conference_041818

Somdip Dey [1,1], Amit Kumar Singh [2], Dilip Kumar Prasad [2], and Klaus D. Mcdonald-Maier [2]

[1]University of Essex
[2]Affiliation not available

November 8, 2023

Abstract

Analyzing video for traffic categorization is an important pillar of Intelligent Transport Systems. However, it is difficult to analyze and predict traffic based on image frames because the representation of each frame may vary significantly within a short time period. This also would inaccurately represent the traffic over a longer period of time such as the case of video. We propose a novel bio-inspired methodology that integrates analysis of the previous image frames of the video to represent the analysis of the current image frame, the same way a human being analyzes the current situation based on past experience. In our proposed methodology, called IRON-MAN (Integrated Rational prediction and Motionless ANalysis), we utilize Bayesian update on top of the individual image frame analysis in the videos and this has resulted in highly accurate prediction of Temporal Motionless Analysis of the Videos (TMAV) for most of the chosen test cases. The proposed approach could be used for TMAV using Convolutional Neural Network (CNN) for applications where the number of objects in an image is the deciding factor for prediction and results also show that our proposed approach outperforms the state-of-the-art for the chosen test case. We also introduce a new metric named, Energy Consumption per Training Image (ECTI). Since, different CNN based models have different training capability and computing resource utilization, some of the models are more suitable for embedded device implementation than the others, and ECTI metric is useful to assess the suitability of using a CNN model in multi-processor systems-on-chips (MPSoCs) with a focus on energy consumption and reliability in terms of lifespan of the embedded device using these MPSoCs.

TMAV: Temporal Motionless Analysis of Video using CNN in MPSoC

Somdip Dey, *Student Member, IEEE,* Amit K. Singh, *Member, IEEE,* Dilip K. Prasad, *Member, IEEE,*
and Klaus D. McDonald-Maier, *Senior Member, IEEE*

Abstract—Analyzing video for traffic categorization is an important pillar of Intelligent Transport Systems. However, it is difficult to analyze and predict traffic based on image frames because the representation of each frame may vary significantly within a short time period. This also would inaccurately represent the traffic over a longer period of time such as the case of video. We propose a novel bio-inspired methodology that integrates analysis of the previous image frames of the video to represent the analysis of the current image frame, the same way a human being analyzes the current situation based on past experience. In our proposed methodology, called *IRON-MAN* (Integrated Rational prediction and Motionless textbfANalysis), we utilize Bayesian update on top of the individual image frame analysis in the videos and this has resulted in highly accurate prediction of Temporal Motionless Analysis of the Videos (*TMAV*) for most of the chosen test cases. The proposed approach could be used for *TMAV* using Convolutional Neural Network (CNN) for applications where the number of objects in an image is the deciding factor for prediction and results also show that our proposed approach outperforms the state-of-the-art for the chosen test case. We also introduce a new metric named, *Energy Consumption per Training Image* (*ECTI*). Since, different CNN based models have different training capability and computing resource utilization, some of the models are more suitable for embedded device implementation than the others, and *ECTI* metric is useful to assess the suitability of using a CNN model in multi-processor systems-on-chips (MPSoCs) with a focus on energy consumption and reliability in terms of lifespan of the embedded device using these 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 traffic categorization [4], [5], weather forecasting [6], [7], 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.

Traffic load categorization is challenging given the increase in vehicles on road. Some of the popular ways of monitoring and categorizing traffic load from videos include vehicle based assess method [4], [10]–[12] and holistic approach [13]–[15]. In vehicle based assess methodologies, either vehicles are first localized on the road with a background subtraction

S. Dey, A. K. Singh and K. D. McDonald-Maier are with the Embedded and Intelligent Systems Laboratory, University of Essex, UK.
E-mail: somdip.dey@essex.ac.uk
D. K. Prasad is with the Arctic University of Norway, Norway.

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

Fig. 1. Frames (Images) from the same *Light* traffic category of UCSD dataset [8] and associated prediction by a trained CNN model [9]

method [10], [16], [17] or the vehicles are localized with moving feature keypoints [11], [12], [18]. Whereas, in holistic approach, 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], [14], [19].

In recent times, there has also been emergence of several methods capable of monitoring and analyzing traffic using motionless analysis of videos [4], [9], [20], 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 [4], [9]. Moreover, to analyze 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 [9] instead of relying on a server system over th WIFI network. Another motivation to devise such approaches in embedded devices is the afford-ability of such devices instead of employing powerful server systems used for analysis purposes [9]. 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 application that requires analysis of low-frame rate

videos. Although motionless analysis of videos have their own benefits, they 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 [4], [9], [20], but in some test cases the analysis were not accurate at all. The reason for failed 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 of the image is overlapping with several other categories (labels) instead of falling under one category. For example, in the dataset of traffic released by UCSD [4], [8], 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 [9]. 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 [9] 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 fames belong to the same video under the *Light* category. If the ground truth[1] of the two images ((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 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 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. Although there have been some recent studies, which focused on future predictions of motion in ego-centric videos [21], [22], such as predicting the future position of a person based on the current image frame, but no study to our best knowledge have tried to predict the scene[3] [23], [24] of a video from image frames taking predictions from immediate previous frames into account to provide a more holistic analysis over a time period. Hence, we call such an analysis as *Temporal Motionless Analysis of Video (TMAV)* and several target applications of CNN such as traffic categorization require such kind of analysis in comparison with traditional one [4], [9], [25].

Observation 2: In the study [9], 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 also 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) [26]–[28], which implements several different types of processing elements 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 [9] is implemented and trained on a MPSoC such as Odroid XU4 [29] (more on the hardware of Odroid XU4 in Sec. V-B), 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 [30]–[32] it has been found that an increase in the operating temperature by 10-15° centigrades could reduce the lifespan of the device by 2×. An increase in operating temperature of the device could be both temporal and spatial [31], where increase in temperature over a time period is known as an increase in temporal thermal gradient and an increase in operating temperature with respect to space is called an increase in spatial thermal gradient. Additionally, an increase in energy consumption on embedded devices is harmful for battery operation itself [33], [34], especially in low-power embedded devices. Thus it is important to design not just 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.

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 makes the following contributions:

1) An energy efficient prediction methodology (*IRON-MAN*) which integrates predictions of previous 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 [29].
4) Comparative study of energy consumption using $ECTI$ metric and temporal thermal gradient on the device while

---

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

training the model on the device and off the device[4].

5) Effect on lifespan of the device utilizing CNN based approaches on such platforms

II. RELATED WORK

Majority of traffic analysis and categorization approaches before 2015 were mostly performed using the following methodologies:

- Vehicle based methodologies where either vehicles are first localized on the road with a background subtraction method [10], [16], [17] or with moving feature keypoints [11], [12]. 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. The scene is obtained by accounting for spatio-temporal features except tracking using background subtraction and moving feature keypoints [13], [14], [19].

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, Luo et al. [4] proposed a methodology to use various image processing and CNN based approaches to analyze traffic without moving features. In this paper the authors used four different visual descriptors: bag of visual words (BOVW), Vector of Locally Aggregated Descriptors (VLAD), improved Fisher Vector (IFV) and Locality-constrained Linear Coding (LLC). They have also used pre-trained deep CNN models such as Caffe and VGG to analyze traffic and predict categorization of the same. The approach taken by Luo et al. to use popular image processing and CNN methods to classify traffic is novel and solves the low-frame-rate video streaming issue. In another extended paper published by Luo et al. [20], the researchers have used SegCNN and RegCNN to analyze and classify traffic. Using SegCNN [20] the goal is to segment/predict a pixel in the image frame into 3 categories: Road, Vehicle, Background, since traffic density can be estimated by counting vehicles, road and background pixels in a traffic image. Due to SegCNN being a compute intensive and time consuming approach, instead of predicting pixel wise semantic label of an image frame, a section or patch of the image frame is predicted and this is called RegCNN [20]. In both the aforementioned works [4], [20] the authors are training and classifying traffic images after the video frames are transferred to the server from the interconnected camera network. However, installing and implementing such hardware infrastructure to analyze

[4]Training a CNN model could be performed both on the embedded device or off the device. During 'off the device training' it could be performed on a more resourceful device with higher computing resources (CPU/GPU/Memory) and then the weights and parameters of the trained CNN model are saved and transfered to the embedded system for prediction/analysis.

traffic is a challenging issue [35] due to the associated implementation costs [9]. In another study [25], Luo et al. utilizes image segmentation, Deep Learning [36] and analysis techniques with CNN to learn features from the traffic scene and categorize without motion features. The traffic analysis methodologies proposed in [4], [20], [25] are not capable of being implemented on low powered embedded devices during the training phase because such methodologies utilize a lot of computing resources due to the computational cost involved. Therefore, training of the CNN model is performed on a powerful computer with a lot of computing resources and then use the trained CNN on the device for prediction/analysis, which makes the methodologies inflexible to new training dataset. This requires the models to be trained on a powerful computing device and only be used on the embedded device for prediction/analysis.

In a study, Dey et al. [9] proposed a traffic categorization methodology using CNN where the training could be performed on the device mimicking a human being's ability to learn from its surroundings and makes such an approach more flexible in terms of learning adaptation than compared to [4], [20], [25]. Further, neither of the studies in [4], [9], [20], [25] are capable of performing temporal analysis of video, where a holistic analysis of video could be provided instead of analyzing each image frame.

Other state-of-the-art methodologies include detecting and counting the numbers of cars and computing traffic density based on that using CNN-based vehicle detectors with high accuracy at near real time [37]–[39]. This way of detecting traffic density could be classified as a vehicle based approach and has become popular in recent times but there are associated limitations [9] with these methods as follows:

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

In contrast to all the aforementioned works, we propose an easy to train CNN model, which does not require a lot of images in the training dataset. The model uses combination of transfer learning, continuous learning and Bayesian update capabilities on the multi-processor system-on-a-chip (MPSoC) platform, utilizing different types of processing elements. In transfer learning, the learning is 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 [9], whereas, in continuous learning, the learning is achieved by re-training the classifier with wrong predictions till operating period of the system [9]. Bayesian update is the ability to update the probability for a hypothesis as more new information or evidence becomes available [40].

III. SYSTEM AND PROBLEM FORMULATION

A. System Description

Multi-processor systems-on- chips (MPSoCs) consisting of different types of cores such as CPU and/or GPU. These architectures provide opportunities to exploit distinct features of CPU and/or GPU cores to meet performance and energy consumption requirements. One of the most popular MPSoCs is the Samsung Exynos 5422 SoC [41], which is an excellent playground for demanding computer vision applications due to its powerful processing elements (see Sec. V-B for more details on the hardware setup for the experiments performed). Since most recent computer vision applications such as image recognition with CNN [42], consist of task and thread level parallelism as well as data level parallelism [43], [44], such applications are really good candidates for utilizing the full computing capacity of MPSoCs such of the Exynos 5422. Moreover, nowadays MPSoCs are used extensively in smart phone devices such as the Exynos 5422 MPSoC is utilized in several of the Samsung Note and Galaxy smart phones. Hence, implementing our proposed methodology to solve *TMAV* on an MPSoC is the focus of our paper.

B. Problem Definition

If we consider a video (V) consists of m number of image frames (f) such that $V = \{f_1, f_2, ...f_i, ...f_m\}$, instead of using CNN to analyze and predict (P_i) an instance of the image frames (f_i) of the video such that $P_i = PredictionOf(f_i)$ (where $PredictionOf(f_i)$ is a function to evaluate prediction of f_i), our goal is to analyze and predict (P) n number of image frames out of m number of frames of the video in a holistic manner, where $n \leq m$. Here, by the term 'holistic' we try to provide the overall analysis and prediction of n image frames rather than just providing the prediction of one instance (f_i) of the image frames. Therefore, in this paper we tackle the problem as defined below:

Given a video (V) consisting of m number of image frames and F is a set of images frames consisting of n frames of V such that $F = \{f_i, f_{i+1}, ...f_{i+n-1}\}$

Compute holistic prediction (P) of F such that $P = PredictionOf(F)$

Subject to $n \leq m$.

IV. PROPOSED METHODOLOGY: IRON-MAN

In our proposed approach, we utilize the concept of *Hybrid Training Method* [9], 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*[5] [45], [46] by utilizing an existing pre-trained network and train

---

[5]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

---

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) [9], then we utilize the CNN for prediction in the prediction module. However, if the desired prediction accuracy is not achieved then we *retrain* the CNN with the failed predicted images. This *retrain* methodology is bio-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. IV-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 account. Our CNN model's prediction is inspired by *Bayes' theorem* and *sequential Bayesian updating* [40], where the model updates the possibility of the prediction label occurring by taking the probability of the label occurring in the previous frames. This approach is again bio-inspired as it is adopted from the ideology of humans updating their knowledge using Bayesian inference logic. Detailed algorithm and inner working of the IRON-MAN is provided in the following two subsections.
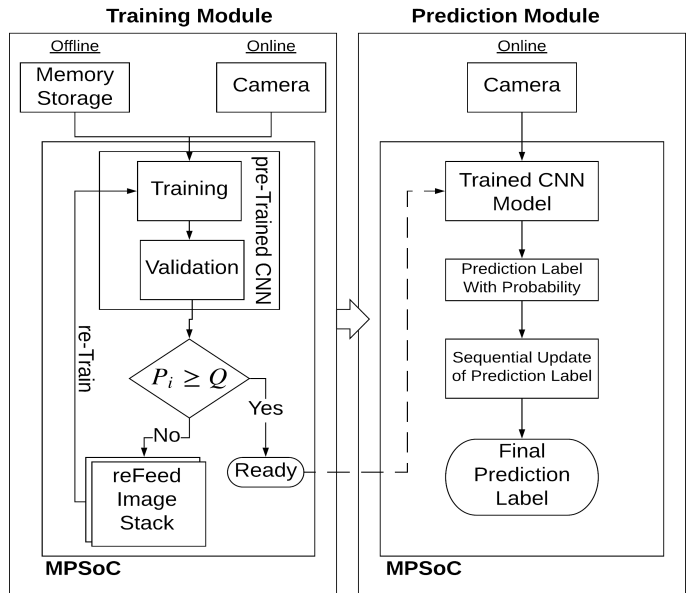

Fig. 2. IRON-MAN Model Work-flow

A. Training Module

The algorithm for the Training (Learning) module is provided in Algo. 1. For the proposed approach, any pre-trained CNN model such as VGG [3], ResNet [47], etc could be selected. 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. Upon failure in prediction for each image, the image is stored in a stack implementation called "*reFeed Image Stack*". After utilizing cross-validation technique [48], 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 [9], 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* (using the function S.Push(i) as mentioned in Algo. 1, where S is 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. The retrain mechanism is bound to improve prediction accuracy because we train the CNN model with the failed images[6] saved in the reFeed image stack and this approach mimics a human being's ability to rectify his/her mistake after making one.

$$\forall\{i \in I : i > 1\}, P_i \geq Q \qquad (1)$$

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.

---

[6]Here, failed images are the image frames which were predicted/labeled incorrectly during the testing of the trained CNN model.

Algorithm 1: Training Module Execution

Input:
1. $\mathcal{I}$: set of n Images from training & validation dataset
2. $\mathcal{C}$: set of m Images from camera (for cross-validation)
Output: P: prediction accuracy after training
Initialize: $Q = 0.7$; ▷ Quality of experience is set to 70% by default
$\mathcal{S}.Count = 0$;   ▷ $\mathcal{S}$: *reFeed image stack*

Offline Training:
Train (pre-trained CNN model , $\mathcal{I}$);   ▷ Train model with $\mathcal{I}$ dataset

Online Training:
for *each image $i \in \mathcal{C}$* **do**
    Prediction = Test (CNN model); ▷ Test(CNN model) is a function which outputs whether prediction is correct or wrong
    ▷ Prediction.IsWrong() is a function to return True when Prediction.Label ! = Original.Label of test image i
    if *Prediction.IsWrong()* **then**
        $S.Push(i)$;
    P_i = CalculatePredictionAccuracy();
        ▷ CalculatePredictionAccuracy() is the function to evaluate the overall prediction accuracy of the CNN and return the value (P_i)
{*re-Train with reFeed Image Stack if $P_i < Q$*}
if $P_i < Q$ **then**
        ▷ Need to satisfy condition of Eq. 1
    if *S.Empty() == False* **then**
        {*Traing CNN with reFeed image stack*}
        Train (CNN model , $\mathcal{S}$);
        P_i^f = CalculatePredictionAccuracy();
        ▷ CalculateMeanPredictionAccuracy() is the function to evaluate the overall prediction accuracy of the CNN after retraining and return the value (P_i^f)
    $P = P_i^f$;
    $\mathcal{S}.Count = 0$;   ▷ reset *reFeed image stack*
else
    return P;

B. Prediction Module

The detailed algorithm for the Prediction module implementation is provided in Algo. 2. From Bayes' Theorem [40] we can represent the expression representing the *Bayesian Update Scheme* as follows:

$$posterior \propto prior \times likelihood \qquad (2)$$

In Eq. 2, *posterior* is the revised probability of an event occurring after taking new information into consideration, *prior* is the probability of the event assessed before revising posterior and *likelihood* is the probability that an event which has already occurred would yield a specific outcome. Now, using sequential update scheme where we take the past into account and the modified expression for Bayesian Update Scheme is as follows:

$$new\ posterior \propto current \times new\ likelihood \qquad (3)$$

In Eq. 3, *current* is the probability of some entity occurring whereas the *new likelihood* 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* 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 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:

$$P_{updated}^{category} = \begin{cases} P_{this}^{category}, & \text{if } N = 0 \\[2ex] \frac{P_{this-1}^{category} \times P_{this}^{category}}{(P_{this-1}^{category} \times P_{this}^{category}) + P_{CNN}}, & \text{if } N = 1 \\[2ex] \prod_{1}^{N} \frac{P_{this-N}^{category} \times P_{this-(N-1)}^{category}}{(P_{this-N}^{category} \times P_{this-(N-1)}^{category}) + P_{CNN}}, & \text{if } N > 1 \end{cases} \tag{6}$$

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 1st position of the *Image Queue* is popped out and the $(N+1)^{th}$ image frame is pushed in the Nth 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 the updated prediction of other categories and the label for the maximum value of the prediction is provided as output.

Algorithm 2: Prediction Module Execution

Input:
$\mathcal{N}$: number of frames in the *frame window*
Output:
1. P_{label}: prediction label
2. $P_{updated}^{category}$: updated prediction label using Eq. 6
Initialize: $\mathcal{I}.Count = 0$;   ▷ $\mathcal{I}$: *Image queue of size N*

while *Camera is feeding image frames* **do**
    if $\mathcal{I}.Count > 0$ **then**
        $\mathcal{I}.Pop()$;
        ▷ Pop() is a function to remove the object/image in the 1st position of the *image queue*
    $\mathcal{I}.Push(Current Frame)$;
    ▷ Push(Image) is a function to remove the object/image in the Nth position of the *Image Queue*

    P_{label} = PredictCurrentFrame();
    ▷ PredictCurrentFrame() is a function to predict the label (category) for the current frame

    $P_{updated}^{category}$ = UpdatedPrediction(N);
    ▷ UpdatedPrediction(N) is a function to predict the label taking prior N sequential frames into account by utlizing the Eq. 6 for all available categories and retunring the label for the category with higgest probability

---

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 $ECTI$.

V. EXPERIMENTAL RESULTS

A. Dataset Used

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

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	**(Label)** │ **Empty** │ **Fluid** │ **Heavy** │ **Jam** (Fluid) │ 0.01 │ **0.793** │ 0.1683 │ 0.0286
	(Heavy) │ 0.0131 │ 0.3091 │ **0.5098** │ 0.1681	(Fluid) │ 0.0001 │ **0.1986** │ 0.0798 │ 0.0048
	(Jam) │ 0.0131 │ 0.2754 │ 0.3399 │ **0.3717**	(Fluid) │ 0.0 │ **0.0524** │ 0.0267 │ 0.0018

TABLE I

PREDICTIONS OF TRAFFIC CATEGORIZATION 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**	**(Label)** │ **No-Obstruction** │ **Obstruction** (Obstruction) │ 0.3270 │ **0.6730**
	(No-Obstruction) │ **0.5010** │ 0.4990	(Obstruction) │ 0.1843 │ **0.3166**
	(Obstruction) │ 0.4600 │ **0.5400**	(Obstruction) │ 0.1047 │ **0.1908**

TABLE II

PREDICTIONS OF PEDESTRIAN OBSTRUCTION IMAGE FRAMES WITH AND WITHOUT IRON-MAN

1) Traffic categorization dataset: For our *traffic categorization* experimentation we are using the same dataset used by Luo et al. [4], [20] and Dey et al. [9] for traffic categorization to validate the performance of our proposed methodology. Mainly two dataset are used in this experiment. The first dataset is released by UCSD traffic control department [8]. 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*. Another dataset consisting of the 400 images, which is provided by

Luo et al. [4] as well as Dey et al. [9], 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 segregated into 4 categories: *Jam, Heavy, Fluid, Empty*; and each category having 100 images. This dataset was used for training and validation purposes. 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.

2) Pedestrian obstruction dataset: For our *pedestrian obstruction* experimentation we have used the pedestrian dataset used by Li et al. [49], 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 validate our proposed IRON-MAN approach we have only used the training dataset of [49] consisting of 9742 images for both training and validation, whereas we used the validation dataset 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 [29], which employs Exynos 5422 MPSoC [41] 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 [50] with networking capabilities over WIFI to take power consumption readings. Odroid-XU4 platform has 4 temperature sensors on 4 ARM Cortex A-15 big CPU cores, which we have read to monitor temperature behavior during our experiments. For all our experiments we have only utilized 4 A-15 big CPU cores to monitor the operating temperature and power consumption.

C. Experimental Results

1) Prediction accuracy evaluation: We chose two pre-Trained CNN models, which were trained on millions of ImageNet images, for our validation. These two CNN models are VGG16 [3] and ResNet50 [47]. In our experiments, we have chosen the *quality of experience* (Q) to be 0.7 i.e. 70%. For VGG16 it took us 360 images to train the pre-Trained CNN for traffic categorization using Transfer Learning and retrain approaches, (see Sec. IV-A) and gained a testing prediction accuracy of 98.93%. The **average power consumption was 10.63 W** on the Odroid-XU4 MPSoC and the total execution time was 97 minutes 44 seconds. Therefore, using the formula in Eq. 7, the *Energy Consumption per Training Image* comes out to be $48.097 \times 10^{-6} \ kWh$ (average) ($\simeq (10.63 \times \frac{1}{1000}) \times \frac{(97 \times 60 + 44)}{3600} \times \frac{1}{360}$).

For ResNet50 it took us 117 minutes and 27 seconds, and 330 images to train the pre-Trained CNN for traffic categorization using transfer learning and retrain approaches. The testing prediction accuracy attained by the model is 92.79%. The **average power consumption was 10.59 W**. Therefore, using Eq. 7, the *Energy Consumption per Training Image* comes out to be $62.817 \times 10^{-6} \ kWh$ (average) ($\simeq (10.59 \times \frac{1}{1000}) \times \frac{(117 \times 60 + 27)}{3600} \times \frac{1}{330}$). Since both VGG16 and ResNet50 were able to achieve a prediction accuracy higher than Q (70%) and the ECTI of VGG16 is lower than ResNet50, we have chosen VGG16 as a more suitable CNN model for training on the embedded device, i.e. Odroid-XU4. During training VGG16 and ResNet50, the average maximum temperature achieved by the big CPU cores are 93.60°C (average) and 93.72°C (average) respectively. The maximum operating temperature of the big CPU cores for different CNN models are shown graphically in Fig. 4. We 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 and 24.48 °C (average) for ResNet50. Max T. and Max Baseline T. stands for maximum operating temperature and maximum baseline temperature in Fig. 4. In Fig. 5 we show graphical representation of power consumption for both VGG16 and ResNet50 training period. For both Fig. 4 and Fig. 5 we have only shown a snapshot of the total training execution period due to repetition in behavior of the graph.

Now, to prove efficacy of our integrated rational prediction of image frames from the video, we randomly chose a video from UCSD traffic dataset [8] 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 category and utilized the VGG16 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 the experiment we chose the *Frame Window* of 3 images representing a video sequence of 4 seconds (approx.). Table. I shows the prediction of labels if we only use VGG16 CNN without our IRON-MAN approach and if we use IRON-MAN as well. The table shows that VGG16 could predict the label of the image frame, but that prediction did not match the label for the whole video, whereas the prediction label for the video using IRON-MAN was achieved with 100% accuracy.

In another experiment (*pedestrian obstruction*), we chose simultaneous image frames having pedestrians as obstruction from the pedestrian dataset [49] to validate efficacy of IRON-MAN. Table II 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. ResNet50 CNN was utilized in this experiment, which gained a testing prediction accuracy of 72.50% after *transfer learning* and *retraining* approaches for this application. For pedestrian obstruction experiment we trained the classifier of ResNet50 and VGG16

on a general purpose computer instead due to massive training dataset and associated energy consumption, and migrated the saved parameters and weights of the model to Odroid-XU4 for further prediction. ResNet50 was selected over VGG16 in this experiment because the model gained a better testing prediction accuracy for this application, which outperformed VGG16's 66.87% testing prediction accuracy for the same.

2) Comparison between traditional method & IRON-MAN: If we consider a video consisting of M number of image frames need to be analyzed and predicted using CNN based approaches such as VGG and ResNet, traditional method such as VGG or ResNet would analyze each individual image frames from M number of images and output the result. Whereas, from the results of experiments provided in Sec. V-C1, we could notice that *IRON-MAN* is capable of analyzing N, where $N \leq M$, number of image frames together and provides a consolidated prediction of N frames. Therefore, *IRON-MAN* is able to categorize video with 100% prediction accuracy during testing and is able to perform *TMAV*, which could not be performed by traditional CNN based approaches.

3) 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. [25]. Since, the methodology proposed in [25] is closely related to the target application of traffic categorzation without motion features, it is justifiable to use the proposed methodology for a comparative study with *IRON-MAN*. Additionally, *IRON-MAN* being the first methodology to perform *TMAV* in traffic categorization, it would be unjustified to compare the methodology with any other existing approaches.

In [25], 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.

Although 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 slightly higher ($\simeq 2.01\%$) than the prediction accuracy achieved by *IRON-MAN*, but when the approach was tested for temporal analysis of image frames to provide holistic prediction of the video instead, it achieved a prediction accuracy of 65% for the same target application. In comparison *IRON-MAN* was able to achieve 100% prediction accuracy for temporal analysis of 9 sequential image frames in the *frame window* as mentioned in Sec. V-C.

Moreover, given the computation complexity of [25], executing the approach also has an overhead of $5.2\times$ on the Odroid XU4 MPSoC than compared to *IRON-MAN*. Fig. 3 shows the execution time (in seconds) of analyzing 9 sequential image frames of the same video from the UCSD dataset [8]. Each test was performed 5 times on the Odroid XU4, utlizing 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. 3. Since the concept of *frame window* or temporal analysis of image frames of video does not exist in [25] approach, we sequentially feed each image
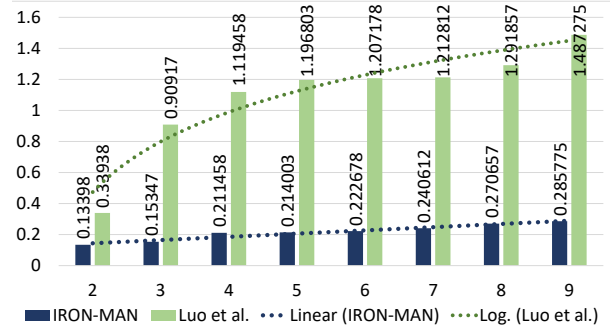

Fig. 3. Execution time taken to analyze 9 image frames sequentially from the traffic video (Execution time in seconds vs number of image frames analyzed)
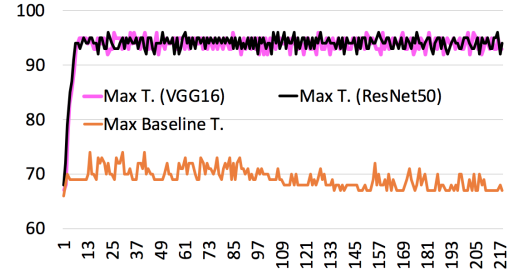

Fig. 4. Graphical representation of operating temperature peaks of VGG16 and ResNet50 training and baseline temperature (°C vs time interval)

frames from the video to the methodology to get the output analysis.

From Fig. 3 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 $5.2\times$ (approx.) on the Odroid XU4 MPSoC, making the *IRON-MAN* more suitable for real-time execution for traffic categorization application on the embedded devices utilizing MPSoC.

4) Effect on lifespan of the device: In the studies [30]– [32], it has been shown that an increase in the operating temperature of an embedded device by 10-15°C could reduce the lifespan of the device by $2\times$. In Sec. V-C and from the knowledge on thermal deviation in Fig. 4, we have already noticed that if training is performed on the device then there is an increase in operating temperature of the device by 24.36°C (average) for VGG16 and 24.48°C (average) for ResNet50. Therefore, considering that the device lifespan reduces by $2\times$ for every 10°C increase in operating temperature and if training is performed on the device then the lifespan of the same device reduces by $4.872\times$ ($\simeq \frac{24.36}{10} \times 2$) for utilizing VGG16 and $4.896\times$ ($\simeq \frac{24.48}{10} \times 2$) for utilizing ResNet50. Now, if we consider that lifespan reduces by $2\times$ for an increase of 15°C in operating temperature instead, then the same evaluation changes to $3.248\times$ for VGG16 and $3.264\times$ for ResNet50. 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.

VI. LIMITATIONS AND DISCUSSION

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

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

$$P_{updated}^{category} \longrightarrow 0 \text{ as } P_{this}^{category} \longrightarrow 0 \qquad (8)$$

Theorem 1. *If $N \geq 1$ in the frame window and $P_{this}^{category} \longrightarrow 0$ then the updated prediction ($P_{updated}^{category}$) 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.

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

In the Eq. 9, if we consider the term, $\frac{P_{this-N}^{category} \times P_{this-(N-1)}^{category}}{(P_{this-N}^{category} \times P_{this-(N-1)}^{category}) + P_{CNN}}$ as P_k, since the preceding term is part of the product series represented in Eq. 9 and consider $P_{updated}^{category}$ 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:

$$F_{N,i} = \prod_{k=N-i+1}^{N} P_k \qquad (10)$$

Since $P_{this}^{category}$ is an integral part of P_k in Eq. 10 (see Eq. 9), using the knowledge from *lemma 1* we can say that as $P_{this}^{category} \longrightarrow 0$ 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 $P_{this}^{category}$ gets smaller (close to zero), the updated prediction ($P_{updated}^{category}$) 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

---

[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.
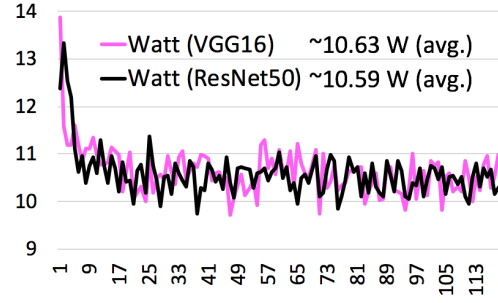

Fig. 5. Power consumption for VGG16 and ResNet50 in Watts (Watt vs time interval)

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. V-A) interval then using IRON-MAN we are able to analyze 9.1 (1.3×7) seconds of the video with high prediction accuracy.

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.

VII. 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. Our approach is able to achieve 100% prediction accuracy in analyzing video from the image frames for certain applications and it was also noticed that for our chosen applications VGG16 had better energy-efficiency on Exynos 5422 platform compared to ResNet50 using *Energy Consumption per Training Image* ($ECTI$) metric comparison. Based on the results we have also shown that training CNN based approaches on MPSoCs could lead up to $4.8\times$ (approx.) reduction in lifespan of the embedded device and therefore, it is recommended to perform the training off the embedded device for improved longevity. It is also shown that for traffic categorization application, our approach outperforms the state-of-the-art.

REFERENCES

[1] S. Chakradhar *et al.*, "A dynamically configurable coprocessor for convolutional neural networks," in *ACM SIGARCH Comp. Arch. News*, vol. 38, no. 3. ACM, 2010, pp. 247–257.
[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," *arXiv preprint arXiv:1409.1556*, 2014.

[4] Z. Luo *et al.*, "Traffic analysis without motion features," in *2015 IEE ICIP*. IEEE, 2015, pp. 3290–3294.

[5] G. Kalliatakis *et al.*, "Detection of human rights violations in images: Can convolutional neural networks help?" *Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP*, 2017.

[6] G. Zhang *et al.*, "Forecasting with artificial neural networks:: The state of the art," *International journal of forecasting*, vol. 14, no. 1, pp. 35–62, 1998.

[7] M. Grecu *et al.*, "Detection of anomalous propagation echoes in weather radar data using neural networks," *IEEE Tran. on geoscience and remote sensing*, vol. 37, no. 1, pp. 287–296, 1999.

[8] A. B. Chan and N. Vasconcelos, "Classification and retrieval of traffic video using auto-regressive stochastic processes," in *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*. IEEE, 2005, pp. 771–776.

[9] S. Dey *et al.*, "Mat-cnn-sopc: Motionless analysis of traffic using convolutional neural networks on system-on-a-programmable-chip," *NASA/ESA AHS*, 2018.

[10] O. Asmaa *et al.*, "Road traffic density estimation using microscopic and macroscopic parameters," *Image and Vision Computing*, vol. 31, no. 11, pp. 887–894, 2013.

[11] S. Hu, J. Wu, and L. Xu, "Real-time traffic congestion detection based on video analysis," *JOURNAL OF INFORMATION &COMPUTATIONAL SCIENCE*, vol. 9, no. 10, pp. 2907–2914, 2012.

[12] A. Riaz and S. A. Khan, "Traffic congestion classification using motion vector statistical features," in *Sixth International Conference on Machine Vision (ICMV 2013)*, vol. 9067. International Society for ics and Photonics, 2013, p. 90671A.

[13] K. G. Derpanis and R. P. Wildes, "Classification of traffic video based on a spatiotemporal orientation analysis," in *Applications of Computer Vision (WACV), 2011 IEEE Workshop on*. IEEE, 2011, pp. 606–613.

[14] F. Porikli and X. Li, "Traffic congestion estimation using hmm models without vehicle tracking," in *Intelligent Vehicles Symposium, 2004 IEEE*. IEEE, 2004, pp. 188–193.

[15] A. Ess *et al.*, "Segmentation-based urban traffic scene understanding." in *BMVC*, vol. 1. Citeseer, 2009, p. 2.

[16] Y.-K. Jung, K.-W. Lee, and Y.-S. Ho, "Content-based event retrieval using semantic scene interpretation for automated traffic surveillance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 2, no. 3, pp. 151–163, 2001.

[17] L. O. Andrews Sobral, L. Schnitman, and F. De Souza, "Highway traffic congestion classification using holistic properties," in *10th IASTED International Conference on Signal Processing, Pattern Recognition and Applications*, 2013.

[18] S. Kamijo, Y. Matsushita, K. Ikeuchi, and M. Sakauchi, "Traffic monitoring and accident detection at intersections," *IEEE transactions on Intelligent transportation systems*, vol. 1, no. 2, pp. 108–118, 2000.

[19] A. B. Chan and N. Vasconcelos, "Classification and retrieval of traffic video using auto-regressive stochastic processes," in *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*. IEEE, 2005, pp. 771–776.

[20] Z. Luo *et al.*, "Traffic analytics with low frame rate videos," *IEEE Tran. Circ. and Sys. for Video Technology*, 2016.

[21] T. Yagi *et al.*, "Future person localization in first-person videos," *arXiv preprint arXiv:1711.11217*, 2017.

[22] G.-Y. Lai *et al.*, "People trajectory forecasting and collision avoidance in first-person viewpoint," in *2018 IEEE ICCE-TW*. IEEE, 2018, pp. 1–2.

[23] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Advances in neural information processing systems*, 2014, pp. 487–495.

[24] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "Sun database: Large-scale scene recognition from abbey to zoo," in *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*. IEEE, 2010, pp. 3485–3492.

[25] Z. Luo, P.-M. Jodoin, S.-Z. Su, S.-Z. Li, and H. Larochelle, "Traffic analytics with low-frame-rate videos," *IEEE TCSVT*, vol. 28, no. 4, pp. 878–891, 2018.

[26] 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 Transactions on Embedded Computing Systems (TECS)*, vol. 16, no. 5s, p. 147, 2017.

[27] U. Gupta, C. A. Patil, G. Bhat, P. Mishra, and U. Y. Ogras, "Dypo: Dynamic pareto-imal configuration selection for heterogeneous mpsocs,"

[28] S. Dey, E. Z. Guajardo, K. R. Basireddy, X. Wang, A. K. Singh, and K. D. McDonald-Maier, "Edgecoolingmode: An agent based thermal management mechanism for dvfs enabled heterogeneous mpsocs," in *VLSI Design and 18th International Conference on Embedded Systems, 2019. 32nd International Conference on*. IEEE, 2019.

[29] "Odroid-xu4," https://goo.gl/KmHZRG, accessed: 2018-07-23.

[30] T. Chantem *et al.*, "Temperature-aware scheduling and assignment for hard real-time applications on mpsocs," in *DATE*. ACM, 2008, pp. 288–293.

[31] A. K. Coskun *et al.*, "Temperature aware task scheduling in mpsocs," in *DATE*. IEEE, 2007, pp. 1–6.

[32] J. Zhou *et al.*, "Thermal-aware task scheduling for energy minimization in heterogeneous real-time mpsoc systems," *IEEE TCAD*, vol. 35, no. 8, pp. 1269–1282, 2016.

[33] D. Panigrahi, C. Chiasserini, S. Dey, R. Rao, A. Raghunathan, and K. Lahiri, "Battery life estimation of mobile embedded systems," in *VLSI Design, 2001. Fourteenth International Conference on*. IEEE, 2001, pp. 57–63.

[34] K. Lahiri, S. Dey, D. Panigrahi, and A. Raghunathan, "Battery-driven system design: A new frontier in low power design," in *Proceedings of the 2002 Asia and South Pacific Design Automation Conference*. IEEE Computer Society, 2002, p. 261.

[35] B. S. Mokha and S. Kumar, "A review of computer vision system for the vehicle identification and classification from online and offline videos," *An International Journal on Signal and Image Processing*, vol. 6, pp. 63–76, 2015.

[36] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[37] W. Zhang, L. Chen, W. Gong, Z. Li, Q. Lu, and S. Yang, "An integrated approach for vehicle detection and type recognition," in *Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom), 2015 IEEE 12th Intl Conf on*. IEEE, 2015, pp. 798–801.

[38] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos, "A unified multi-scale deep convolutional neural network for fast object detection," in *European Conference on Computer Vision*. Springer, 2016, pp. 354–370.

[39] 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 *Digital Signal Processing (DSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 276–280.

[40] A. Doucet *et al.*, "On sequential monte carlo sampling methods for bayesian filtering," *Statistics and computing*, vol. 10, no. 3, pp. 197–208, 2000.

[41] "Exynos 5 octa (5422)," https://www.samsung.com/exynos, accessed: 2018-07-23.

[42] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[43] N. Sundaram, "Making computer vision computationally efficient," Ph.D. dissertation, UC Berkeley, 2012.

[44] J. Y. Mori and M. Hübner, "Multi-level parallelism analysis and system-level simulation for many-core vision processor design," in *2016 5th Mediterranean Conference on Embedded Computing (MECO)*. IEEE, 2016, pp. 90–95.

[45] S. J. Pan *et al.*, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[46] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[47] K. He *et al.*, "Deep residual learning for image recognition," in *IEEE CVPR*, 2016, pp. 770–778.

[48] S. Arlot, A. Celisse *et al.*, "A survey of cross-validation procedures for model selection," *Statistics surveys*, vol. 4, pp. 40–79, 2010.

[49] X. Li *et al.*, "A new benchmark for vision-based cyclist detection," in *Intelligent Vehicles Symposium (IV), 2016 IEEE*. IEEE, 2016, pp. 1028–1033.

[50] "Odroid smartpower2," https://www.hardkernel.com/main/products/prdt_info.php\?g_code=G148048570542, accessed: 2018-07-23.