MAT-CNN-SOPC: Motionless Analysis of Traffic Using Convolutional Neural Networks on System-On-a-Programmable-Chip

Somdip Dey, Grigorios Kalliatakis, Sangeet Saha, Amit Kumar Singh, Shoaib Ehsan, Klaus McDonald-Maier
Embedded and Intelligent Systems Laboratory
University of Essex
Colchester, UK
{ somdip.dey, gkallia, sangeet.saha, a.k.singh, sehsan, kdm } @essex.ac.uk

***Abstract*—Intelligent Transportation Systems (ITS) have become an important pillar in modern "smart city" framework which demands intelligent involvement of machines. Traffic load recognition can be categorized as an important and challenging issue for such systems. Recently, Convolutional Neural Network (CNN) models have drawn considerable amount of interest in many areas such as weather classification, human rights violation detection through images, due to its accurate prediction capabilities. This work tackles real-life traffic load recognition problem on System-On-a-Programmable-Chip (SOPC) platform and coin it as MAT-CNN-SOPC, which uses an intelligent re-training mechanism of the CNN with known environments. The proposed methodology is capable of enhancing the efficacy of the approach by 2.44x in comparison to the state-of-art and proven through experimental analysis. We have also introduced a mathematical equation, which is capable of quantifying the suitability of using different CNN models over the other for a particular application based implementation.**

***Index Terms*—Convolutional neural network (CNN), traffic analysis, traffic density, transfer learning, system-on-a-programmable-chip (SOPC).**

I. Introduction

Some of the popular ways of traffic monitoring and analysis for categorization of traffic load is either using vehicle based assess method [1]–[8] or a holistic approach [9]–[12]. But analysis of traffic using these popular methods require high frame rate videos with a stable environmental condition, which could be the biggest limiting factor in many places. Without these conditions being met [8], [13]–[15], reliable motion features cannot be extracted, which might result in corrupted output.

Because of large-scale camera network not being able to stream and store high-frame rate videos gathered by a network of interconnected cameras due to bandwidth limitation and limited on-board storage capacity, streaming low-frame videos on these camera is very common. In many cases when these cameras stream over a WIFI network, it is often difficult to stream more than 2 frames per second due to the limited bandwidth of the network [8], [14]. Moreover due to cost constraint of such interconnected camera networks and associated servers, many developing countries might not be able to adopt and implement such sophisticated state-of-the-art traffic analysis and categorization methodologies. On the other hand image processing [16]–[23] and computer vision applications [16], [24], [25] are very well known for their thread, task and data level parallelism. Recently we could also notice a huge increase in integrating Convolutional Neural Networks [26]–[30] in computer vision to solve several real-life challenges such as human rights violation detection through images [31], [32], weather forecasting [33], [34], etc. Due to high level of data parallelism in computer vision applications using Convolutional Neural Networks and reducing cost factor of field-programmable gate array (FPGA) based system-on-a-programmable-chip (SOPC) [35], [36], such SOPC serves as a cost-effective option to analyze and categorize traffic.

In this paper, we propose a novel methodology to analyze and categorize traffic using Convolutional Neural Networks on SOPC without the need of streaming the video-frames to the server for further categorization as is usually done in state-of-the-art traffic categorization methodologies. The proposed methodology is coined as Motionless Analysis of Traffic Using Convolutional Neural Networks on SOPC: MAT-CNN-SOPC and we have also introduced a *Quality of Experience* variable, which would enhance the predicting mechanism of the chosen CNN model. The remainder of this paper is organized as follows. Section II mentions the related work in the field and Section III provides a breakdown of the software and hardware infrastructure used for the implementation and validation purposes of the proposed methodology along with the dataset used and problem definition solved using this solution. Section IV provides a comprehensive view of the proposed methodology and in Section V we could analyze the experimental results. Section VI briefly mentions some related discussion on the proposed methodology. Finally, Section VII concludes the paper.

This work is supported by the UK Engineering and Physical Sciences Research Council EPSRC [EP/R02572X/1 and EP/P017487/1].

978-1-5386-7753-7/18/$31.00 2018 IEEE

II. 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 [3]–[5] or the vehicles are localized with moving feature keypoints [6], [7]. 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 [9]–[11].

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. [8] 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 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. 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. However, the experimental setups and results provided in the paper is susceptible to some biasness as the cross-dataset validation was not performed. In Section VI we have compared our experimental setup and achieved results with the ones mentioned in [8]. In another extended paper published by Luo et al. [14], the researchers have used SegCNN and RegCNN to analyze and classify traffic. In both the aforementioned papers the authors are training and classifying traffic images after the video frames are transferred to the server from the interconnected camera network. But installing and implementing such hardware infrastructure to analyze traffic in developing countries is a challenging issue [37].

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 [38]–[40]. 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 [41] could solve the problem of detecting occluded objects properly but such method usually requires large dataset to be trained with. But for the application of traffic categorization there is no such publicly available dataset and hence using Deep Learning would be inefficient.

Compared to all the aforementioned works, we propose an easy to train CNN model, which do not require a lot of images in the training dataset, with combination of transfer learning [1] and continuous learning [2] capabilities on SOPC without the need of communicating the traffic images to the connected server for further analysis.

III. System and Problem Formulation

A. Hardware Infrastructure & Software Infrastructure

It is worth mentioning that the CNN based traffic analysis will demand a huge amount of computing resources. Rather than high performance general purpose processing unit, the application specific computing could also be a lucrative way out. From the recent literature studies [41], [42], it has been observed that software based execution could provide the required flexibility but not the performance efficiency in terms of execution. On the other hand, a dedicated hardware based execution will provide performance efficacy but will under perform when the flexibility becomes the major concern.

Thus, hardware software co-execution ecosystem is emerging as a bright prospect and modern FPGA (ZYNQ) platform is a good solution to implement such functionality [3]. In order to carry out the functionality in FPGA, we have chosen the vivado HLS [43] framework. This framework also extracts the parallelism inside the code. The entire CNN model is created in high level language (C/C++, Matlab, Python). Then it has been converted in to RTL [4] format through vivado high level synthesis. Once the code has been converted, the VIVADO framework will synthesize to the bitstream to make the design executable. Our code (in Matlab, Python & C/C++) is provided on our GitHub repository [44].

[1]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

[2]Learning achieved by re-training the classifier with wrong predictions till operating period of the system

[3]Even though GPUs could be an efficient accelerators for CNNs. However, such devices are expensive & very power hungry and thus, make them not suitable in the aforementioned power-constrained scenarios

[4] RTL: Register-transfer level is a design abstraction, which models a synchronous digital circuit in terms of the flow of digital data between hardware registers and the logical operations performed on those signals.

B. Dataset

For our research we are using the same dataset used by Luo et al. [8], [14] to validate performance of our proposed methodology and theories. Mainly two dataset are used. The first one is the dataset released by UCSD traffic control department [45]. This dataset contains 254 highway video sequences, all filmed by the same camera containing light, heavy and traffic jams filmed at different periods of the day and under different weather conditions. Each UCSD video has a resolution of 320 X 240 with a frame rate of 10 fps.

The second dataset consist of the 400 images[5] captured from highway cameras deployed in 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 (as shown in Fig. 1), and each category having 100 images.

Fig. 1. Random images from 4 Categories of Traffic Classification: Jam, Heavy, Fluid, Empty [8]

C. Problem Definition

The main focus of this research is to be able to implement a hardware-software ecosystem, which is able to analyze and predict traffic effectively on the System-on-programmable-chip without streaming the video-frames to the server over a communication channel even in severe hardware impaired conditions such as poor video recording capabilities of the camera. Since a practical application such as categorization of traffic using CNNs methodologies requires a desirable "Quality of Experience" (*QoE*) in order to be a successful implementation, we also need to define the governing equation to quantify *QoE* so that we could understand the overall desirability of the CovNet methodology being used for the problem in hand. Let us consider the (*QoE*) that will decide whether the accuracy of the CovNet methodology is desirable as Q and the predicted label (categorization) of the CovNet as P_i for any image (i) from a dataset of images (I) at an instance.

[5]Only 400 images were available in the existing dataset provided by Luo et al. [8]

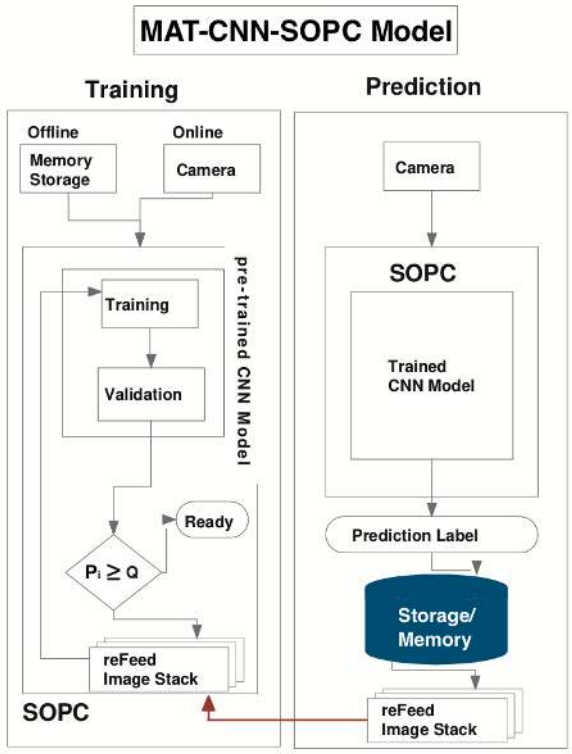


Fig. 2. MAT-CNN-SOPC Model Work flow

Then the governing equation which could be used to predict the label (category) of the traffic as desirable at an instance is as follows:

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

In the aforementioned equation (1), we are not taking the training time of the CovNet model into consideration as part of *QoE* since it is assumed that training is mandatory and completed while the hardware-software ecosystem is setup on the section of the road or highway for the purpose of categorizing traffic. Later in Section VI we would also provide a minimum threshold value for *QoE* for the given problem in hand based on the experimental results (Section V) performed.

IV. Proposed Methodology: MAT-CNN-SOPC

In this section we propose the hardware-software ecosystem, MAT-CNN-SOPC, which would be utilizing the categorization power of a pre-trained CNN model to be trained to effectively categorize traffic based on the desired categories. We propose a two fold module of MAT-CNN-SOPC: *Training* & *Prediction* (as shown in Fig. 2). Both the *Training* and *Prediction* modules are implemented in application layer of the SOPC. For this hardware-software ecosystem we assume that a camera is connected to the system-on-a-programmable-chip and the primary training of the classifier of the pre-trained CNN model is performed [6] while the SOPC is setup on the section of the road in the first place.

[6]Using transfer learning of pre-trained CNN model

For our proposed model we could select any available pre-trained CNN model such as AlexNet [29], VGG [28], ResNet [30], etc. for the *Training* module. In this module we train the system with various known images of traffic. Since FPGA on the SOPC are excellent candidates for SIMD programming exploration, we use FPGA on board as accelerators for the Convolutional layers during the training. The training module consists of both offline training as well as online training. During the offline training, the model is trained on the dataset, which is either pre-stored on the SOPC or stored on an external storage connected to the system. After the initial (offline) training is complete with the pre-stored dataset, the camera connected on the SOPC is activated to send in images of the current traffic/section of the road with determined labels (categories) and the training of the model is validated. If the model predicts a wrong category of the streamed image then that image along with it's correct category is stored in a *reFeed Image Stack*, a special stack implementation to hold images with labels, on the system for later (online) training. If during this validation stage of the model, the total prediction accuracy falls below the desired accuracy (*Q* as mentioned in Eq. 1) of the model then the model is re-trained with the images stored in the *reFeed Image Stack*. After completion of every training process the validation phase is re-executed till the prediction accuracy of the model is equal or more than *Q* (Quality of Experience). Methodology of the training module is algorithmically provided in Algo. 1. The main motivation to re-train the CNN model with failed prediction dataset of a known environment is to artificially enhance the accuracy of the model and we call this enrichment in performance as *reFeed Gain factor (r)*. In Section V-B, we have provided the value of *reFeed Gain factor*: *r* noted from the performed experiments and we have also provided a generic mathematical notation of this terminology for better representation as follows:

$$r = |P_i^f - P_i^0|, \text{where } P_i^0 \leq Q \leq P_i^f \tag{2}$$

In the aforementioned equation (Eq. 2) *Q* is the Quality of Experience (see Eq. 1), which is desired for the system to perform well (related to predicting traffic categories), P_i^f is the prediction accuracy of the CNN model after re-trained with *reFeed Image Stack* and P_i^0 is the prediction accuracy in the initial training.

Based on Eq. 2, if we consider *S* as the boost function in prediction accuracy of the CNN model after re-training with reFeed Image Stack feature, which we denote as *reFeed Gain* (*R*), we could represent the *reFeed Gain* as follows:

$$R \leftarrow S(P_i^0) = (P_i^f / P_i^0) \tag{3}$$

Therefore, using Eq. 2 & 3 we could generalize the relationship between P_i^0, R, r, Q [7] as follows:

$$R \times P_i^0 = r + P_i^0, \text{ where } P_i^0 \leq Q \tag{4}$$

Now, in the prediction module our CNN model keeps predicting the traffic category (label) and it either broadcasts the label over the network or it stores the labels along with the video frames on a memory storage, which could be either on-board or external. Later we could use the concept of "*assistive learning*", where a human being manually goes through the stored video frames along with their predicted labels and rectifies any label if there was a wrong prediction. Whenever an image is classified as wrong by the assistive human being then that image goes into the *reFeed Image Stack* of the Prediction module and later the images from this stack is transfered to the *reFeed Image Stack* in the Training module so that the CNN model could be further trained with the images from the *reFeed Image Stack* to enhance *reFeed Gain* (R). We call this method to improving the prediction accuracy of the existing CNN model as "*Continuous Learning*" of the CNN Model for a specific category (as shown in Fig. 2). In this particular work we are only focused on the implementation of *reFeed Image Stack* and *reFeed Gain* in the Training module.

Algorithm 1: Training Module Execution

Input:
1. $\mathcal{I}$: set of n Images from Training & Validation Dataset
2. $\mathcal{T}$: set of m Images from Testing Dataset (for cross-validation)

Output: P: prediction accuracy after training

Initialize: $\mathcal{Q}$ = 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

for *each image* $i \in \mathcal{T}$ **do**
  Prediction = Test (CNN model); ▷ Test 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**
    $S.Push(i)$;
  Calculate mean Prediction Accuracy (P_i^0);

$P = P_i^0$;

Online Training:

{*re-Train with reFeed Image Stack if* $P_i^0 < Q$}

if $P_i^0 < Q$ **then**
  ▷ Need to satisfy condition of Eq. 1
  if *S.Empty() == False* **then**
    {*Traing CNN with reFeed Image Stack*}
    Train (CNN model , $\mathcal{S}$);
    Calculate mean Prediction Accuracy (P_i^f);
  $P = P_i^f$;
  $\mathcal{S}.Count$ = 0; ▷ reset *reFeed Image Stack*
else
  return P;

The proposed methodology (MAT-CNN-SOPC) is bio-inspired due to the fact that human beings constantly keep

[7] P_i^0 denotes initial prediction accuracy, R denotes reFeed Gain, r denotes reFeed Gain Factor and Q corresponds to the Quality of Experience

learning even when they are introduced to a completely new environment so that they could adjust to that environment quickly and adapt to it. By using this same concept we could enhance the learning mechanism of the CNN model for a particular scene-based application.

A. Employed CNN Models

In order to prove the effectiveness of our proposed methodology we chose two popular object-centric CNN architectures, VGG 16 convolutional-layer (VGG16) [28] and ResNet50 [30] CNN. The selected CNN architectures contain 138 million parameters for VGG16 and 26 million parameters for ResNet50.

A typical approach to enable training of very deep networks on small datasets is to use a model pre-trained on a very large dataset, and then use the CNN as as an initialization for the task of interest. This method, referred to as 'transfer learning' [8] [42], [46] injects knowledge from other tasks by deploying weights and parameters from a pre-trained network to the new one. The rationale behind this is that the internal layers of the CNN can act as a generic extractor of image representations which have been pre-trained on one large dataset (source task) and then re-used on other target tasks. Considering the size of the dataset we have used (see Sec. III-B), the only way to apply a deep CNN such as VGG16 and ResNet50, is to reduce the number of trainable parameters. In order to achieve this the first filter stages are held fixed during training (weights are not updated) and overfitting [9] can be avoided. We initialize the feature extraction modules using pre-trained models from a large scale dataset, ImageNet [29], [47]. For the target task (traffic analysis), we design a network that will output scores for the 4 target categories of the dataset used.

V. Experimental Results

A. Experimental Setup

For this research we have taken the 400 highway images (mentioned in Section III-B) and have used that for our training and validation purposes. The dataset is partitioned into two dataset consisting of training and validation sets and during every test randomization algorithm was used on the whole dataset to create the training and validation subsets. We have selected 3 random videos from each category (light, heavy and traffic) of the UCSD dataset and then converted the video stream to image by processing 1 frame out of every 8 frames (˜1.3 *fps*). Since the videos from the UCSD dataset is categorized based on light, heavy and traffic jams, we had to manually categorized into our generic 4 categories: Jam, Heavy, Fluid, Empty and generated 192 images (48 images for each category) for testing purposes. We have performed the following tests, which are separated into groups, as follows:

In Group 1 of tests (*G1*), in test *G1.i* we have broken the 400 training images into two dataset: 360 images for training and 40 images for validation (in 9:4 ratio) of VGG16 pre-trained model. In test *G1.ii* we have broken the dataset into 320 for training and 80 for validation sets, whereas in *G1.iii* it is broken in the ratio of 7:3 and in *G1.iv* it is broken in 3:2. No separate tests were performed to check the accuracy of the categorization after training in Group 1 of tests, but the main motivation was to check the performance of training the VGG16 model on the 400 traffic images.

TABLE I
Tests Performed

Test Groups		
G1: VGG16 performance on Dataset	**G2: VGG16 performance on UCSD Dataset**	***G3: ResNet50 performance on UCSD Dataset***
(i) 90% Training / 10% Validation	(i) 90% Training / 10% Validation	(i) 90% Training / 10% Validation
(ii) 80% Training / 20% Validation	(ii) 75% Training / 25% Validation	(ii) 75% Training / 25% Validation
(iii) 70% Training / 30% Validation	(iii) 50% Training / 50% Validation	(iii) 50% Training / 50% Validation
(iv) 60% Training / 40% Validation	(iv) 75%Training / 25% Validation with reFeed Image Stack Feature	(iv) 75%Training / 25% Validation with reFeed Image Stack Feature

In Group 2 of tests we have taken the pre-trained VGG16 model and have trained the model with training and validation dataset in the ratio as mentioned in Table V-A. But in this group of tests we have checked the categorization accuracy of the model after training is complete with the 192 images of UCSD dataset as mentioned earlier in this section. The UCSD dataset was completely kept hidden during the training process so that we could evaluate the desirability of using VGG16 in scenarios of traffic analysis, which it has not been exposed to in advance (cross validation using unseen UCSD dataset). In Group 3 of tests we ran the similar set of tests as in Group 2 but we replaced the pre-trained CovNet model with ResNet50 and check the categorization accuracy with the UCSD dataset. For each test in every group, we have completely re-trained the CovNet model on our dataset to avoid bias of the model.

To prove our proposed MAT-CNN-SOPC model (Fig. 2) and effective use of *reFeed Image Stack* for further training (transfer learning), we have also performed a series of tests where the model is further trained with images from *reFeed Image Stack*, which is segregated into training and validation set in the ratio of 75:25. Tests G2.iv and G3.iv represents those tests for VGG16 and ResNet50 respectively. To check the testing accuracy after this training method we used a different set of 192 images of the UCSD dataset for the purpose. We trained the CNN models for 10 epochs with a batch size of 10 images. Since we have worked with a small dataset for the problem in hand, we have used several image augmentation

[8] Transfer 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 on top of the semantic image output vector $\mathbf{Y}_{out}$.

[9] Overfitting happens when the CNN model recognizes specific images in your training set instead of general patterns.

techniques such as Reflection [10], Translation [11], etc. to fit the training of the CNN model. We also implemented the training module on ZYNQ FPGA using Vivado HLS (see Section III-A). This is an alternate attempt to accelerate some of the functionalities of CNN.

B. Classification Results

For every single instance of the tests in each group (*G1, G2, G3*) mentioned in the previous subsection (V-A), we have performed the same tests to check consistency and only the maximum result of those tests are reported in this section. In Table V-B we could see the performance of each test, where validation accuracy along with categorization accuracy (testing) are reported.

TABLE II
TESTS PERFORMED

Results of Test Groups		
G1: VGG16 performance on Dataset	**G2: VGG16 performance on UCSD Dataset**	***G3: ResNet50 performance on UCSD Dataset***
(i) Validation Accuracy: 92.50%	(i) Validation Accuracy: 90.00%; **Testing Accuracy: 65.60%**	(i) Validation Accuracy: 92.50%; Testing Accuracy: 40.00%
(ii) Validation Accuracy: 87.50%	(ii) Validation Accuracy: 89.50%; Testing Accuracy: 60.00%	(ii) **Validation Accuracy: 88.00%; Testing Accuracy: 33.33%**
(iii) Validation Accuracy: 89.17%	(iii) Validation Accuracy: 90.00%; Testing Accuracy: 62.30%	(iii) Validation Accuracy: 84.50%; **Testing Accuracy: 61.67%**
(iv) Validation Accuracy: 89.38.50%	(iv) **Validation Accuracy: 94.59%; Testing Accuracy: 87.50%**	(iv) **Validation Accuracy: 95.50%; Testing Accuracy: 81.25%**

As we could see from Table V-B, initially after using the stock traffic image dataset for training the testing prediction accuracy in *G2.i* was 65.60%, which was the highest in that group. But when we have used re-training mechanism (refer to Algo. 1) on the CNN model with *re-Feed Image Stack*, the testing prediction accuracy got boosted to 87.50% for the same group (*G2*) and boosted to 81.25% in *G3* group compared to 33.33% (without re-training). Although, it is a common knowledge that with more images for training accuracy of the CNN model improves but the images used for re-training did not exceed more than 10% of the initial training (offline) dataset in size and given the size of the dataset we are working on, the gain (*reFeed Gain*) in prediction accuracy is solely because of the methodology (training with *reFeed Image Stack*) used rather than the possibility of using more images during training.

Now, using the Eq. 2 and the resulting values from Table V, the calculated *reFeed Gain Factor* (*r*) is 47.92 and the *reFeed Gain* (*R*) (using Eq. 3) is 2.44x for *G3.iv*. Example 1 sheds some light on the phenomenon of enrichment of accuracy as described through *reFeed Gain*.

Observation:

Example 1. In *G3.iv*, the testing accuracy is 81.25% (P_i^f), whereas in *G3.iv* the testing accuracy is 33.33% (P_i^0), thus from Eq. 3:

$$R = (81.25/33.33) = 2.4377 \approx 2.44$$

Therefore, the boost in prediction accuracy for ResNet50 for this example using *reFeed Image Stack* is 2.44x.

The hardware implementation is carried out on Zynq ZC-Z7045. It is observed that near about 95% of DSP (858 out of 900), 55% of BRAM (301 out of 545) and 41% of LUTs (89626 out of 218600) have been utilized.

VI. DISCUSSION

In the work [8], [14], the authors have used the same 400 images dataset and have split it into two: Training and Testing, which means that the authors have used the same dataset for training, validation and testing, which is highly undesirable in this field to evaluate accuracy of the implemented CNN [12]. For example, in [8] they have used the same UCSD dataset to both train and test the VGG model (after splitting the dataset into 75% for training and 25% for testing) and have achieved an accuracy of 96.10%. This way of predicting accuracy of an application based CNN model is highly biased. When we trained our VGG 16 model with separate image dataset and tested the accuracy on the UCSD one, we got an accuracy of just 60.00% (refer G2.ii in Tab. V-B) in comparison. Additionally given the small size of the dataset used, there are two possible challenges, which could be faced. One of those issues being overfitting [13] The other issue is that the model might not be able to train properly and result into less accurate predictions. In [8], the reported accuracy results of the implemented models were on validation instead of reporting the testing accuracy of the same. When the UCSD dataset was used for testing and the curated 400 traffic images for training in our model, we found out that the testing accuracy was very less compared to the validation one, contradicting their results. In order to improve the testing accuracy of CNN models for traffic analysis we came up with MAT-CNN-SOPC Model.

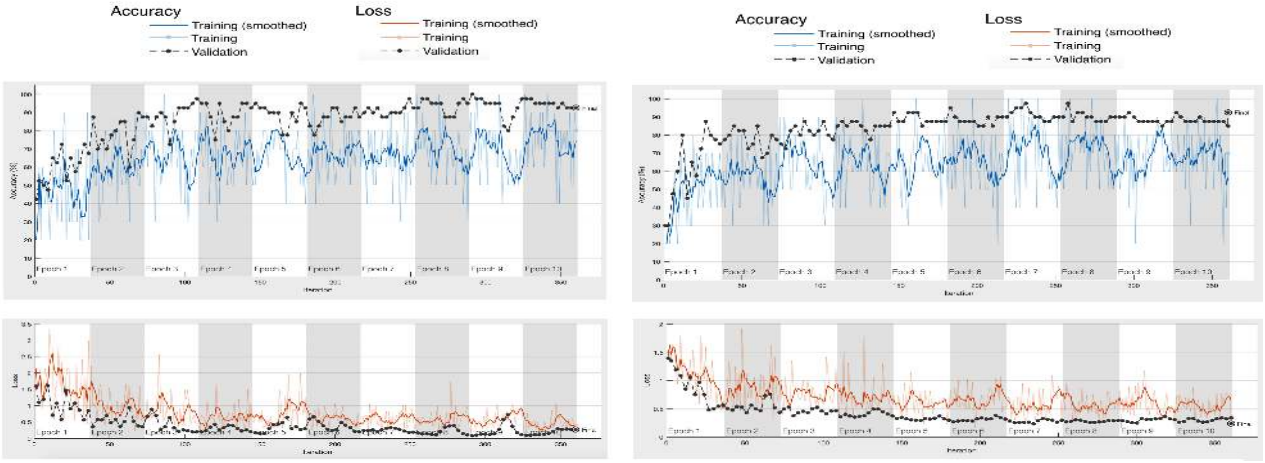
In Table V, we could also see an anomaly in using ResNet50, where with less training images it performed better. One of the possible reasons being overfitting of images when trained with less number of images but from the training graphs (see Fig. V-B) we could understand that is not the case.

[10] Where each image is reflected horizontally.

[11] Where each image is translated by a distance, measured in pixels.

[12] It is undesirable to use the same dataset for training, validation and testing since it introduces high level of bias.

[13] Overfitting happens when the CNN model recognizes specific images in your training set instead of general patterns.



(a) Result: Validation Accuracy & Loss of VGG16 in G1.i Test

(b) Result: Validation Accuracy & Loss of ResNet50 in G3.i Test

Fig. 3. Graph Showing Validation Accuracy & Loss

The other possible reason being mislabeling of the images while testing. For our example we have noticed that sometimes it was difficult even for a human to differentiate between 'Heavy' and 'Fluid' traffic and since the testing images were labeled manually.

From the graphs in Fig. 3 we could also see that the model is somehow underfitting rather than overfitting, but incorporating the MAT-CNN-SOPC Model for the training and prediction has actually made the gap between the training, validation and testing accuracy narrower. Although it could be argued upon that since we have used images from the same camera and on the same road junction to improve the training quality of the CNN model but given the practical application of traffic analysis it is highly likely that the same camera system would operate in the same junction/street region for its lifetime. Thus training the camera system with known environment seems to solve the problem of analyzing and categorizing traffic in a cost effective way. Another noteworthy thing to mention is that for this application and for our tests we have chosen the value of Quality of Experience (QoE) as 70% [14] by default but, this value could be modified based on the desired accuracy for the problem in hand and we could also utilize Eq. 4 to fine tune MAT-CNN-SOPC for the same purpose.

VII. Conclusion

In this paper, we have proposed a novel CNN based categorization model, which could categorize traffic effectively on the programmable system board even with less number of training images in the dataset. To effectively train the CNN to improve prediction accuracy, we have used a combination of transfer learning as well as a novel re-training mechanism on pre-trained CNN models, where the model is re-trained with images from a known environment. We have also introduced Quality of Experience, which researchers in this field could use to choose the right CNN model for their problem and achieve the desired results (in terms of accuracy).

[14] For our traffic categorization issue we found out through testing that chosing QoE value of 70% produced better result in re-training the model for accuracy.

Acknowledgment

This work is supported by the UK Engineering and Physical Sciences Research Council EPSRC [EP/R02572X/1 and EP/P017487/1] and the authors would like to thank the people associated with National Centre for Nuclear Robotics (NCNR) and Extreme Environments for their support and feedback. Somdip would also like to thank everyone from the Embedded and Intelligent Systems Laboratory at the University of Essex for their feedback on this project.

References

[1] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik, "A real-time computer vision system for measuring traffic parameters," in *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*. IEEE, 1997, pp. 495–501.

[2] H.-T. Chen, L.-W. Tsai, H.-Z. Gu, S.-Y. Lee, and B.-S. P. Lin, "Traffic congestion classification for nighttime surveillance videos," in *Multimedia and Expo Workshops (ICMEW), 2012 IEEE International Conference on*. IEEE, 2012, pp. 169–174.

[3] 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.

[4] 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.

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

[6] 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.

[7] 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 Optics and Photonics, 2013, p. 90671A.

[8] Z. Luo, P.-M. Jodoin, S.-Z. Li, and S.-Z. Su, "Traffic analysis without motion features," in *Image Processing (ICIP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 3290–3294.

[9] 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.

[10] 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.

[11] 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.

[12] A. Ess, T. Mueller, H. Grabner, and L. J. Van Gool, "Segmentation-based urban traffic scene understanding." in *BMVC*, vol. 1. Citeseer, 2009, p. 2.

[13] V. Kastrinaki, M. Zervakis, and K. Kalaitzakis, "A survey of video processing techniques for traffic applications," *Image and vision computing*, vol. 21, no. 4, pp. 359–381, 2003.

[14] Z. Luo, P.-M. Jodoin, S.-Z. Su, S.-Z. Li, and H. Larochelle, "Traffic analytics with low frame rate videos," *IEEE Transactions on Circuits and Systems for Video Technology*, 2016.

[15] Y. Wang, Z. Luo, and P.-M. Jodoin, "Interactive deep learning method for segmenting moving objects," *Pattern Recognition Letters*, vol. 96, pp. 66–75, 2017.

[16] R. Jain, R. Kasturi, and B. G. Schunck, *Machine vision*. McGraw-Hill New York, 1995, vol. 5.

[17] S. Dey, "Amalgamation of cyclic bit operation in sd-ei image encryption method: An advanced version of sd-ei method: Sd-ei ver-2," *International Journal of Cyber-Security and Digital Forensics (IJCSDF)*, vol. 1, no. 3, pp. 221–225, 2012.

[18] S. Dey, S. S. Ayyar, S. Subin, and P. A. Asis, "Sd-ies: An advanced image encryption standard application of different cryptographic modules in a new image encryption system," in *Intelligent Systems and Control (ISCO), 2013 7th International Conference on*. IEEE, 2013, pp. 285–289.

[19] S. Dey, "Sd-eqr: A new technique to use qr codestm in cryptography," *arXiv preprint arXiv:1205.4829*, 2012.

[20] S. Dey, J. Nath, and A. Nath, "An advanced combined symmetric key cryptographic method using bit manipulation, bit reversal, modified caesar cipher (sd-ree), djsa method, ttjsa method: Sja-i algorithm," *International Journal of Computer Applications (IJCA 0975-8887, USA)*, vol. 46, no. 20, pp. 46–53, 2012.

[21] S. Dey, S. Agarwal, and A. Nath, "Confidential encrypted data hiding and retrieval using qr authentication system," in *Communication Systems and Network Technologies (CSNT), 2013 International Conference on*. IEEE, 2013, pp. 512–517.

[22] S. Dey and A. Nath, "Modern encryption standard (mes) version-i: An advanced cryptographic method," in *Information and Communication Technologies (WICT), 2012 World Congress on*. IEEE, 2012, pp. 242–247.

[23] S. Dey, K. Mondal, J. Nath, and A. Nath, "Advanced steganography algorithm using randomized intermediate qr host embedded with any encrypted secret message: Asa_qr algorithm," *International Journal of Modern Education and Computer Science*, vol. 4, no. 6, p. 59, 2012.

[24] S. Ehsan, A. F. Clark, K. D. McDonald-Maier *et al.*, "Integral images: efficient algorithms for their computation and storage in resource-constrained embedded vision systems," *Sensors*, vol. 15, no. 7, pp. 16 804–16 830, 2015.

[25] P. Viola and M. J. Jones, "Robust real-time face detection," *International journal of computer vision*, vol. 57, no. 2, pp. 137–154, 2004.

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

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

[28] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[31] G. Kalliatakis, S. Ehsan, and K. D. McDonald-Maier, "A paradigm shift: Detecting human rights violations through web images," *arXiv preprint arXiv:1703.10501*, 2017.

[32] 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?" *arXiv preprint arXiv:1703.04103*, 2017.

[33] G. Zhang, B. E. Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks:: The state of the art," *International journal of forecasting*, vol. 14, no. 1, pp. 35–62, 1998.

[34] M. Grecu and W. F. Krajewski, "Detection of anomalous propagation echoes in weather radar data using neural networks," *IEEE transactions on geoscience and remote sensing*, vol. 37, no. 1, pp. 287–296, 1999.

[35] "Industry's most cost-effective programmable architecture," https://www.altera.com/products/fpga/cyclone-series/cyclone/features/cyc-architecture.html, accessed: 2018-06-26.

[36] "Bdti pocket guide to embedded processors," https://www.bdti.com/Resources/Pocket-Guide, accessed: 2018-06-26.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

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

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

[43] "Vivado high-level synthesis," https://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html, accessed: 2018-07-01.

[44] S. Dey and G. Kalliatakis, "Mat-cnn-sopc: Traffic analysis using cnns on fpga," https://github.com/somdipdey/MAT-CNN-SOPC, accessed: 2018-07-01.

[45] 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.

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

[47] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.