SoildNet: Soiling Degradation Detection in Autonomous Driving

Arindam Das
Detection Vision Systems
Valeo India
arindam.das@valeo.com

Abstract

In the field of autonomous driving, camera sensors are extremely prone to soiling because they are located outside of the car and interact with environmental sources of soiling such as rain drops, snow, dust, sand, mud and so on. This can lead to either partial or complete vision degradation. Hence detecting such decay in vision is very important for safety and overall to preserve the functionality of the "autonomous" components in autonomous driving. The contribution of this work involves: 1) Designing a Deep Convolutional Neural Network (DCNN) based baseline network, 2) Exploiting several network remodelling techniques such as employing static and dynamic group convolution, channel reordering to compress the baseline architecture and make it suitable for low power embedded systems with $\sim$1 TOPS, 3) Comparing various result metrics of all interim networks dedicated for soiling degradation detection at tile level of size 64×64 on input resolution 1280×768. The compressed network, is called SoildNet (**S**and, sn**O**w, ra**I**n/d**I**rt, oi**L**, **D**ust/mu**D**) that uses only 9.72% trainable parameters of the base network and reduces the model size by more than 7 times with no loss in accuracy.

1 Introduction

Vision based algorithms are particularly depending on the image data that are passed from the camera sensors with almost 360°surrounding view as shown in figure 1. The quality of the input vision needs a certain level of validation before being fed to other downstream algorithms since the performance of the allied processes degrade severely if there is substantial decay in the vision. Hence it is extremely critical to detect about the degradation in the input vision and report to the system while aiming for Level 4 autonomous driving. This will ensure the safety of the passengers and others to avoid any unprecedented event.

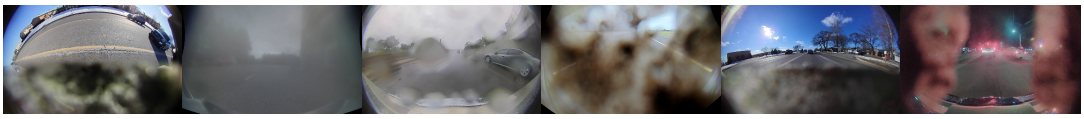
(a) (b) (c) (d)

Figure 1: Different field of vision of surround-view cameras: (a) Front, (b) Rear, (c) Left and (d) Right

There are very few works available in the literature on the reported problem statement and the approaches can be classified in two categories, $1)$ Image restoration and $2)$ Soiling detection. In the first category, there has been attempts to recover the input image by removing rain drops [1].

Machine Learning for Autonomous Driving Workshop at the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Another successful effort has been to dehaze [2] the high resolution ultrasound images. For both the approaches, the network was trained with a pair of *defective-clean* images. However, the deployability of the techniques in real-time on a low power automotive SoC is questionable. In the second category, in [3] GAN (Generative Adverserial Network) was used to augment the soiling samples. The same approach was followed in [4] as well.



(a) (b) (c) (d) (e) (f)

Figure 2: Different types of *soiling*: (a) grass, (b) fog, (c) rain drops, (d) dirt, (e) splashes of mud, (f) splashes of mud in night

2 Contribution

- **Critical but less explored problem**: Soiled vision can impede other vision based applications and cause several safety issues if it goes unnoticed. Hence this important problem requires a fast and *low resource environment* friendly solution. However, the availability of very few works in the literature demonstrates that this task is quite unexplored. To the best of author's knowledge, this is the first study on tile based soiling degradation detection where the network recommendations have been investigated meticulously from the embedded platform perspective.
- **Network optimization and analysis**: In this study, it has been demonstrated how the main take aways of few of the existing network remodeling methods [5] [6] can help to optimize a base network incredibly by 7 folds in model size and use only 9.72% trainable parameters of the base network with no sign of loss in accuracy. Eventually the best optimized network outperforms all the other incrementally optimized variants of the base network for soiling detection task. However, it is to be noted that in any of the earlier works, group convolution was never applied at all convolution layers due to the issue with insufficient feature blending. The present study overcomes this bottleneck and discusses in section 5.
- **Tile level soiling detection**: Most of the times, soiling effect appears discontinuously over the image. Hence it makes more sense to detect soiling more locally than at the image level. Detecting soiling more locally does not completely disqualify the input image to be used for other algorithms such as VSLAM (Visual Simultaneous Localization and Mapping) [7], motion stereo [8], environment perception [9], 3D object detection [10], semantic segmentation [11]. Rather all the downstream algorithms can still run on the tiles that are predicted as soiling free. It should be noted that the definition of tile in this paper is not limited to the defined dimension, rather it can be extended to as far as the whole image and reduced as much as pixel level.

3 Soiling Degradation Detection

With reference to the detail explanation in earlier sections, while it is clear that there is no way to protect the camera sensors from being effected by the various sources of degradation. It is important for obvious reasons to recover the visual field when degradation is detected on the substantial portion of the input image. It is the cleaning system that is invoked automatically to remove the soiled objects by spraying warm water. This system includes a separate tank that reserves water for this purpose and needs refuelling just like gas. System level details on the cleaning system is shown in [4]. Various types of the soiling and its classes that are considered in this experiment are discussed below.

3.1 Types of Soiling

The decline in vision can be either due to adverse weather conditions and this covers soiling types for example *snow*, *rain drops*, *fog* etc. or the other types that emerge regardless bad weather such as *mud*, *grass*, *oil*, *dust*, *sand* etc. Figure 2 shows few examples of different soiling types that are considered in this experiment.

3.2 Classes of Soiling

Different types of soiling discussed in the previous section are divided into three categories: *clean*, *opaque* and *transparent*. This categorization is done based on the visibility within the region of interest that is per tile.

- **Clean**: When a tile has completely freeview then it is categorized as clean.
- **Opaque**: A tile is marked as opaque when the vision is totally blocked. The complete decay in vision can happen because of any type of soiling that is discussed in the previous section.
- **Transparent**: Sometimes, due to the uneven distribution of the soiling objects on the camera lens, majority of the tiles in the input image do not loose complete visibility. Rather one can see through few of the tiles that are partly affected. Tiles with such qualities are considered as *transparent* in this study.

It is possible that presence of multiple classes are observed within one tile, however the tiles are annotated based on the presence of the dominant class per tile. In this study, an input image of resolution 1280×768 is annotated per tile of size 64×64 and each tile represents a soiling class, hence it is possible to see an input image that contains all soiling categories across tiles.

4 Dataset

Due to unavailability of any public dataset, several driving scenes or videos have been used to extract the frames, however not all successive frames were extracted. This is because highly correlated samples do not contribute much during training.

For the reported problem, it is more critical to predict a *clean* tile correctly. This is because, a high number of false positives will lead to cleaning a camera that is already clean more frequently. As an effect the water tank will need refuelling repeatedly. Hence it makes sense to make the model moderately biased towards the *clean* class. In this experiment, total $144\,053$ sample images are used out of which $70\,000$ samples are pure clean images, which means that all tiles are soiling free. Higher number of clean samples will help to learn better discriminative features of clean class, hence the model tends to be biased towards clean. The distribution of sample across cameras is as follows, FV: $36\,259$; RV: $36\,160$, MVR: $35\,435$; MVL: $36\,199$. A tile of size 64×64 on input resolution 1280×768 makes 20 tiles along width and 12 tiles along height, thus a single sample contains total 20×12 tiles and the tile based class distributions are - Clean: $25\,459\,238$; Opaque: $6\,341\,435$; Transparent: $2\,772\,047$. Change in camera view impacts the appearance of the object significantly, and thus it is necessary to check how the classes are well spread at tile level across different camera views. Table 1 shows that the classes are adequately distributed in all four camera views.

Class	FV	RV	MVR	MVL
Clean	36141	35926	35435	35755
Opaque	17877	17813	16740	17764
Transparent	17866	17716	17648	17753

Table 1: Camera view-wise presence of all three classes

The dataset is subdivided into training, validation and test sets with partition ratios of 60%, 20% and 20% respectively. In Figure 2, a few samples are shown from the dataset. It is to be noted that all the samples are fisheye and in YUV420 [12] planar image format as produced by the ISP (Image Signal Processor) cameras. For soiling detection task, fisheye images were not corrected because a separate preprocessing module would be required and that would increase the overall inference time to get the end-to-end results.

5 Proposed Method

To accomplish soiling degradation detection task on low power automotive SoC, first a base network (Net-1) is designed to take input in YUV420 planar format where the dimension of Y, U and V

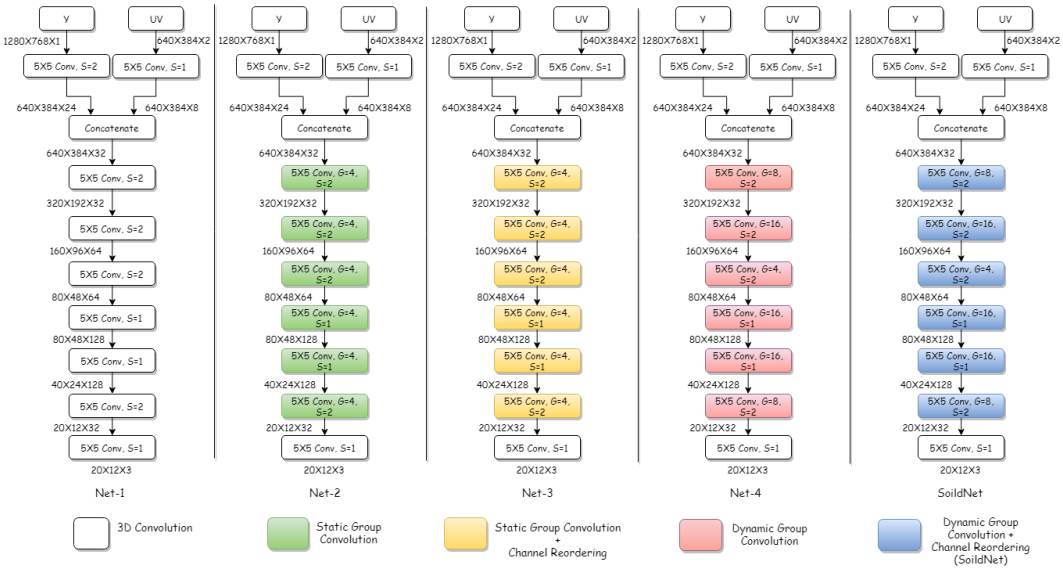


Figure 3: Proposed networks with different schemes for soiling detection. $Conv.$: Convolution, G : group size, S : stride size

channel are 1280×768, 640×384 and 640×384 respectively. As there is a mismatch of dimension between Y and UV, so the network is designed to take two inputs, one is for Y and the other one is UV together. Later through convolution operation, both set of feature maps (Y and UV) are brought down to similar dimension and concatenated to make the network single stream. This approach can be checked in details in [13]. Net-1 is further refactored to 4 other networks (Net-2, Net-3, Net-4 and SoildNet) as shown in figure 3 to obtain the best variant of Net-1 that will be lightweight and more efficient. The network refactoring is done using group convolution and channel reordering that are discussed in the next section.

5.1 Group Convolution

The idea to perform convolution operation group wise was first introduced in AlexNet [14]. However the main intention was to distribute the number of operations in two GPUs. Later in ResNeXt [5], this proposal was used to boost the accuracy with reduced network complexity on an object recognition task. The earlier work in [5] considered static number of groups in the network. The current work extends this concept by adding group convolution in all convolution layers (Net-2, Net-3, Net-4, SoildNet), also we experiment with dynamic group size (Net-4, SoildNet) to reduce the network complexity by more than two times in trainable parameters (Net-3 vs. Net-4). The network schemes do not contain residual connections because group convolution was found to be not very effective for the networks with low depth as presented in [15]. Also, a similar study on residual connection for lightweight networks [16] is the reason not to use them in any of the proposals. While adding group convolution at all layers of the network brings another challenge of insufficient feature blending. This is overcome through channel reordering that is discussed in the following section.

5.2 Channel Reordering

The concept of channel reordering is highly inspired from ShuffleNet [6]. While performing group convolution, the feature information are limited within the group. To make the features blend across groups, the feature maps are shuffled in an ordered way that makes sure in the next layer each group contains at least a candidate feature map from each group of the previous layer.

In this experiment, two constraints have been found in ShuffleNet and they are solved in this study. First, it is now well known that group convolution is effective to bring down the network complexity but at the some time this method can not be applied in all convolution layers. This is because the

feature information will not be spread across all feature maps and this step is necessary to learn better descriptors. The main reason of the insufficient feature blending is that all the feature maps will never undergo convolution operation together as they are separated by groups. After performing one or two layers of consecutive group convolution, generally a convolution layer with kernel size 1×1 is added to blend the features across channel. As an effect of this, convolution operation on all feature maps again shoots the number of trainable parameters significantly high. In order to execute group convolution at all convolution layers throughout the network, channel reordering is added that helps to ensure feature blending across groups. Certainly following this way, feature blending will not be as effective as normal convolution on all feature maps but definitely the blending will be mostly same as the network is trained for higher number of epochs. And the late convergence of the network with group convolution at all layers and channel reordering impact only on the training time. Another reason that the channel reordering was not applied at all layers in ShuffleNet due to its usage of residual connection.

ShuffleNet uses channel shuffling while maintaining the constant number of groups in the network. This significantly limits further reduction of the GMACS (discussed in the next section), number of parameters and model size considering group convolution is not performed at all layers. In this experiment, we designed two networks (Net-4 and SoildNet) such that Net-4 contains different number of groups at each layer and SoildNet contains same number of groups as Net-4 but it includes channel reordering method. Here, the group sizes are determined based on the following idea: One convolution layer with higher number of groups heavily reduces the number of trainable parameters but it limits feature blending due to the feature maps are separated by more number groups then next convolution layer should use less number of groups to blend the features well. So a good balance is maintained between reducing the number of trainable parameters and feature blending. In SoildNet, apart from group convolution with dynamic group size, channel reordering ensures that the features are blended even when the group is size less by shuffling the feature maps across groups. The effectiveness of this approach can be seen in table 3.

6 Analysis of SoildNet

CNN mostly follows two major operations while performing a convolution task - multiplication and addition. Total number of operations involved in a network is represented by GMACS (Giga Multiply Accumulate Operations per Second) unit. Table 2 furnishes the details about the number of trainable parameters used in all five network schemes (Net-1, Net-2, Net-3, Net-4, SoildNet) along with their GMACS and model size. As an effect of more than 90% reduction of network parameters due to group convolution from baseline network in two variants of SoildNet (with and without channel reordering), the model size is reduced by more than 7 times. This is quite a significant and encouraging number while deploying a model on a low power SoC. Also it is clearly seen that the GMACS of SoildNet (and Net-4) is quite less than the baseline or other network schemes. Thus this factor helps SoildNet to be faster during inference.

Network	Operations (GMACS)	Parameters	Model size (KB)
ResNet-10	24.19	4,937,881	68,261
Net-1	4.203	900,849	3,569
Net-2	1.236	228,401	965
Net-3	1.236	228,401	965
Net-4	0.6672	87,601	478
SoildNet	0.6672	87,601	478

Table 2: Analysis of computation complexity of all network proposals including ResNet-10 [17]

It is to be noted that channel reordering technique does not have any influence on the size of network parameters because it only changes the position of the feature maps thus the number of parameters in the network remain same. For the very same reason, there is no impact on model size as well for the network with and without channel reordering.

As the present study is focused on designing efficient lightweight networks for soiling degradation detection task, it seemed interesting to do an analysis from the perspective of computation complexity of a standard network such as ResNet-10 [17] (the lightest version of the ResNet family) for an embedded platform with $\sim$1 TOPS (Tera Operations per Second). Table 2 shows the GMACS, number of trainable paramaters and model size of ResNet-10 with respect to the input resolution used in this work. However, due to the reasons stated below ResNet-10 is not considered further in this work.

- **Number of operations**: The reported GMACS of ResNet-10 is way too much high to be considered for an embedded platform.
- **Model size**: Automotive SoCs generally provide only few megabytes of memory where all the autonomous algorithms of the ADAS system need to be accommodated. Hence, with such budget constraint, acceptance of a model of size more than 5MB is questionable, especially when we target higher FPS (Frames Per Second).
- **Residual connections**: The memory budget heavily increases with the networks containing residual connections since the feature maps need to be saved in the memory to perform addition later at the end of the residual connection. During feature maps retrieval, DMA (Direct Memory Access) transfers the data from the storage to the very limited cache memory for feature map summation. If the cache memory fails to hold the entire data then DMA keeps on copying and processing the data partially. Eventually with more number of feature maps this rolling buffer method is performed quite a number of times and it leads towards higher inference time of the network on the embedded platform.

7 Embedded Platform Constraints

The network models explained in section 5 follow floating point operations because these architectures are trained in Keras [18] framework on GPU. However, most of the embedded SoCs follow 16-bit fixed point operations, hence the data are quantized when the model trained on a GPU is deployed on a target device. In this study, the throughput of the SoC is $\sim$1 TOPS and capable to support 400 GMACS. All the network proposals are well aligned with the constraints of the CNN IP (Image Processor) to make sure its full utilization of the resources. For example, pooling layer is not used in the network to reduce an extra clock-cycle, rather stride is applied to reduce the problem space. Also all convolution kernels are 5×5 to ensure 100% core utilization of the CNN IP.

As per GPU implementation, performing group convolution involves first slicing input feature maps into a number of groups, then execute convolution operation on each group and finally concatenate the output feature maps of all groups. However this extra overhead does not exist in the embedded environment. This is because on CNN IP memory address of each feature map is passed while doing convolution operation, so to follow group wise convolution, simply memory address of the feature maps need to be sent in group wise fashion. It is also to be highlighted that the channel reordering needs extra effort on GPU that includes again feature map slicing in a way so that resultant feature maps are in desired order. However, this effort is completely neutralized on the hardware since the feature maps are handled only through memory locations. Apparently when the succeeding convolution layer would be expecting reordered feature maps then the feature maps from the desired memory locations would be sent to ensure channels are reordered.

8 Experimental Results

This section provides details about the performance of all network propositions on the test dataset. The discussion includes training strategy that was followed for all 5 networks and reporting classwise standard metrics for overall network evaluation.

8.1 Training Strategy

All the network schemes are implemented using Keras [18] framework. Batch normalization layer is added between each convolution layer and ReLU as activation. Training was done batch wise with batch of size 16 for 50 epoch, initial learning rate was set to 0.001 along with an optimizer Adam [19]. Categorical cross entropy and categorical accuracy were used as loss and metrics respectively

for all networks. As the networks are less in depth and no pre-training was done, to make the network weights more robust, the concept of layer-wise training in a supervised fashion could be adapted as presented in [20].

8.2 Evaluation

To execute a fair comparison about the efficacy of all network schemes, few standard metrics are considered such as TPR (True Positive Rate), TNR (True Negative Rate), FPR (False Positive Rate), FNR (False Negative Rate) and FDR (False Discovery Rate) respectively. In order to get better insight about the performance, these metrics are computed for each class on the test dataset. The rule to interpret these metrics is to aim for higher values of TPR, TNR and lower values of FPR, FNR, FDR respectively.

-	True Positive Rate (TPR)			True Negative Rate (TNR)			False Positive Rate (FPR)			False Negative Rate (FNR)			False Discovery Rate (FDR)		
Network	Clean	Opaque	Transparent	Clean	Opaque	Transparent	Clean	Opaque	Transparent	Clean	Opaque	Transparent	Clean	Opaque	Transparent
Net-1	0.9607	0.9157	0.4939	0.8864	0.9602	0.9753	0.1135	0.0397	0.024	0.0392	0.0842	0.506	0.0402	0.1639	0.3632
Net-2	**0.9902**	0.8923	0.3706	0.8048	**0.9835**	0.9861	0.1951	**0.0164**	0.0138	0.0097	0.1076	0.6293	0.0652	**0.0766**	0.3001
Net-3	0.9724	0.921	0.5413	0.9024	0.9708	0.9759	0.0975	0.0291	0.024	0.0275	0.0789	0.4586	0.0343	0.1249	0.3371
Net-4	0.9916	0.9302	0.2859	0.8136	0.9739	**0.9934**	0.1863	0.026	**0.0065**	**0.0083**	0.0697	0.714	0.0624	0.1123	**0.2087**
SoildNet	0.9556	**0.9303**	**0.5973**	**0.938**	0.9642	0.9649	**0.0619**	0.0357	0.035	0.0443	**0.0696**	**0.4026**	**0.0224**	0.1479	0.4019

Table 3: Comparison of classwise accuracy between the base model (Net-1) and other network propositions (Net-2, Net-3, Net-4, SoildNet) for tile level soiling degradation detection

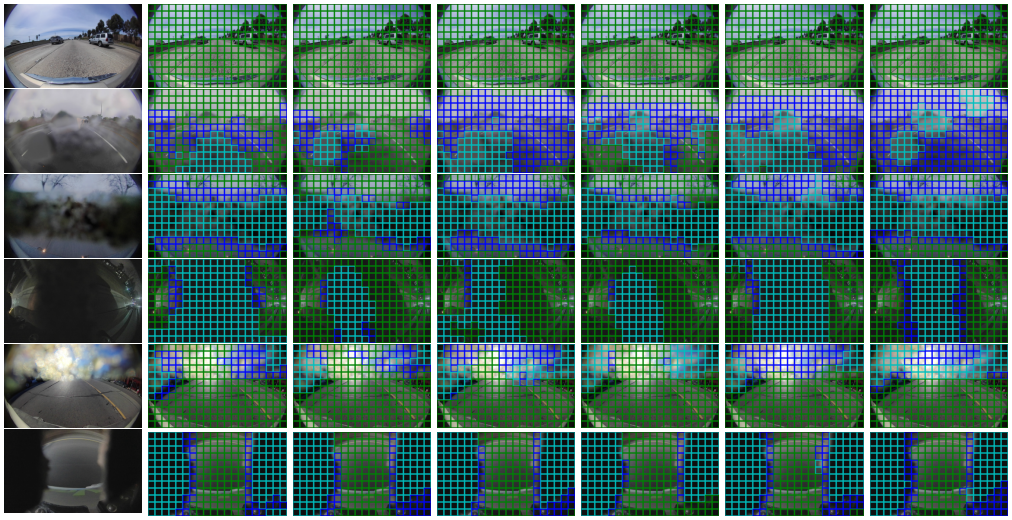


Figure 4: Examples of 64×64 tile based soiling degradation detection output by the proposed network recommendations compared to GT (Ground Truth). From left to right: Input image, output from Net-1, Net-2, Net-3, Net-4, SoildNet, GT.
Color codes: Green - *Clean*, Cyan - *Opaque*, Blue - *Transparent*. Best viewed in color.

Table 3 summarizes the performance of all networks and the effectiveness of SoildNet is noticeable among other network propositions. The recipe of dynamic group convolution with channel reordering makes the network robust to learn better discriminative features for all classes equally and the proof is about 10% gain in TPR for class *transparent* from the base network (Net-1) without degrading the performance of other classes. The results between Net-2 vs. Net-3 furnish the efficacy of channel reordering with static number of groups through out the network. The effectiveness of channel reordering with group convolution of dynamic number of groups can be seen in Net-4 vs. SoildNet performance. Even though Net-2 shows promising performance for class *clean* but it fails to provide a reasonable accuracy for class *transparent*. However, it is true that *transparent* class has comparatively low TPR across all networks and the possible justification is that it is often confused

with *clean* class. Table 4 further summarizes the results by computing the average of class wise accuracy for each metric. The main take away of this result is that out of 5 standard metrics used in this experiment SoildNet outperforms other networks on 4 metrics and thus it becomes the best proposition among all. Apart from metrics, the output of all network schemes on soiling degradation detection is demonstrated in figure 4 where the soiling outputs are at tile level of size 64×64. In grid representation, different color codes such as green, cyan and blue are used to indicate *clear*, *opaque* and *transparent* classes.

-	Average				
Network	TPR	TNR	FPR	FNR	FDR
Net-1	0.7901	0.9406	0.059	0.2098	0.1891
Net-2	0.751	0.9248	0.0751	0.2488	0.1473
Net-3	0.8115	0.9497	0.0502	0.1883	0.1654
Net-4	0.7359	0.9269	0.0729	0.264	**0.1278**
SoildNet	**0.8277**	**0.9557**	**0.0442**	**0.1721**	0.1907

Table 4: Comparison of average classwise accuracy between the base model (Net-1) and other network schemes (Net-2, Net-3, Net-4, SoildNet)

9 Conclusion

In this work, soiling degradation detection task, an extremely critical but relatively less explored problem has been presented in the field of autonomous driving. The solution proposed in this paper came through several interim network propositions, in particular adaptability of group convolution with static or dynamic number of groups and channel reordering in low resource environment. In this study, extensive experiment outcomes on a considerably large soiling dataset can be summarized as follows: 1) group convolution at all convolution layers reduces the network complexity immensely, 2) channel reordering can be effective to blend the features across channels and 3) channel reordering is more effective with group convolution with dynamic number of groups. The network schemes presented in this paper are domain agnostic and can be easily adapted in the encoder architecture for any vision tasks to be deployed on low resource platform.

References

[1] A. Pfeuffer and K. Dietmayer, "Robust semantic segmentation in adverse weather conditions by means of sensor data fusion," *arXiv preprint arXiv:1905.10117*, 2019.

[2] S. Ki, H. Sim, J.-S. Choi, S. Kim, and M. Kim, "Fully end-to-end learning based conditional boundary equilibrium gan with receptive field sizes enlarged for single ultra-high resolution image dehazing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 817–824, 2018.

[3] M. Uřičář, P. Křížek, D. Hurych, I. Sobh, S. Yogamani, and P. Denny, "Yes, we gan: Applying adversarial techniques for autonomous driving," *arXiv preprint arXiv:1902.03442*, 2019.

[4] M. Uřičář, P. Křížek, G. Sistu, and S. Yogamani, "Soilingnet: Soiling detection on automotive surround-view cameras," *arXiv preprint arXiv:1905.01492*, 2019.

[5] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

[6] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6848–6856, 2018.

[7] N. Karlsson, E. Di Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian, and M. E. Munich, "The vslam algorithm for robust localization and mapping," in *ICRA*, pp. 24–29, 2005.

[8] C. Unger, E. Wahl, and S. Ilic, "Parking assistance using dense motion-stereo," *Machine Vision and Applications*, vol. 25, no. 3, pp. 561–581, 2014.

[9] U. Franke, C. Rabe, H. Badino, and S. Gehrig, "6d-vision: Fusion of stereo and motion for robust environment perception," in *Joint Pattern Recognition Symposium*, pp. 216–223, Springer, 2005.

[10] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun, "Monocular 3d object detection for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2147–2156, 2016.

[11] A. Das, S. Kandan, S. Yogamani, and P. Křížek, "Design of real-time semantic segmentation decoder for automated driving," *arXiv preprint arXiv:1901.06580*, 2019.

[12] L. Yuan, G. Shen, F. Wu, S. Li, and W. Gao, "Color space compatible coding framework for yuv422 video coding," in *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, pp. iii–185, IEEE, 2004.

[13] T. Boulay, S. El-Hachimi, M. K. Surisetti, P. Maddu, and S. Kandan, "Yuvmultinet: Real-time yuv multi-task cnn for autonomous driving," *arXiv preprint arXiv:1904.05673*, 2019.

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

[15] A. Das, T. Boulay, S. Yogamani, and S. Ou, "Evaluation of group convolution in lightweight deep networks for object classification," in *Video Analytics. Face and Facial Expression Recognition*, pp. 48–60, Springer, 2018.

[16] A. Das and S. Yogamani, "Evaluation of residual learning in lightweight deep networks for object classification," in *Proceedings of the 20th Irish Machine Vision and Image Processing Conference*, pp. 205–208, 2018.

[17] 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*, pp. 770–778, 2016.

[18] F. Chollet *et al.*, "Keras." `https://keras.io`, 2015.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[20] S. Roy, A. Das, and U. Bhattacharya, "Generalized stacking of layerwise-trained deep convolutional neural networks for document image classification," in *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 1273–1278, IEEE, 2016.