world applications on embedded devices keeping energy-efficiency in mind.
3) Validation of the proposed approach on a real hardware platform, the Odroid-XU4 [15].
4) Effect on lifespan of the device utilizing CNN based approaches on such platforms.
5) Comparative study of IRON-MAN with existing state-of-the-art approaches.
6) A framework to deploy IRON-MAN in large-scale over the cloud.
7) Discussion and limitations of IRON-MAN.
8) Proposal of an alternative prediction module in IRON-MAN to overcome its limitations.

II. PAPER ORGANIZATION

The rest of the paper is organized as follows: in Sec. III, we introduce some of the concepts and relevant technologies that are utilized in this work; in Sec. IV, we discuss the state-of-the-art methodologies related to traffic categorization and pedestrian detection; in Sec. V, we introduce our proposed methodology, IRON-MAN, to perform TMAV, and also introduced the ECTI metric; in Sec. VI, we show evaluation results from the experiments along with comparative study with the sate-of-the-art; in Sec. VII, we propose the framework to deploy IRON-MAN in large-scale over the cloud and also show experimental evaluation in the cloud; in Sec. VIII, we discuss the limitations of the IRON-MAN; in Sec. IX, we propose an alternative approach for the IRON-MAN to overcome its limitations; finally, we conclude the paper in Sec. X.

III. PRELIMINARIES

A. CONVOLUTIONAL NEURAL NETWORKS AND DEEP LEARNING

A Deep Learning (DL) model [38] consists of an input layer, several intermediate (hidden) layers stacked on top of each other and an output layer. In the input layer, which is the first layer of the model, the raw values of data features are fed into it. In each of the hidden layers a mathematical operation called convolution is applied to extract specific features, which is then utilized to predict the label of the raw data in the last (output) layer of the DL network. Most of the time, if a model utilize an input layer, a hidden layer and an output layer then the model is denoted as Convolutional Neural Network (CNN) model or simply, CovNet. If such a model uses a lot of stacked hidden layers only then it is denoted as a DL model or Deep Neural Networks (DNN).

B. PRE-TRAINED NETWORKS AND TRANSFER LEARNING

A conventional approach to enable training of DNN/CNN on relative small datasets is to use a model pre-trained on a very large dataset, and then use the CNN as an initialization for the applicative task of interest. Such a method of training is called "*transfer learning*" [39] and we have followed the same principle.. The chosen CNN models mentioned in Sec. V-A are pre-trained on ImageNet.

C. POWER CONSUMPTION, THERMAL BEHAVIOUR AND RELIABILITY ISSUES IN MPSoCs

Energy efficient (reduced power consumption) execution of applications on multi-processor systems such as MPSoCs is desired in order to improve the operation time of battery-powered systems. This requires development of efficient run-time management (RTM) approaches and/or designing the executing application in an energy-efficient approach. Due to computational complexity of CNN models it is very important to design CNN model based applications, which are energy efficient on MPSoCs. On the other hand, on systems utilizing MPSoCs if proper energy consumption control measures are not taken then it could lead to heat generation in the system. The availability of multiple PEs on the system in comparison with uniprocessors can lead to more non-uniformity of heat generation/dissipation capable of reducing performance and lifespan reliability of the system over the period of time [40]. Therefore, optimizing thermal behaviour of the MPSoC is very important for such devices.

IV. RELATED WORK

Before 2015, majority of traffic analysis and categorization was mostly performed using the following methodologies:

- Vehicle based methodologies, where either vehicles are first localized on the road with a background subtraction method [41]–[43] or the vehicles are localized with moving feature keypoints [44]–[46]. In these methodologies, the resulting tracks are concatenated together to identify key features of traffic such as traffic lanes, average traffic speed, average traffic density, etc.
- A holistic approach, where a macroscopic analysis of traffic flow is understood through global representation of a scene, which is obtained by accounting for spatio-temporal features except tracking using background subtraction and moving feature keypoints [13], [47], [48].

Although the aforementioned methodologies are highly effective to analyze traffic, the biggest limiting factor is the cost of sophisticated camera-network involved and the requirement for high-frame-rate videos to compute reliable motion features. To break away from this trend of

traffic analysis, in 2015 Luo *et al.* [5] proposed a methodology to use various image processing and CNN based approaches to analyze traffic without moving features. In [5], the authors used four different visual descriptors such as bag of visual words (BOVW), Vector of Locally Aggregated Descriptors (VLAD), improved Fisher Vector (IFV) and Locality-constrained Linear Coding (LLC), and have also used pre-trained deep CNN models such as Caffe and VGG to analyze traffic and predict categorization of the same. Although the theory proposed by Luo to use popular image processing and CNN methods to classify traffic is novel and solves the low-frame-rate video streaming issue, but the methodology was not implemented on embedded device for local training/inference. In [6], Dey *et al.* proposed an improved traffic categorization based on CNN in embedded devices utilizing System-on-a-programmable-chip (SoPC), however, this work does not consider holistic prediction of the video (no TMAV). In another extended paper published by Luo *et al.* [14], the researchers have used SegCNN and RegCNN to analyze and classify traffic. In this aforementioned paper as well, the authors are training and classifying traffic images after the video frames are transferred to the server from the interconnected camera network and does not perform TMAV.

Other state-of-the-art methodologies include detecting & counting the numbers of cars and computing traffic density based on that using CNN-based vehicle detectors with high accuracy at near real time [49]–[51]. Although this way of detecting traffic density could still be classified as a vehicle based approach and has become popular in recent times but there are associated challenges with these methods as follows:

- Training and test data should belong to the same dataset taken from the same camera with same configuration and hence require consistency in training.
- Cars detected need to be within a particular range or scope of the image and these methodologies fail to detect cars, which are far away in the images captured.
- These methodologies performed poorly if the captured images were occluded, especially in case of heavy traffic & jam.

From the aforementioned list of issues with the state-of-the-art methods, although Deep Learning [52] could solve the problem of detecting occluded objects properly but such method usually requires large dataset to be trained with. Since, our methodology is applied to an embedded/mobile device, connected to its own camera sensor, collecting local image frame (data) to continuously train the model on the device for prediction is not an issue.

On the other hand, another key challenge in Intelligent Transport Systems is pedestrian detection. Using CNNs to detect pedestrians is not a new topic [53]. In [8], Dollár *et al.* proposed a benchmark for pedestrian detection with improved evaluation metrics. In [8], the benchmark dataset was developed to solve the following: scalability, occlusion and positioning of pedestrians in the images. In [10], Burton *et al.* performed pedestrian detection by locating an object resembling the class of a person using CNN from a distance of 100m, with a lateral accuracy of ±20 cm. In [9], Flohr and Gavrila proposed another dataset for pedestrian detection as benchmark, where the pedestrians are observed from a traffic vehicle by using on-board stereo cameras. The same work and dataset is extended by Li *et al.* [54], which includes cyclist video samples captured with the same setup as [9]. We utilize IRON-MAN to predict pedestrian obstruction detection to evaluate the suitability and scalability of our proposed methodology for this type of target application as well. More details on experimental results and evaluation is provided in Sec. VI.

V. PROPOSED METHODOLOGY: IRON-MAN

In our proposed approach, we utilize the concept of *Hybrid Training Method* [6], where we train our model both during offline (training period) and online (runtime/post-training period) modes. IRON-MAN (**I**ntegrated **R**ati**ON**al prediction and **M**otionless **AN**alysis of videos) has two modules in it: *Training* and *Prediction* (as shown in Fig. 2). The strength of our approach is that it provides temporal analysis of videos without motion features i.e. *TMAV*.

In the *training module*, we use *transfer learning*[6] [39], [55] by utilizing an existing pre-trained network and training the classifier with our data categories. First, we train the pre-trained CNN with our dataset, which could be either performed on the MPSoC or on a powerful computing system, which has a lot more computing resources than the MPSoC that could be leveraged to improve the training time. After the initial phase of training is complete, we evaluate the overall prediction accuracy of the trained CNN. If the overall prediction accuracy (P_i) of the CNN is equal or more than the desired quality of experience (Q) [6], then we utilize the CNN for prediction in the prediction module. Here, the desired quality of experience (Q) is the minimum (desired) validation prediction accuracy that the model has to achieve after the completion of the training. However, if the desired prediction accuracy is not achieved then we *retrain* (details in Sec. V-A) the CNN with the failed predicted images. This *retrain* methodology is human-inspired as it mimics one of the key intelligence feature of a human being, which is learning from the surrounding environment to adapt. When a human being meets a new environment and is not aware of the rules and regulations associated with it, the human tries to adjust and adapt by learning the new set of rules and regulations. We have utilized the same concept in our approach as well, which is described subsequently in Sec. V-A.

After the *retraining* of the CNN, when the desired prediction accuracy is achieved we use the trained CNN in the *prediction* module. Now, instead of providing prediction result for each individual image frame, we integrate the final prediction by taking previous image frames into consider-

[6]Learning achieved by taking the convolutional base of a pre-trained network, running the new data of 4 traffic categories through it and training a new randomly initialized classifier

ation. Our CNN model's prediction is inspired by *Bayes' theorem* and *sequential Bayesian updating* [56], where the model updates the probability of the occurring prediction label by incorporating the probability of the label occurring in the previous frames. This approach is again human-inspired as it is adopted from the ideology of humans updating their knowledge using Bayesian inference logic. Detailed algorithm and inner working (*Training* and *Prediction* modules) of the IRON-MAN is provided in the following two subsections.

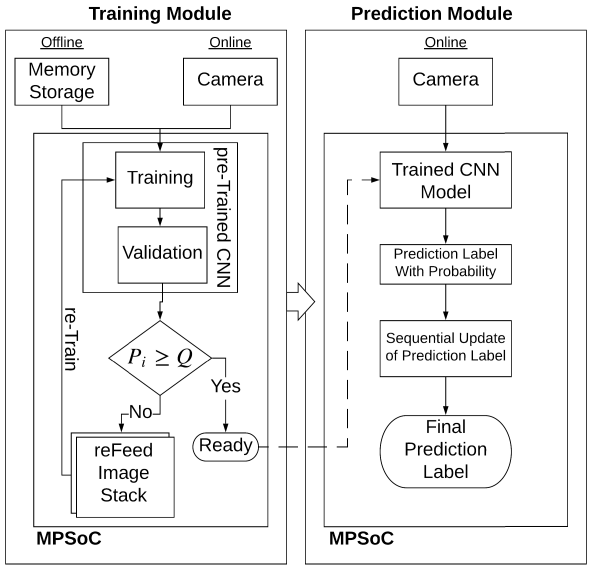


FIGURE 2. IRON-MAN Model Work-flow.

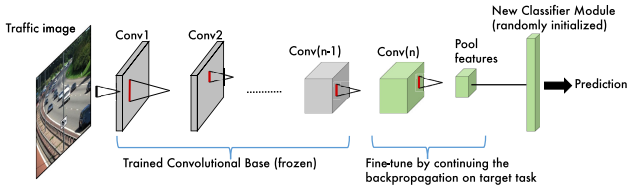


FIGURE 3. Network architecture used for fine-tuning pre-trained CNN for traffic categorization.

A. TRAINING MODULE

For the proposed approach, any pre-trained CNN model such as VGG [3], ResNet [57], MobileNet [58], etc could be selected. It is very common to choose a pre-trained CNN model and fine-tune the model to train on a target application. Fine-tuning is the process of taking the weights of a pre-trained CNN and using it as initialization for a new model being trained on a dataset from the same domain. This approach is used to speed up the training process while being able to train on small dataset. We fine-tuned our pre-trained CNN model by adding our a new randomly initialized classifier, and training the last fully connected layer by freezing all the layers of the base model (frozen layers represented with gray colour in Fig. 3 with respect to traffic categorization prediction) and unfreezing the last fully connected layer (unfrozen layers represented with green colour in Fig. 3 with respect to traffic categorization prediction). If the target application changes such as the case for pedestrian detection, only the randomly initialized classifier changes to fine-tune for the current target application.

The training module itself consist of two part training: Offline and Online mode. During the offline mode, the CNN is trained with stock images from a dataset stored on a memory. After the initial training period (offline) the CNN is then fed with live images from the camera and the prediction for each image frame is evaluated during the validation of the model. Upon failure in prediction for each image during validation testing, the image is stored in a stack implementation called "*reFeed Image Stack*". After utilizing cross-validation technique [59], where validation testing is performed on a separate dataset such as live image frames from the camera stream, the overall prediction accuracy (P_i) of the CNN model is evaluated. If the overall prediction accuracy is equal or more than the desired quality of experience (Q) then the training process concludes and the CNN is ready to start predicting categories in the *prediction* module. The governing equation to check for the suitability of the CNN for further prediction is provided in Eq. 1 [6], where I is the dataset consisting of images, i is an image in the dataset and P_i is the prediction accuracy of the CNN for i. However, if the desired quality of experience is not met in terms of prediction accuracy then the CNN is trained with the failed prediction images, which are stored in the *reFeed image stack*. We call this as the *reTrain* approach so that the CNN can achieve a higher *localized prediction accuracy*. Here, the term *localized prediction accuracy* means the prediction accuracy of the model for a set of images that is restricted to a specific task or place. In this application, it should be kept in mind that *reTrain* does not mean training the whole model from the beginning, however, it means to continue the training process with the images from the reFeed image stack in order to improve learning capability of the CNN model. The retrain mechanism is bound to improve prediction accuracy because we train the CNN model's classifier and the last fully connected layer (as mentioned earlier in the fine-tuning mechanism) with the failed images saved in the reFeed image stack and this approach mimics a human being's ability to rectify his/her mistake after making one. Here, the failed images are the image frames which were predicted/labeled incorrectly during the testing of the trained CNN model.

$$\forall\{i \in I : i > 1\}, \quad P_i \geq Q \tag{1}$$

Note: Using the proposed approach, training could be performed both on the MPSoC or on a more powerful computing device. If the CNN is trained on a device other than the MPSoC, then after the aforementioned training phases (offline and online) are complete, the CNN model's parameters and weights could be saved and migrated to the MPSoC to act as the *Prediction* module. Utilizing this method of

training the CNN can also improve reliability (lifespan) and energy efficiency of the device due to the required operating resources during the training period being high. Here, the operating resources represent the computing resources such as CPU and/or GPU, memory, etc. required during the operation of an executing application.

B. PREDICTION MODULE

From Bayes' Theorem [56], if we assume the *posterior*, *prior* and *likelihood* to be *po*, *pr* and *li* respectively then we can represent the expression representing the *Bayesian Update Scheme* as follows:

$$po \propto pr \times li \tag{2}$$

In Eq. 2, *posterior* (*po*) is the revised probability of an event occurring after taking new information into consideration, *prior* (*pr*) is the probability of the event assessed before revising posterior and *likelihood* (*li*) is the probability that an event which has already occurred would yield a specific outcome. Now, if we assume *new posterior*, *current* and *new likelihood* to be *npo*, *cur* and *nli* respectively then using sequential update scheme where we take the past into account and the modified expression for Bayesian Update Scheme is as follows:

$$npo \propto cur \times nli \tag{3}$$

In Eq. 3, *current* (*cur*) is the probability of some entity occurring whereas the *new likelihood* (*nli*) is the Bayesian Update taking posterior from the past into account. This approach sometimes is also called a Recursive Bayesian Update. For our scenario, we are trying to predict the current probability for the label (category), which becomes the *new posterior* (*npo*) in the equation, *current* is the probability prediction of the category of the image frame provided by the CNN and *new likelihood* is the probability of the category occurring in some previous time steps. Here, the reason to mention *some previous time steps* is because the number of previous time steps to take into consideration will be a heuristic choice of the user. In our case, we call the number of previous frames (images), which is considered to provide an integrated prediction for the chosen category, as *Frame Window*. *Frame Window* consists of N number of frames, which are taken into consideration.

If we consider that the prediction for the category in the current frame as $P_{this}^{category}$, prediction for the same category in the previous frame as $P_{this-1}^{category}$ and the total prediction accuracy of the model after the training/cross-validation of the model is complete as P_{CNN} then the updated equation for Bayes' Theorem is as follows:

$$P_{updated}^{category} \propto P_{this-1}^{category} \times P_{this}^{category} \tag{4}$$

Eq. 4 could be utilized to predict the frame using the prediction of previous frame as follows:

$$P_{updated}^{category} = \frac{P_{this-1}^{category} \times P_{this}^{category}}{(P_{this-1}^{category} \times P_{this}^{category}) + P_{CNN}} \tag{5}$$

In Eq. 5, $P_{updated}^{category}$ is the updated prediction using Bayes' Theorem for the same category by the CNN model. We should also note that both $P_{this-1}^{category}$ and $P_{this}^{category}$ are *conjugate priors* for our scenario since they belong to the same category as the posterior ($P_{updated}^{category}$) and hence, in the same probability distribution family. Now, depending on the *Frame Window*, the evaluation of $P_{updated}^{category}$ will vary, which leads us to an updated equation as follows:

$$P_{updated}^{category} = \begin{cases} P_{this}^{category}, & \text{if } N = 0 \\ \dfrac{P_{this-1}^{category} \times P_{this}^{category}}{(P_{this-1}^{category} \times P_{this}^{category}) + P_{CNN}}, & \text{if N = 1} \\ \prod_{1}^{N} \dfrac{P_{this-N}^{category} \times P_{this-(N-1)}^{category}}{(P_{this-N}^{category} \times P_{this-(N-1)}^{category}) + P_{CNN}}, & \text{if } N > 1 \end{cases} \tag{6}$$

Eq. 6 is the governing equation, which is utilized to predict the probability of the category during the *Frame Window*.

In the Prediction module, IRON-MAN has a queue implementation of the image stack (called as *Image Queue*), where the N number of frames are stored and N is defined by the user to denote the size of the *Frame Window*. When an $(N+1)^{th}$ image frame comes from the camera for prediction, the images stored at 1^{st} position of the *Image Queue* is popped out and the $(N+1)^{th}$ image frame is pushed in the N^{th} position of the queue while everything getting shifted a place in the middle just like in a first-in-first-out (FIFO) queue implementation. When prediction for a particular frame is required, the prediction of the frame by the CNN model is provided as well as the prediction of the *Frame Window* is provided by using the Eq. 6. After utilizing Eq. 6 the updated prediction ($P_{updated}^{category_i}$) for a specific category i is compared with the the updated prediction of other categories and the label for the maximum value of the prediction is provided as output. In our experimentation, we use $N = 7$, which is discovered empirically to produce the best prediction result. More details on the selection of the value of N is provided in Sec. VIII.

C. ECTI: ENERGY CONSUMPTION PER TRAINING IMAGE

A new metric, *ECTI* (*Energy Consumption per Training Image*), is introduced to choose the suitability of a CNN model in embedded systems. If we consider ET as the total execution time period required to train the CNN with a dataset I consisting of n number of images to achieve a validation prediction accuracy of P, Q as the *quality of experience*, and the average power consumption per second during the training period as e then the equation for *ECTI* could be defined as follows:

$$ECTI = (\frac{ET}{n} \times e) \text{ iff } P \geq Q \tag{7}$$

The unit of *ECTI* is $kilo-watt-hour$ (kWh), where ET is represented in hours and e in *kilo-Watt* (kW). To choose the most suitable CNN for an embedded application we have to select the CNN with the least value of *ECTI*.

VI. EXPERIMENTAL RESULTS

A. DATASET USED

We have performed our validation on two different test cases: *traffic categorization* and *pedestrian obstruction*.

1) TRAFFIC CATEGORIZATION DATASET

For our *traffic categorization* experimentation we are using the same dataset used in [5], [6], [14]. Mainly two dataset are used in this experiment. The first dataset is released by UCSD traffic control department [13]. This dataset contains 254 highway video sequences, all of which are filmed using the same camera containing light, heavy and traffic jams filmed at different periods of the day under different weather conditions. Each UCSD video has a resolution of 320 × 240 pixels with a frame rate of 10 fps. The video streams from the UCSD dataset were converted to images by processing 1 frame out of every 8 frames (∼1.3 fps). This UCSD dataset was used as the testing dataset. Since UCSD dataset is categorized into 3 labels: *Light, Medium and Heavy*, we manually annotated the images into our desired 4 categories: *Jam, Heavy, Fluid, Empty*. When the road was empty the images were categorized as Empty and for light traffic it is labeled as Fluid. When the traffic was heavy the images are labeled as Heavy and for slow moving heavy traffic the images are labeled as Jam. Another dataset consisting of the 400 images, which is used in [5] and [6], captured from highway cameras deployed all over the UK and also consist of several examples of different weather and lighting conditions in order to provide a better training performance. These 400 images are also segregated into 4 categories: *Jam, Heavy, Fluid, Empty*; with each category having 100 images. This dataset was used for training and validation purposes.

2) PEDESTRIAN OBSTRUCTION DATASET

For our *pedestrian obstruction* detection experimentation we have used the pedestrian dataset used by Li *et al.* [54], which consists of more than 5 million images for a total of 32361 labeled vulnerable road users (VRUs), including cyclists, pedestrians, tri-cyclists and motor-cyclists etc. To evaluate our proposed IRON-MAN approach we have only used the training dataset of [54] consisting of 9742 images for both training and validation, whereas we used the validation dataset (for cross-validation) from the same study for testing purposes. The images were manually labeled into 2 categories: *Obstruction, No-Obstruction*. Whenever there were pedestrians in front or nearby the camera view, then it is categorized as obstruction and in contrary it is labeled as no-obstruction.

B. HARDWARE SETUP

We have implemented the methodology on an Odroid XU4 [15] (see Fig. 4), which employs Exynos 5422 MPSoC [60] used in popular Samsung Note phones and phablets. Exynos 5422 implements ARM's big.LITTLE architecture utilizing 4 ARM Cortex A-15 big CPUs, 4 ARM Cortex A-7 LITTLE CPUs and 6 MALI T628 MP6 GPUs. The Odroid XU4 does not have an internal power sensor, and we had to use an external power monitor, Odroid smart power 2 [61], with networking capabilities over WIFI to take power consumption readings. The Odroid smart power 2 [61] is capable of powering single-board devices with power requirement ranging from 4 to 5.3 volts and 5 amp in 100 mV step, and was used to monitor the power consumed by the MPSoC. Odroid-XU4 platform has 4 temperature sensors on 4 ARM Cortex A-15 big CPU cores, which we read to monitor temperature behavior during our experiments. For all our experiments we have only monitored 4 A-15 big CPU cores to observe the operating temperature, and the highest temperature among the big CPUs is considered for empirical evaluation.

The Odroid XU4 runs on UbuntuMate OS version 14.04 (Linux Odroid Kernel: 3.10.105), which is executed on ondemand power saving scheme of Linux and hence, implements its respective power and thermal management scheme. The Odroid XU4 also supports external micro-SD memory card and we have utilized a 256 GB SanDisk Extreme micro-SD card [62], capable of reading up to 160MB/s and writing up to 90MB/s, to hold the dataset image frames for training. In the traffic dataset, each image frame with a resolution of 320 × 240 pixels consumes approximately 12KB in memory and on the 256 GB micro-SD memory card we are able to store more than 21,333,333 image frames on the device.

C. EXPERIMENTAL RESULTS

1) TRAFFIC CATEGORIZATION

To categorize traffic we chose four pre-Trained CNN models, which were trained on millions of ImageNet images, for our validation. These four CNN models are VGG16, VGG19 (a deeper network of VGG16), ResNet50 and MobileNet. In our experiments, we have chosen the *quality of experience* (Q) to be 0.7 i.e. 70%. Through empirical evidence it was noticed that the CNN model performs best when the prediction accuracy of 70% or more is chosen in order to be utilized for traffic categorization application in the real world, hence, 0.7 was chosen to be the *quality of experience*. For VGG16, VGG19, ResNet50 and MobileNet it took us 360, 360, 330, 360 images respectively to train the pre-Trained using Transfer Learning and our proposed retrain approaches (see Sec. V-A), and gained a testing prediction accuracy of 98.93%, 96.62%, 92.79% and 85.75% respectively. The total execution time of the CNN model training, average power consumption and average operating temperature on the MPSoC during the training of the respective CNN models are shown in Fig. 5.

Based on Eq. 7 the evaluated ECTI values for VGG16, VGG19, ResNet50 and MobileNet are shown in Fig. 6. On the X-axis of Fig. 6 the respective CNN models are reflected, whereas, the Y-axis reflects the ECTI value for the corresponding model in kWh. Based on the values from Fig. 6 MobileNet is the most energy efficient model, which also has a prediction accuracy over the chosen *quality of experience*

TABLE 1. Predictions of traffic categorization image frames with and without IRON-MAN.

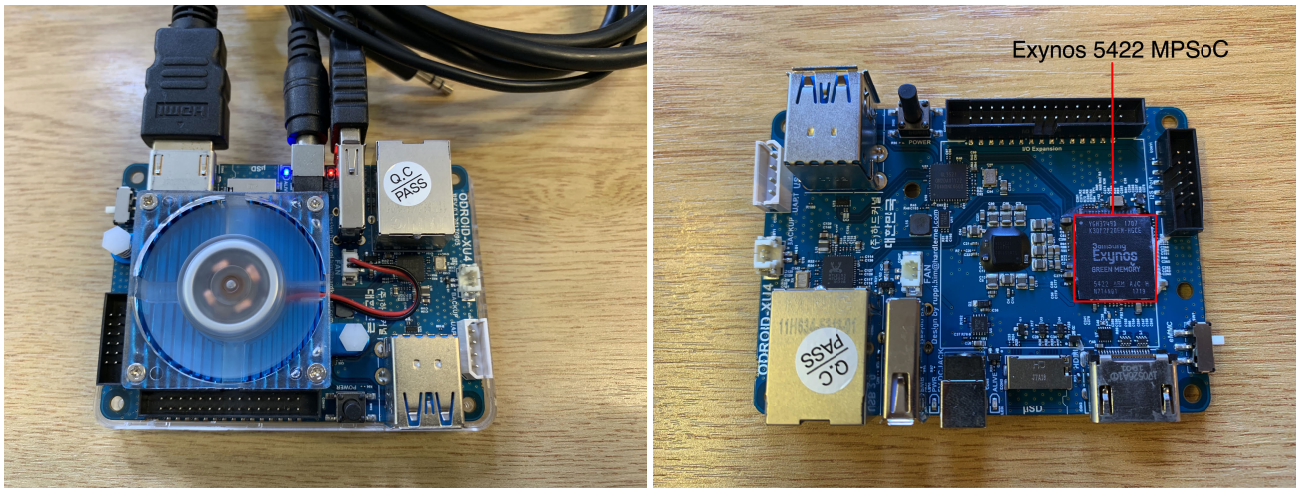
Image frames	Normal predictions without IRON-MAN	Predictions by IRON-MAN
	**(Label) | Empty | Fluid | Heavy | Jam** (Fluid) | 0.0100 | **0.7930** | 0.1683 | 0.0286 (Heavy) | 0.0131 | 0.3091 | **0.5098** | 0.1681 (Jam) | 0.0131 | 0.2754 | 0.3399 | **0.3717**	**(Label) | Empty | Fluid | Heavy | Jam** (Fluid) | 0.01 | **0.793** | 0.1683 | 0.0286 (Fluid) | 0.0001 | **0.1986** | 0.0798 | 0.0048 (Fluid) | 0.0 | **0.0524** | 0.0267 | 0.0018

TABLE 2. Predictions of pedestrian obstruction image frames with and without IRON-MAN.

Image frames	Normal predictions without IRON-MAN	Predictions by IRON-MAN
	**(Label) | No-Obstruction | Obstruction** (Obstruction) | 0.3270 | **0.6730** (No-Obstruction) | **0.5010** | 0.4990 (Obstruction) | 0.4600 | **0.5400**	**(Label) | No-Obstruction | Obstruction** (Obstruction) | 0.3270 | **0.6730** (Obstruction) | 0.1843 | **0.3166** (Obstruction) | 0.1047 | **0.1908**

(Q), that could be utilized for training on the device utlizing MPSoC.

Now, to prove efficacy of our integrated rational prediction of image frames from the video, we randomly chose a video from UCSD traffic dataset [13] and broke the video into image frames, which represents the same category as the video itself. We chose a video from medium traffic category, which corresponds to *Fluid* traffic category, and utilized the MobileNet model as the CNN model in IRON-MAN to predict the label of a sequence of image frames as well as the label for video. For this experiment we chose the *Frame Window* of 3 images representing a video sequence of 4 seconds (approx.). Table. 1 shows the prediction of labels if we only use MobileNet without our IRON-MAN approach, where prediction of previous image frames are not taken into consideration, and if we use IRON-MAN, where prediction from previous image frames are taken into consideration, as well. The table shows that IRON-MAN is able to demon-



(a) Odroid XU4 in action (b) Exynos 5422 MPSoC on Odroid XU4 development board

FIGURE 4. Odroid XU4 development board and Exynos 5422 MPSoC.

strate the prediction of the correct label of the image frames belonging to the same video category.

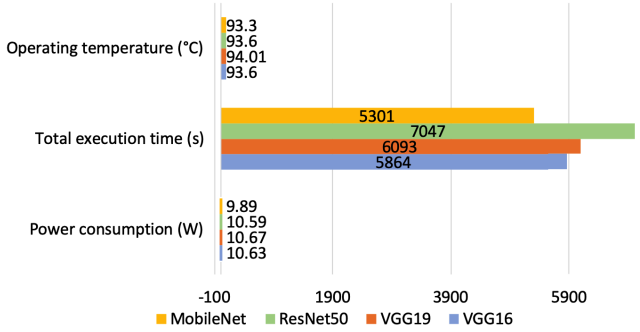


FIGURE 5. Total execution time of the CNN model training, average power consumption and average operating temperature on the MPSoC for the following CNN models: MobileNet, ResNet50, VGG16, VGG19.

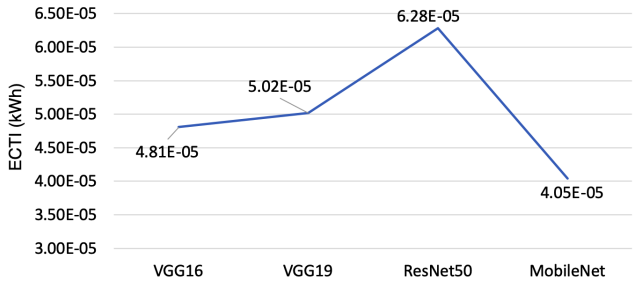


FIGURE 6. Evaluated ECTI values for VGG16, VGG19, ResNet50 and MobileNet CNN models.

2) PEDESTRIAN OBSTRUCTION

In another experiment, we chose simultaneous image frames having pedestrians as obstruction from the pedestrian dataset [54] to validate efficacy of IRON-MAN. Table 2 shows that IRON-MAN was again able to predict whether the path is obstructed by a pedestrian or not using integrated results from previous frames. MobileNet CNN was utilized in this experiment, which gained a testing prediction accuracy of 79.50% after *transfer learning* and *retraining* approaches for this application. MobileNet was also selected as the choice of CNN model due to its energy efficiency (refer to Sec. VI-C1) on the MPSoC device.

a: EFFECT ON LIFESPAN OF THE DEVICE

Given the architecture of IRON-MAN, where training/inference is automated and executing at all times on the device, and based on the assumption that an increase in the operating temperature by 10-15° centigrades could reduce the lifespan of the device by 2× [33], [35], we evaluated the effect on the device's lifespan for different CNN models used in IRON-MAN. We also noticed that the maximum *baseline temperature*, which is the operating temperature of the CPU core when idle i.e. only executing background tasks while on Linux's ondemand power scheme, of the ARM Cortex A-15 big CPU core was 69.24°C (average). Therefore, the deviation of operating temperature while training and the baseline temperature was 24.36 °C (average) for VGG16, 24.77 °C (average) for VGG19, 24.48 °C (average) for ResNet50 and 24.06 °C (average) for MobileNet. Therefore, considering that the device lifespan reduces by 2× for every 10°C increase in operating temperature, if training is performed on the device then the lifespan of the same device reduces by 4.872× ($\simeq \frac{24.36}{10} \times 2$) for utilizing VGG16, 4.954× ($\simeq \frac{24.77}{10} \times 2$) for utilizing VGG19, 4.896× ($\simeq \frac{24.48}{10} \times 2$) for utilizing ResNet50 and 4.812× ($\simeq \frac{24.06}{10} \times 2$) for utilizing MobileNet. Therefore, until the application requires the CNN to be trained on the embedded MPSoC to continue providing desirable analysis it is highly recommended that the CNN is not trained on the embedded system, and instead, trained off

device (on a server) with hardware capabilities to accelerate CNN model training.

b: COMPARATIVE STUDY OF IRON-MAN

To evaluate the efficacy of *IRON-MAN*, we compared the methodology with the state-of-the-art approach for traffic categorization proposed by Luo *et al.* [14]. Since, the methodology proposed in [14] is closely related to the target application of traffic categorzation without motion features, it is a very suitable methodology for a comparative study with *IRON-MAN*. Additionally, *IRON-MAN* being the first methodology to perform *TMAV* in traffic categorization, it would be unreasonable to compare the methodology with any other existing approaches. In [14], the researchers have used SegCNN and RegCNN to analyze and categorize traffic. The main motivation of utilizing SegCNN and RegCNN in their approach is to improve the accuracy of the prediction model even without training the CNN with a large dataset (consisting of millions of images). After the training, Luo *et al.*'s approach was able to achieve a prediction accuracy of 94.8% during testing for traffic categorization, which is comparable to prediction accuracy achieved utilizing ResNet in IRON-MAN (refer to Sec. VI-C1), however, it has lower prediction accuracy than VGG when chosen for IRON-MAN. Since, ResNet achieved comparable prediction accuracy with Luo *et al.*'s approach, hence, we utilized ResNet in IRON-MAN to perform the comparative study. It should be kept in mind that given the complexity of [14], we trained the model on a general purpose computer with GPU accelerator and then the trained model was transferred to the Odroid XU4 and utilized for prediction (inference) only. Therefore, in this comparative study we have only provided a comparison based on inference time.

Moreover, given the computation complexity of [14], executing (inference) the approach has an overhead of 5.2× on the Odroid XU4 MPSoC than compared to *IRON-MAN*. Fig. 7 shows the execution time (in seconds) of analyzing 9 sequential image frames of the same video from the UCSD dataset [13]. Each test was performed 5 times on the Odroid XU4, utilizing all eight CPU cores (big.LITTLE CPU cores) and the average execution time for image analysis for each number of image frame is provided in Fig. 7. Since the concept of *frame window* or temporal analysis of image frames of video does not exist in [14], we sequentially feed each image frames from the video to the methodology to get the output analysis. From Fig. 7 we can notice that it takes 1.487275 seconds for Luo *et al.*'s approach to analyze 9 sequential image frames, whereas it takes 0.285775 seconds to analyze 9 sequential image frames in the *frame window* for the *IRON-MAN*. Therefore, the overhead associated with Luo *et al.*'s approach is 5.2× (approx.) on the Odroid XU4 MPSoC.

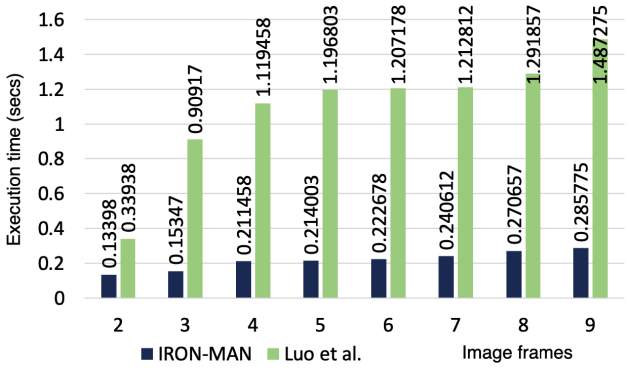


FIGURE 7. Execution time taken to analyze 9 image frames sequentially from the traffic video by different methods (Execution time in seconds vs number of image frames analyzed).

VII. LARGE-SCALE INTEGRATION OF IRON-MAN OVER THE CLOUD

IRON-MAN can be deployed in large-scale over the cloud/Internet using database/web-services. Fig. 8 shows the diagrammatic representation of a large scale integration of IRON-MAN, where the network consists of n number of Odroid XU4 devices, each connected to their own camera sensors and each Odroid is denoted as Dev^i (i^{th} device). Each Odroid performs prediction using IRON-MAN and then the prediction data is relayed to an online database/web-service in the cloud for further processing.

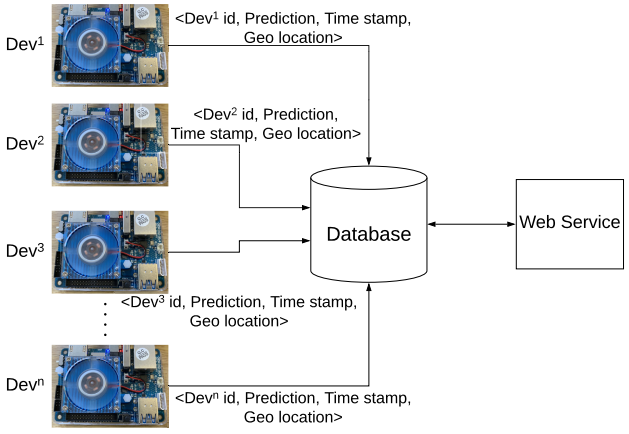


FIGURE 8. Diagram representing large-scale integration of IRON-MAN over the cloud.

In order to identify data from each device accurately, data consisting of device id, prediction result, time stamp and Geo-location of the device are sent as a JSON (**J**ava**S**cript **O**bject **N**otation) object over the Internet to the cloud. Listing 1 shows the python code implemented in the Odroid XU4 to send prediction data from the device as a JSON object, where the device id is the mac address of the Odroid to properly identify the device, location is the Geo-location (latitude, longitude) of the implementation of the Odroid and time is the current time-stamp when the prediction was performed and relayed over the network. Fig. 9 shows the JSON object output which is relayed to the online database/web-service.

Performance of IRON-MAN over the cloud: We implemented the online database/web-service by utilizing Google Cloud Platform's (GCP) [63] Compute Engine, which is a

```python
import json
from datetime import datetime

# a Python object (dictionary)
data = {
  "devid": "00:1e:06:32:bb:02",
  "prediction": "fluid",
  "time": str(datetime.now()),
  "location": "51.509865,-0.118092"
}

# convert into JSON:
json_data = json.dumps(data)
```

Listing 1. Example of python code snippet to send prediction data as JSON object over the Internet network for large-scale integration


```
{"location": "51.509865,-0.118092",
"devid": "00:1e:06:32:bb:02",
"prediction": "fluid",
"time": "2020-07-07 14:54:12.419049"}
```


FIGURE 9. JSON object output of the python code snippet (Listing 1) to send prediction data as JSON object over the Internet network for large-scale integration.

virtual machine consisting of 2 vCPUs, 2 GB RAM memory and 100 GB disk memory space. On the online virtual machine (on the cloud) we implemented the web-service, which processes the JSON data sent from each Odroid device (we utilized 3 Odroid devices, each with their own camera sensors, connected to the online virtual machine over the Internet). After the data from the Odroid is sent to the online virtual machine, the prediction data (device id, prediction, Geo-location & time stamp) is stored in the database on the virtual machine. The Odroid devices were set in Colchester, UK, whereas, the GCP online virtual machine is set-up in Western Europe region. The communication overhead between each Odroid and the online virtual machine was 112.57 milliseconds on an average. The online virtual machine, with its current computing resource, is capable of handling up to 300 connection requests per seconds without contributing to the communication overhead. Therefore, every second 300 Odroid devices can send data to the online virtual machine in parallel for further processing as mentioned earlier.

VIII. LIMITATIONS AND DISCUSSION

Lemma 1: Updated prediction ($P^{category}_{updated}$) of the category using Baye's Theorem tends to zero as prediction ($P^{category}_{this}$) for the category in the current frame tends to zero.

In the Eq. 4, if $P^{category}_{this} \longrightarrow 0$ then $P^{category}_{updated} \longrightarrow 0$ (see Eq. 8). Hence, for a lower value of $P^{category}_{this}$ we would be achieving a lower value of $P^{category}_{this}$ if prediction ($P^{category}_{this-1}$) for the same category in the previous image frame is less than or equal to one[7] ($P^{category}_{this-1} \leq 1$).

$$P^{category}_{updated} \longrightarrow 0 \text{ as } P^{category}_{this} \longrightarrow 0 \tag{8}$$

Theorem 1: If $N \geq 1$ in the *frame window* and $P^{category}_{this} \longrightarrow 0$ then the updated prediction ($P^{category}_{updated}$) of the category converging to zero increases as N increases.

In Eq. 6, if the number of image frames (N) is greater than one then the updated prediction for the category is represented by Eq. 9.

$$P^{category}_{updated} = \prod_{1}^{N} \frac{P^{category}_{this-N} \times P^{category}_{this-(N-1)}}{(P^{category}_{this-N} \times P^{category}_{this-(N-1)}) + P_{CNN}}, \quad \text{if } N > 1 \tag{9}$$

$$P_k = \frac{P^{category}_{this-N} \times P^{category}_{this-(N-1)}}{(P^{category}_{this-N} \times P^{category}_{this-(N-1)}) + P_{CNN}} \tag{10}$$

In the Eq. 9, if we consider the term P_k as shown in Eq. 10, since the preceding term is part of the product series represented in Eq. 9 and consider $P^{category}_{updated}$ as a function ($F_{N,i}$) of N and i, where i is the i_{th} image in the frame window consisting of N image frames, and both i and N are whole numbers ($N, i \in W$ and $N > i \geq 1$), then the aforementioned equation (Eq. 9) could be represented as follows:

$$F_{N,i} = \prod_{k=N-i+1}^{N} P_k \tag{11}$$

Since $P^{category}_{this}$ is an integral part of P_k in Eq. 11 (see Eq. 9), using the knowledge from *lemma 1* we can say that as $P^{category}_{this} \longrightarrow 0$ then $P_k \longrightarrow 0$ as well. Now, as $N \longrightarrow \infty$ and $P_k \longrightarrow 0$ then $F_{N,i} \longrightarrow 0$. Therefore, using this knowledge we could state that: $0 \leq F_{N,i} \leq P_k$ and proving the fact that as N gets larger and $P^{category}_{this}$ gets smaller (close to zero), the updated prediction ($P^{category}_{updated}$) also converges to zero.

Through our empirical data we noticed that the aforementioned theorem holds true and this could be considered as a potential limitation of using Bayes' Theorem along with CNN's prediction as proposed in IRON-MAN. In our experiments, when the number of image frames in the frame window was selected to be more than 7, the prediction for the category became equal to zero for all the classes and hence, a frame window of more than 7 image frames could not be chosen for accurate predictions. Now, if we consider t seconds as the time interval between two image frames in the video then because of Lemma 1 and Theorem 1 using IRON-MAN methodology we are only able to analyze a snippet of the video of $t \times 7$ seconds duration. If we consider the traffic categorization problem where each image frames were taken every 1.3 seconds (see Sec. VI-A) interval then using IRON-MAN we are able to analyze 9.1 (1.3×7) seconds of the video with high prediction accuracy.

[7]Prediction of an image frame could not be more than one since one represents 100% probability of the category occurring and probability could only range from 0 to 100.

In order to overcome the aforementioned limitations of IRON-MAN as mentioned earlier, we need to develop a more robust algorithm so that we are able to analyze larger snippets of videos (larger than 9.1 seconds for traffic categorization problem) without motion features more accurately.

IX. AN ALTERNATIVE ONLINE PREDICTION: AI-PREDICTION

To overcome the limitations mentioned in Sec. VIII, we develop an alternate online prediction module to replace the original prediction module proposed in V-B and improve scalability such that more number of image frames could be considered for the holistic analysis/prediction of the video. We call this alternate online prediction module as the **A**veraged **I**ntegrated **Prediction** (we also refer to this module as the *AI-Prediction*), which is implemented as a software agent in the application layer of the system.

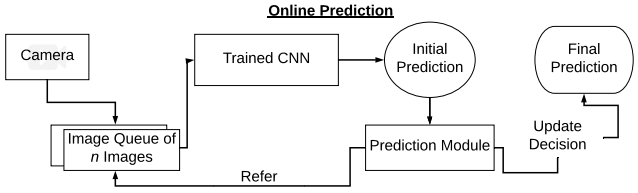


FIGURE 10. Block digram of an alternative Online Prediction (*AI-Prediction*).

The attached camera or video stream sends the image frames to a queue implementation of images called as the *Image Queue*, which is capable of holding n number of images along with their associated prediction scores for each classes of the target task. The latest image (i^{th}) of the Image Queue is sent to the *Trained CNN*, which is trained in the *Offline Training* module for prediction. The CNN then predicts the associated prediction scores for each classes of the target task of the current image frame (i^{th} image frame). If the Image Queue is full then the n^{th} image is the i^{th} image or else the last image of the Image Queue is the i^{th} image. When the CNN predicts the i^{th} image frame, it is called the *Initial Prediction* (P_{CNN_i}). P_{CNN_i} is a set of prediction scores for different classes of the target task for the i^{th} image frame and P_{CNN_i} is forwarded to a sub-module of *TMAV-CNN* named *Prediction Module*. The prediction module has two operations: *Refer* and *Update Decision*. **Note**: When a new image frame is inserted into the Image Queue and if the queue happens to be full then the 1st image frame of the queue is popped (called as PopFront() function) and the new image frame is inserted at the n^{th} position of the queue (called as PopBack() function). The Image Queue works the same way queue is implemented in data-structures.

1) REFER

If we assume that the multi-class target task consists of m number of classes (labels) such that the classes could be represented as $\{label_1, label_2, \ldots .label_m\}$ and the associated prediction scores for the classes of i^{th} image frame could be represented as $\{P^{label_1}_{CNN_i}, P^{label_2}_{CNN_i}, \ldots .P^{label_m}_{CNN_i}\}$, then P_{CNN_i} could be represented as a set such that $P_{CNN_i} = \{P^{label_1}_{CNN_i}, P^{label_2}_{CNN_i}, \ldots .P^{label_m}_{CNN_i}\}$. Now, the prediction module updates the prediction scores of the i^{th} image frame in the Image Queue and refers to the prediction scores for each classes of $(n-1)$ number of image frames and evaluates the average prediction score for each classes for those image frames (shown in Eq. 12). The reason to evaluate the average prediction scores for each classes for $(n-1)$ number of image frames is to take those prediction scores for previous image frames into consideration before providing the final prediction of the i^{th} image so that a holistic scene prediction is provided instead of just providing the prediction for the i^{th} image. Here, for ease of understanding let us assume that the Image Queue is always full such that i^{th} image frame is same as the n^{th} image frame.

$$Avg(P_{CNN_{n-1}}) = \frac{1}{n-1} \times \sum_{1}^{n-1} P_{CNN} \tag{12}$$

2) UPDATE DECISION

Now, in the Update Decision operation of the prediction module to provide the final prediction ($P^{final}_{CNN_n}$) of the n^{th} image frame, the prediction module evaluates the weighted average of n number of image frames in the Image Queue using Eq. 12 such that the governing equation is represented as Eq. 13. Since $P^{final}_{CNN_n}$ ($P^{final}_{CNN_n} = \{P^{final\ label_1}_{CNN_n}, P^{final\ label_2}_{CNN_n}, \ldots .P^{final\ label_m}_{CNN_n}\}$ where $P^{final\ label_i}_{CNN_n}$ is the respective prediction of ith class for the nth image frame) is a set consisting of weighted average of the prediction scores for m classes of the target task, to provide the final prediction label ($P^{final\ label}_{CNN_n}$) the associated label of the maximum of $P^{final}_{CNN_n}$ is provided as a result (see Eq. 14).

$$\begin{aligned} P^{final}_{CNN_n} &= \frac{(n-1) \times P_{CNN_n} + Avg(P_{CNN_{n-1}})}{n} \\ &= \frac{(n-1) \times P_{CNN_n} + (\frac{1}{n-1} \times \sum_{1}^{n-1} P_{CNN})}{n} \end{aligned} \tag{13}$$

$$P^{final\ label}_{CNN_n} = Max(P^{final}_{CNN_n}) \tag{14}$$

A. LIMITATIONS OF AI-PREDICTION

Unlike the limitations shown in Sec. VIII, the number of image frames chosen to be considered for holistic prediction of the video could be more than 7, since, the $P^{final}_{CNN_n}$ does not tend to zero if prediction of any intermediate class tends to zero. However, given the fact that *AIPrediction* evaluates the final prediction based on the average prediction of the image frames, such methodology could lead to inaccuracy if the chosen n number of image frames considered for the prediction is very large. Therefore, it is to be kept in mind that the developer/engineer of the IRON-MAN methodology has to choose whether to use prediction module based on Bayesian updating scheme (as proposed in Sec. V-B) or use *AIPrediction* module depending on the accuracy vs scalability requirement of the target application.

X. CONCLUSION

In this paper, we propose IRON-MAN, which is capable of providing *Temporal Motionless Analysis of Videos* (*TMAV*) i.e. analyzing videos without motion features and providing a holistic temporal analysis while utilizing predictions of the past image frames into consideration. Based on the results we have shown that training CNN based approaches on MPSoCs could lead up to 4.8× (approx.) reduction in lifespan of the embedded device and MobileNet is more energy efficient compared to VGG and ResNet50 models. Therefore, it is recommended to perform the training off the embedded device for improved longevity or utilize MobileNet for on-device traffic categorization. It is also shown that for traffic categorization application, our approach outperforms the state-of-the-art. We have also discussed limitations of IRON-MAN and provided an alternative approach to improve scalability of the proposed methodology.

REFERENCES

[1] S. Chakradhar, M. Sankaradas, V. Jakkula, and S. Cadambi, "A dynamically configurable coprocessor for convolutional neural networks," *ACM SIGARCH Comput. Archit. News*, vol. 38, no. 3, pp. 247–257, Jun. 2010.

[2] X.-W. Chen and X. Lin, "Big data deep learning: Challenges and perspectives," *IEEE Access*, vol. 2, pp. 514–525, 2014.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556

[4] G. Kalliatakis, S. Ehsan, M. Fasli, A. Leonardis, J. Gall, and K. D. McDonald-Maier, "Detection of human rights violations in images: Can convolutional neural networks help?" in *Proc. 12th Int. Joint Conf. Comput. Vis., Imag. Comput. Graph. Theory Appl.*, vol. 5, 2017.

[5] Z. Luo, P.-M. Jodoin, S.-Z. Li, and S.-Z. Su, "Traffic analysis without motion features," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2015, pp. 3290–3294.

[6] S. Dey, G. Kalliatakis, S. Saha, A. K. Singh, S. Ehsan, and K. McDonald-Maier, "MAT-CNN-SOPC: Motionless analysis of traffic using convolutional neural networks on system-on-a-programmable-chip," in *Proc. NASA/ESA Conf. Adapt. Hardw. Syst. (AHS)*, Aug. 2018, pp. 291–298.

[7] S. Dey, A. K. Singh, D. K. Prasad, and K. D. Mcdonald-Maier, "Temporal motionless analysis of video using CNN in MPSoC," in *Proc. 31st IEEE Int. Conf. Appl.-Specific Syst., Archit. Processors (ASAP)*, Jul. 2020.

[8] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: A benchmark," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 304–311.

[9] F. Flohr and D. Gavrila, "Daimler pedestrian segmentation benchmark dataset," in *Proc. Brit. Mach. Vis. Conf.*, 2013.

[10] S. Burton, L. Gauerhof, and C. Heinzemann, "Making the case for safety of machine learning in highly automated driving," in *Proc. Int. Conf. Comput. Saf., Rel., Secur.* Cham, Switzerland: Springer, 2017 pp. 5–16.

[11] G. Zhang, B. E. Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks: The state of the art," *Int. J. Forecasting*, vol. 14, pp. 35–62, Mar. 1998.

[12] M. Grecu and W. F. Krajewski, "Detection of anomalous propagation echoes in weather radar data using neural networks," *IEEE Trans. Geosci. Remote Sens.*, vol. 37, no. 1, pp. 287–296, Jan. 1999.

[13] A. B. Chan and N. Vasconcelos, "Classification and retrieval of traffic video using auto-regressive stochastic processes," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2005, pp. 771–776.

[14] Z. Luo, P.-M. Jodoin, S.-Z. Su, S.-Z. Li, and H. Larochelle, "Traffic analytics with low-frame-rate videos," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 4, pp. 878–891, Apr. 2018.

[15] *Odroid-xu4*. Accessed: Jul. 23, 2018. [Online]. Available: https://goo.gl/KmHZRG

[16] *GPU Servers for Machine Learning Startups: Cloud vs on-Premise?* Accessed: Jul. 7, 2020. [Online]. Available: https://medium.com/@thereibel/gpu-servers-for-machine-learning-startups-cloud-vs-on-premise-9a9dedfcadc9

[17] T. Yagi, K. Mangalam, R. Yonetani, and Y. Sato, "Future person localization in first-person videos," 2017, *arXiv:1711.11217*. [Online]. Available: http://arxiv.org/abs/1711.11217

[18] G.-Y. Lai, K.-H. Chen, and B.-J. Liang, "People trajectory forecasting and collision avoidance in first-person viewpoint," in *Proc. IEEE Int. Conf. Consum. Electronics-Taiwan (ICCE-TW)*, May 2018, pp. 1–2.

[19] R. De Geest, E. Gavves, A. Ghodrati, Z. Li, C. Snoek, and T. Tuytelaars, "Online action detection," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 269–284.

[20] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human trajectory prediction in crowded spaces," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 961–971.

[21] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 487–495.

[22] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "SUN database: Large-scale scene recognition from abbey to zoo," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 3485–3492.

[23] S. Dey, A. K. Singh, and K. D. McDonald-Maier, "P-EdgeCoolingMode: An agent-based performance aware thermal management unit for DVFS enabled heterogeneous MPSoCs," *IET Comput. Digit. Techn.*, vol. 13, no. 6, pp. 514–523, Nov. 2019.

[24] A. K. Singh, A. Prakash, K. R. Basireddy, G. V. Merrett, and B. M. Al-Hashimi, "Energy-efficient run-time mapping and thread partitioning of concurrent OpenCL applications on CPU-GPU MPSoCs," *ACM Trans. Embedded Comput. Syst.*, vol. 16, no. 5, pp. 1–22, Oct. 2017.

[25] U. Gupta, C. A. Patil, G. Bhat, P. Mishra, and U. Y. Ogras, "DyPO: Dynamic pareto-optimal configuration selection for heterogeneous MpSoCs," *ACM Trans. Embedded Comput. Syst.*, vol. 16, no. 5, pp. 1–20, Oct. 2017.

[26] S. Dey, E. Z. Guajardo, K. R. Basireddy, X. Wang, A. K. Singh, and K. McDonald-Maier, "EdgeCoolingMode: An agent based thermal management mechanism for DVFS enabled heterogeneous MPSoCs," in *Proc. 32nd Int. Conf. VLSI Design 18th Int. Conf. Embedded Syst. (VLSID)*, Jan. 2019, pp. 19–24.

[27] S. Dey, A. K. Singh, X. Wang, and K. D. McDonald-Maier, "DeadPool: Performance deadline based frequency pooling and thermal management agent in DVFS enabled MPSoCs," in *Proc. 6th IEEE Int. Conf. Cyber Secur. Cloud Comput. (CSCloud)/ 5th IEEE Int. Conf. Edge Comput. Scalable Cloud (EdgeCom)*, Jun. 2019, pp. 190–195.

[28] S. Dey, A. K. Singh, S. Saha, X. Wang, and K. D. McDonald-Maier, "RewardProfiler: A reward based design space profiler on DVFS enabled MPSoCs," in *Proc. 6th IEEE Int. Conf. Cyber Secur. Cloud Comput. (CSCloud)/ 5th IEEE Int. Conf. Edge Comput. Scalable Cloud (EdgeCom)*, Jun. 2019, pp. 210–220.

[29] S. Dey, A. K. Singh, D. K. Prasad, and K. D. Mcdonald-Maier, "Socodecnn: Program source code for visual cnn classification using computer vision methodology," *IEEE Access*, vol. 7, pp. 157158–157172, 2019.

[30] S. Isuwa, S. Dey, A. K. Singh, and K. McDonald-Maier, "Teem: Online thermal-and energy-efficiency management on CPU-GPU MPSoCs," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*. Mar. 2019, pp. 438–443.

[31] S. Dey, A. K. Singh, X. Wang, and K. McDonald-Maier, "User interaction aware reinforcement learning for power and thermal efficiency of CPU-GPU mobile MPSoCs," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2020, pp. 1–6.

[32] T. Chantem, R. P. Dick, and X. S. Hu, "Temperature-aware scheduling and assignment for hard real-time applications on MPSoCs," in *Proc. Design, Autom. Test Eur.*, Mar. 2008, pp. 288–293.

[33] A. K. Coskun, T. S. Rosing, and K. Whisnant, "Temperature aware task scheduling in MPSoCs," in *Proc. Design, Autom. Test Eur. Conf. Exhib.*, Apr. 2007, pp. 1–6.

[34] A. K. Singh, S. Dey, K. R. Basireddy, K. McDonald-Maier, G. V. Merrett, and B. M. Al-Hashimi, "Dynamic energy and thermal management of multi-core mobile platforms: A survey," *IEEE Des. Test*, early access, Mar. 23, 2020, doi: 10.1109/MDAT.2020.2982629.

[35] J. Zhou, T. Wei, M. Chen, J. Yan, X. S. Hu, and Y. Ma, "Thermal-aware task scheduling for energy minimization in heterogeneous real-time MPSoC systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 8, pp. 1269–1282, Aug. 2016.

[36] D. Panigrahi, C. Chiasserini, S. Dey, R. Rao, A. Raghunathan, and K. Lahiri, "Battery life estimation of mobile embedded systems," in *Proc. 14th Int. Conf. VLSI Design*, Jan. 2001, pp. 57–63.
[37] K. Lahiri, A. Raghunathan, S. Dey, and D. Panigrahi, "Battery-driven system design: A new frontier in low power design," in *Proc. ASP-DAC/VLSI Design. 7th Asia South Pacific Design Autom. Conf. 15th Int. Conf. VLSI Design*, Jan. 2002, pp. 261–267.
[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[39] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.
[40] A. Iranfar, M. Kamal, A. Afzali-Kusha, M. Pedram, and D. Atienza, "TheSPoT: Thermal stress-aware power and temperature management for multiprocessor systems-on-chip," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 8, pp. 1532–1545, Aug. 2018.
[41] Y.-K. Jung, K.-W. Lee, and Y.-S. Ho, "Content-based event retrieval using semantic scene interpretation for automated traffic surveillance," *IEEE Trans. Intell. Transp. Syst.*, vol. 2, no. 3, pp. 151–163, Sep. 2001.
[42] L. O. A. Sobral, L. Schnitman, and F. De Souza, "Highway traffic congestion classification using holistic properties," in *Proc. 10th IASTED Int. Conf. Signal Process., Pattern Recognit. Appl.*, 2013, pp. 1–8.
[43] O. Asmaa, K. Mokhtar, and O. Abdelaziz, "Road traffic density estimation using microscopic and macroscopic parameters," *Image Vis. Comput.*, vol. 31, no. 11, pp. 887–894, Nov. 2013.
[44] S. Kamijo, Y. Matsushita, K. Ikeuchi, and M. Sakauchi, "Traffic monitoring and accident detection at intersections," *IEEE Trans. Intell. Transp. Syst.*, vol. 1, no. 2, pp. 108–118, Jun. 2000.
[45] S. Hu, J. Wu, and L. Xu, "Real-time traffic congestion detection based on video analysis," *J. Inf. Comput. Sci.*, vol. 9, no. 10, pp. 2907–2914, 2012.
[46] A. Riaz and S. A. Khan, "Traffic congestion classification using motion vector statistical features," *Proc. SPIE*, vol. 9067, Dec. 2013, Art. no. 90671A.
[47] F. Porikli and X. Li, "Traffic congestion estimation using HMM models without vehicle tracking," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2004, pp. 188–193.
[48] K. G. Derpanis and R. P. Wildes, "Classification of traffic video based on a spatiotemporal orientation analysis," in *Proc. IEEE Workshop Appl. Comput. Vis. (WACV)*, Jan. 2011, pp. 606–613.
[49] W. Zhang, L. Chen, W. Gong, Z. Li, Q. Lu, and S. Yang, "An integrated approach for vehicle detection and type recognition," in *Proc. IEEE 12th Int. Conf Ubiquitous Intell. Comput. IEEE 12th Int. Conf. Autonomic Trusted Comput. IEEE 15th Int. Conf. Scalable Comput. Commun. Associated Workshops (UIC-ATC-ScalCom)*, Aug. 2015, pp. 798–801.
[50] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos, "A unified multi-scale deep convolutional neural network for fast object detection," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 354–370.
[51] Y. Zhou, H. Nejati, T.-T. Do, N.-M. Cheung, and L. Cheah, "Image-based vehicle analysis using deep neural network: A systematic study," in *Proc. IEEE Int. Conf. Digit. Signal Process. (DSP)*, Oct. 2016, pp. 276–280.
[52] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.
[53] S. Grigorescu, B. Trasnea, T. Cocias, and G. , "A survey of deep learning techniques for autonomous driving," *J. Field Robot.*, vol. 37, no. 3, pp. 362–386, Apr. 2020.
[54] X. Li, F. Flohr, Y. Yang, H. Xiong, M. Braun, S. Pan, K. Li, and D. M. Gavrila, "A new benchmark for vision-based cyclist detection," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2016, pp. 1028–1033.
[55] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2014, pp. 818–833.
[56] A. Doucet, S. Godsill, and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," *Statist. Comput.*, vol. 10, no. 3, pp. 197–208, Jul. 2000.
[57] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE CVPR*, Jun. 2016, pp. 770–778.
[58] 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," 2017, *arXiv:1704.04861*. [Online]. Available: http://arxiv.org/abs/1704.04861
[59] S. Arlot and A. Celisse, "A survey of cross-validation procedures for model selection," *Statist. Surv.*, vol. 4, pp. 40–79, 2010.
[60] *Exynos 5 Octa (5422)*. Accessed: Jul. 23, 2018. [Online]. Available: https://www.samsung.com/exynos
[61] *Odroid Smartpower2*. Accessed: Jul. 7, 2020. [Online]. Available: https://www.hardkernel.com/shop/smartpower2-with-15v-4a/
[62] *Sandisk Extreme Microsdxc UHS-I Card*. Accessed: Jul. 7, 2020. [Online]. Available: https://shop.westerndigital.com/en-gb/products/memory-cards/sandisk-extreme-uhs-i-microsd?utm_medium=pdsh2&utm_source=gads&utm_campaign=SanDisk-UK-PLA&utm_content=943765731087&utm_term=SDSQXA1-256G-GN6MA#SDSQXA1-256G-GN6MA
[63] S. Krishnan and J. L. U. Gonzalez, *Building Your Next Big Thing With Google Cloud Platform: A Guide for Developers and Enterprise Architects*. Cham, Switzerland: Springer, 2015.

SOMDIP DEY (Graduate Student Member, IEEE) was born in Kolkata, India, on December 13, 1990. He received the B.Sc. degree (Hons.) in computer science from the St. Xavier's College (Autonomous), Kolkata, in 2012, and the M.Sc. degree in advanced computer science with a specialization in computer systems engineering from The University of Manchester, U.K., in 2014.

He is currently an Artificial Intelligence Scientist working on embedded systems at the University of Essex, U.K., and a Serial Entrepreneur with a focus on social impact. He is also a Co-Founder and the CEO of Nosh Technologies, an award-winning food-tech startup offering AI-powered food management mobile app. He has more than ten years of industrial experience, working on developing technologies, including working for Microsoft and Samsung Electronics. His current research interests include affordable artificial intelligence, information security, computer systems engineering, and computing resource optimization for performance, energy, temperature, and security in mobile platforms. He has also served as a Reviewer and a TPC Member of several top conferences, such as ASAP, DATE, DAC, AAAI, CVPR, ICCV, the IEEE EdgeCom, the IEEE CSCloud, and the IEEE CSE.

AMIT KUMAR SINGH (Member, IEEE) received the B.Tech. degree in electronics engineering from IIT (Indian School of Mines) Dhanbad, Dhanbad, India, in 2006, and the Ph.D. degree from the School of Computer Engineering, Nanyang Technological University (NTU), Singapore, in 2013. He was with HCL Technologies, India, for year and a half until 2008. He has a postdoctoral research experience for over five years at several reputed universities. He is currently a Lecturer with the University of Essex, U.K. He has published over 80 articles in reputed journals/conferences. His current research interest includes system-level design-time and runtime optimizations of 2D and 3D multi-core systems for performance, energy, temperature, reliability, and security. He received several best paper awards, such as ICCES 2017, ISORC 2016, and PDP 2015. He has served on the TPC of prestigious IEEE/ACM conferences, such as DAC, DATE, CASES, and CODES+ISSS.

DILIP KUMAR PRASAD (Senior Member, IEEE) received the B.Tech. degree in computer science and engineering from IIT (ISM) Dhanbad, Dhanbad, India, in 2003, and the Ph.D. degree in computer science and engineering from Nanyang Technological University, Singapore, in 2013. He is currently an Associate Professor at UiT The Arctic University of Norway. His current research interests include image processing, machine learning, and computer vision.

KLAUS DIETER MCDONALD-MAIER (Senior Member, IEEE) is currently the Head of the Embedded and Intelligent Systems Laboratory, University of Essex, Colchester, U.K. He is also the Chief Scientist of UltraSoC Technologies Ltd., the CEO of Metrarc Ltd., and a Visiting Professor with the University of Kent. His current research interests include embedded systems and system-on-chip design, security, development support and technology, parallel and energy-efficient architectures, computer vision, data analytics, and the application of soft computing and image processing techniques for real-world problems. He is a member of the VDE and a Fellow of the IET, as well as a Fellow of the IET and the BCS.