# A SIMPLE HYBRID FILTER PRUNING FOR EFFICIENT EDGE INFERENCE

*S.H.Shabbeer Basha, Sheethal N Gowda, Jayachandra Dakala*

DryvAmigo, Bangalore, India.

## ABSTRACT

Convolutional Neural Networks have been extensively used for solving many vision problems. However, due to high memory and computational requirements, deployment of these models on edge devices is limited. Many embedded friendly models such as MobileNet, ShuffleNet, SqueezeNet, and many more are proposed to serve this purpose. But these models are still not compact enough to deploy on edge devices. The popular metric-based pruning methods (which are aimed at pruning insignificant and redundant filters) could achieve limited compression for embedded friendly models such as MobileNet. In this paper, we propose a novel hybrid filter pruning method that prunes both redundant and insignificant filters at the same time. Additionally, we have designed custom regularizers that enable us to prune additional filters from convolutional layers. Pruning experiments are conducted on MobileNetv1 based Single-Shot Object Detector (SSD) for face detection problem. Through our experiments, we could prune 40.11% of parameters and reduce 67.03% of FLOPs from MobileNetv1 with a little drop in model performance (1.67 mAP on MS COCO). On an ARM-based edge device, the inference time is reduced from 198ms to 84ms.

***Index Terms***— Neural Network Compression, Magnitude based filter pruning, hybrid filter pruning, Floating Point Operations (FLOPs).

## 1. INTRODUCTION

Deep Learning models have demonstrated substantial success to solve various computer vision problems such as object recognition [1], object detection [2], and many more. Nevertheless, these models demand high computational and memory requirements that block most of the popular CNNs [1] to be deployed on low compute and storage devices like mobiles, drones, etc. There has been significant progress achieved in developing hardware-specific deep CNN acceleration frameworks like TensorRT. Still, there is a great demand to reduce the FLOPs and trainable parameters of a Convolutional Neural Network (CNN). Popular CNN compression methods include connection pruning [3, 4], weight quantization [5], filter pruning [6, 7] and low-rank filter decomposition [8].

Among these network compression methods, network pruning has shown extensive prospects in numerous applications. There are many attempts to i) either prune filter elements to acquire sparse weight matrices (weight pruning) [3] ii) prune the filters entirely from the network (filter pruning) [6, 9]. Weight quantization [5] is another class of methods used for CNN compression in which network parameter values (floating point numbers) are represented with lower precision. Knowledge distillation [10] methods aim to compress a larger network (teacher model) by transferring the knowledge to a smaller network (student model).

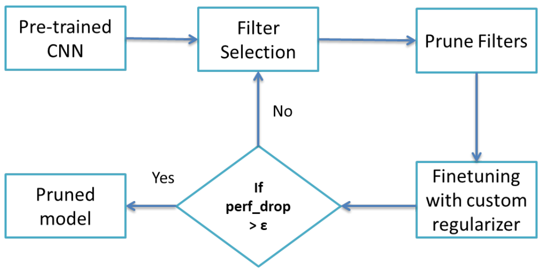

**Fig. 1**. An overview of a typical filter pruning method.

Weight pruning and quantization methods require specialized software [11] or hardware [12] to achieve acceleration. However, filter pruning methods do not require any specialized software or hardware support. Due to this, filter pruning methods achieved model acceleration independent of the embedded platforms like OpenVino, ARMNN, SNPE, etc. Mostly, filter pruning methods are aimed at pruning either insignificant filters or redundant filters. To prune insignificant filters, metrics such as $\ell_1$-norm [9], $\ell_2$-norm [13] are widely used. On the other hand, Pearson's correlation coefficient [6], cosine similarity [14], and many more metrics are used to prune redundant filters. However, these pruning methods are majorly employed on large CNN models such as VGG-16, ResNet-50. It is recommended to use embedded-friendly models such as MobileNetv1 for real-time Driver Monitoring System (DMS) applications due to the reason that these models require low computational and storage requirements. Most of the existing model compression methods for MobileNetv1 [15] are based on knowledge distillation [10]. These methods assume that MobileNetv1 is a lightweight CNN which is much efficient compared to other models like VGG-16, ResNet-50, etc. Although the DMS applications utilize em-

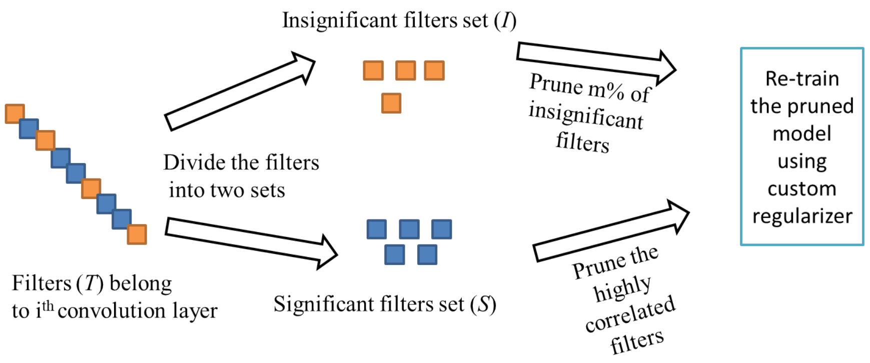
Insignificant filters set (I)
Prune m% of insignificant filters
Re-train the pruned model using custom regularizer
Divide the filters into two sets
Filters (T) belong to ith convolution layer
Significant filters set (S)
Prune the highly correlated filters

**Fig. 2**. The proposed hybrid filter pruning method prunes both insignificant and redundant filters in each pruning iteration. The pruned model is finetuned using the custom loss (given in Eq. 4)

bedded friendly models, still, these models consume a considerable amount of computational and storage resources.

In this paper, we develop filter pruning methods to reduce the storage and computational requirement of MobileNetv1 which is used for face detection problem. We conduct pruning experiments using popular magnitude-based filter pruning methods like $\ell_1$-norm, $\ell_2$-norm based [9, 13] and redundant [6] (Pearson's Correlation coefficient) filter pruning methods. From our experiments, we observe that higher compression can not be achieved on MobileNetv1 using either magnitude-based or redundant filter pruning methods alone. So to achieve a higher compression rate in terms of FLOPs reduction (%), we propose a hybrid filter pruning approach which prunes both insignificant and redundant filters in the same pruning iteration. Additionally, two regularizers are introduced that enable us to prune additional convolutional filters without harming the performance of the model. The overview of a typical pruning method is depicted in Fig. 1.

Next, we discuss the proposed pruning method and the custom regularizers implemented in this paper. In Section 3, we discuss the experimental results and pruning rates achieved for MobileNetv1. Finally, we provide the concluding remarks of the paper.

## 2. METHODOLOGY

Here, we first brief the magnitude based and correlation based filter pruning methods. Later, we discuss the proposed hybrid filter pruning method.

### 2.1. Preliminaries

In general, filter pruning methods use some metrics to measure the significance of a filter. Li *et al.* [9] proposed a filter pruning method based on the $\ell_1$-norm of the convolutional filters. They consider that the filters having the least $\ell_1$-norm are insignificant for the given task which can be pruned without hampering the performance of the model. Similarly, He *et al.* [13] utilized $\ell_2$-norm of the filters to rank them such that the filters with low rank (less $\ell_2$-norm) are considered as insignificant and can be pruned from the network. Recently, Correlation Filter Pruning (CFP) is proposed by Singh *et al.* [6] to prune redundant filters. The CFP method utilizes Pearson's correlation between the filters to measure their similarity.

Let us consider a convolutional layer that has $n$ filters, the correlation coefficient for all $nc_2$ filter pairs is computed. After computing the correlation coefficient for all the filter pairs, then the filter pairs having the correlation above a threshold $\varepsilon$ (which is set to 0.75 in our experiments) are chosen for pruning. Now, instead of pruning one filter from each redundant filter pair, in [6], authors have tried to improve the correlation between the filters using an optimization step.

In this paper, we consider MobileNetv1 [15] based Single Shot Object Detection (SSD) [2] model for object detection task to verify the robustness of the developed filter pruning methods. The original loss function for this task is given in Eq. 1 which involves classification loss, localization loss, and $\ell_2$-regularization terms.

$$L = \frac{1}{N}\left(L_{conf}(x,c) + \lambda_1 L_{loc}(x,l,g) + \lambda_2 \sum_{i=1}^{n} w_i^2\right) \quad (1)$$

here N indicates the number of matched default bounding boxes. $L_{loc}$ represents the localization loss between predicted box ($l$) and ground truth box ($g$). $L_{conf}$ is the softmax loss for multi-class classification. For complete details on these losses, please refer to [2].

### 2.2. The Proposed Filter Pruning Method

Many filter pruning methods either prune insignificant filters [9, 13] or redundant filters [6]. Our contributions are three-fold i) we propose a hybrid filter pruning method ii) introduced two custom regularizers iii) employed these regularizers to $\ell_1$/$\ell_2$-norm based and hybrid filter pruning methods.

**Table 1**. Pruning results obtained for MobileNetv1 based SSD on MS COCO dataset using $\ell_1$-norm, $\ell_2$-norm, CFP, and hybrid filter pruning methods.

| S.No. | Method | mAP | Remaining Parameters (Drop %) | Remaining FLOPs (Drop %) |
|---|---|---|---|---|
| 1 | base model | 25.63 | 6.83M (0) | 1.23B (0) |
| 2 | Knowledge Distillation [10] | 27.70 | 6.83M (0) | 1.23B (0) |
| 3 | LFFSD [16] | 20.40 | 6.50M (4.4) | - |
| 4 | $\ell_1$-norm - I | 25.22 | 6.45M (05.51) | 1.11B (09.77) |
|  | $\ell_1$-norm - V | 24.52 | 5.26M (22.91) | 0.74B (39.91) |
|  | $\ell_1$-norm - VI | 21.86 | 5.03M (26.35) | 0.67B (45.52) |
| 5 | $\ell_1$-norm + $C_1$ - I | 24.66 | 6.45M (05.51) | 1.11B (09.77) |
|  | $\ell_1$-norm + $C_1$ - V | 24.21 | 5.26M (22.91) | 0.74B (39.91) |
|  | $\ell_1$-norm + $C_1$ - VII | 22.53 | 4.82M (29.42) | 0.61B (50.40) |
| 6 | $\ell_2$-norm - I | 24.58 | 6.45M (05.51) | 1.11B (09.77) |
|  | $\ell_2$-norm - V | 24.24 | 5.26M (22.91) | 0.74B (39.91) |
|  | $\ell_2$-norm - VII | 22.60 | 4.66M (26.20) | 0.54B (56.09) |
| 7 | $\ell_2$-norm + $C_2$ - I | 25.05 | 6.45M (05.51) | 1.11B (09.77) |
|  | $\ell_2$-norm + $C_2$ - V | 24.28 | 5.26M (22.91) | 0.74B (39.91) |
|  | $\ell_2$-norm + $C_2$ - XI | 23.82 | 4.68M (31.47) | 0.52B (57.72) |
| 8 | CFP with Optimization [6] - I | 23.84 | 6.81M (00.02) | 1.21B (01.62) |
|  | CFP with Optimization [6] - V | 23.43 | 6.79M (00.05) | 1.15B (06.50) |
| 9 | HFP - I | 25.14 | 6.64M (02.78) | 1.17B (04.87) |
|  | HFP - V | 25.08 | 5.97M (12.59) | 0.96B (21.95) |
|  | HFP - VII | 23.00 | 5.30M (22.40) | 0.75B (39.02) |
| 10 | $\ell_2$-norm + $C_2$ - XV | 24.64 | 4.45M (34.81) | 0.49B (59.71) |
|  | $\ell_2$-norm + $C_2$ - XXII | 24.79 | 4.18M (38.67) | 0.43B (65.08) |
|  | $\ell_2$-norm + $C_2$ - XXV | 23.96 | **4.09M (40.11)** | **0.40B (67.03)** |
| 11 | HFP + $C_2$ - I | 25.49 | 6.64M (02.78) | 1.17B (04.87) |
|  | HFP + $C_2$ - V | 28.00 | 5.97M (12.59) | 0.96B (21.95) |
|  | HFP + $C_2$ - XI | **28.58** | 5.18M (24.15) | 0.72B (41.46) |
|  | HFP + $C_2$ - XVIII | 24.49 | 4.50M (34.11) | 0.52B (57.72) |

In this paper, we attempt to prune the filters that are both redundant and insignificant for the given task. More concretely, the proposed hybrid filter pruning method, partitions the filters of a convolutional layer into two sets, namely, i) significant filters ii) insignificant filters. This partition is performed based on the median of the filter's $\ell_1$-norm of a layer such that the filters having $\ell_1$-norm more than or equal to the median of the filter's $\ell_1$-norm of the same layer will be formed as significant filters. The remaining filters are considered as insignificant filters. The abstract view of the proposed hybrid filter pruning method is shown in Fig. 2.

We prune the top 5% of less important filters from insignificant filters set and one filter from each redundant filter pair which are having a correlation coefficient above 0.9 (absolute value). We utilize Pearson's correlation coefficient as a metric to form redundant filter pairs as in [6]. We empirically observe that pruning the filters which are redundant and least significant from MobileNetv1 not only produces better results but also enables to achieve higher compression compared to CFP [6].

The regularization techniques such as dropout [17], $\ell_2$ regularization [18] are widely used techniques to increase the model's generalization capacity. However, the $\ell_2$ regularization limits us to prune more filters from the model. To overcome this problem, we develop simple but effective regularizers that aim at maximizing the $\ell_1$/$\ell_2$ norms of the remaining filters after pruning. This technique enables the pruned model to have the optimal number of filters that are very significant for the given problem. Next, we formally describe the custom regularizers proposed in this paper.

## 2.3. Custom regularizers

The conventional magnitude-based ($\ell_1$/$\ell_2$-norm) filter pruning methods and redundant filter pruning (CFP [6]) methods can prune a limited amount of filters from CNN. However, if we want to prune more filters, the performance of the model should be compromised to a larger extent. For instance, in DMS applications there is a lot of demand for building compact CNN models for face detection without compromising the performance. To reduce the computational and memory requirement, in this paper, we propose two custom regularizers, namely, $C_1$, $C_2$.

To the $\ell_1$/$\ell_2$-norm based filter pruning methods, we employ a custom regularizer that tries to maximize the $\ell_1$/$\ell_2$-norms of the remaining filters (after pruning), respectively. At each pruning iteration, top $M\%$ (in our experiments, we prune 5% of filters from each layer) of the filters having the least $\ell_1$/$\ell_2$-norm are pruned from the network. Let us consider $R_i$ indicates the set containing the remaining filters across all the layers after pruning. The $\ell_1$-norm custom regularizer is defined as follows,

$$C_1 = -\left( \sum_{\forall f_k \in R_i}^{M} \ell_1(f_k) \right) \tag{2}$$

here $k = 1, 2, ...M$, assuming $M$ filters are remaining in the CNN after filter pruning. The negative sign in Eq. 2 indicates that the objective of custom regularizer is to maximize the $\ell_1$-norm of remaining filters.

Similar to $\ell_1$-norm based filter pruning, we employ a custom regularizer that aims at maximizing the $\ell_2$-norm of remaining filters after each pruning iteration. Mathematically, the regularizer is given by,

$$C_2 = -\left( \sum_{\forall f_k \in R_i}^{M} \ell_2(f_k) \right) \tag{3}$$

The pruned models are finetuned using the new loss function in which the custom regularizers given in Eq. 2, Eq. 3 are added to the loss function (Eq. 1) for $\ell_1$-norm, $\ell_2$-norm based filter pruning methods, respectively. We refer these regularizers as $C_1$, $C_2$ in rest of the paper. Please note that $\ell_2$ regularization term is removed from the objective function (Eq. 1) while finetuning the pruned models using $C_1$, $C_2$ regularizers for $\ell_1$-norm, $\ell_2$-norm based filter pruning methods, respectively. We also employ the custom regularizer $C_2$ to the proposed hybrid filter pruning method which enable to achieve higher compression rate. The custom loss function used for finetuning the pruned models is given as follows,

$$L = \frac{1}{N}\left(L_{conf}(x,c) + \lambda_1 L_{loc}(x,l,g) - \lambda_2 C_i\right) \quad (4)$$

here $i$ is either 1 or 2 represents the employed pruning method. For instance, if we prune the filters based on the $\ell_1$-norm, then the $C_1$ regularizer is utilized. The value of $\lambda_2$ is considered as 0.1 in our experiments.

## 3. RESULTS AND DISCUSSION

### 3.1. Training Details

We consider MobileNetv1 based SSD for conducting the pruning experiments. We utilize MS COCO [19], Wider Face [20] datasets to finetune the base and pruned models, FDDB dataset as the test set to judge the efficacy of these models. The base MobileNetv1 is trained from scratch on MS COCO for 200k steps, which is later finetuned on Wider Face for 100k steps. This model is used as the pre-trained model for filter pruning. The pruned models are finetuned on the MS COCO, Wider face datasets for 50k, 100k steps, respectively.

### 3.2. Observations

The pruning results on MobileNetv1 are reported in Table 1. From Table 1, we can observe that finetuning the $\ell_2$-norm based pruned models using the custom regularizers achieve better performance on MS COCO compared to the pruned models finetuned without custom regularizers. However, we empirically observe that using the conventional $\ell_1$/$\ell_2$-norm based pruned models, there is a significant degradation in the model performance with a little amount of pruning. We can observe the same in block-4 and 6 of Table 1.

It is also observed that redundant filter pruning methods such as CFP [6] prunes less number of filters (few filter pairs have correlation coefficient above the threshold) and also results in higher degradation in the model's performance. However, the proposed HFP method prunes both insignificant and redundant filters at the same time using which we have not only obtained higher compression but also better performance compared to CFP. We have reported the mAP obtained on MS COCO using the pruned models and the reduction in terms of parameters and FLOPs in Table 1. The top metrics in Table 1 are highlighted using bold font. For instance, the proposed HFP method with $C_2$ regularizer achieved the best mAP 28.58 with 24.15%, 41.46% compression in terms of parameters, FLOPs, respectively. This mAP is better than the mAP reported by knowledge distillation methods [10]. Through our experiments, we achieve the highest compression (40.11% of parameters and 67.03% of FLOPs) using XXV - iteration pruned model. The roman letters in Table 1 represent the pruning iteration number.

**Table 2**. Face detection results (recall) are obtained on the FDDB dataset. The inference time, memory requirement of selected pruned models are presented here.

| S.No | Pruning Method | recall (#FPs=500) | Inference Time (in ms) | Memory required (in MB) |
|---|---|---|---|---|
| 1 | base model | 91.85 | 198 | 22 |
| 2 | $\ell_1$-norm - V | 91.28 | 127 | 16 |
| 3 | $\ell_1$-norm + $C_1$ - V | 91.08 | 127 | 16 |
| 4 | $\ell_2$-norm - V | 90.96 | 127 | 16 |
| 5 | $\ell_2$-norm + $C_2$ - V | 91.33 | 127 | 16 |
| 6 | CFP with optimization [6] - V | 90.36 | 171 | 21 |
| 7 | HFP - V | 91.04 | 154 | 19 |
| 8 | $\ell_2$-norm + $C_2$ - XV | 89.40 | 96 | 13 |
| 9 | $\ell_2$-norm + $C_2$ - XXV | 88.78 | 84 | 12 |
| 10 | HFP + $C_2$ - XI | 90.98 | 121 | 15 |
| 11 | HFP + $C_2$ - XVIII | 89.97 | 96 | 13 |

The performance of the pruned models is evaluated on FDDB dataset [21] in terms of recall considering 500 false positives. We have used Renesas R-Car H3 (which has two processors ARM® CortexTM-A57 Quad-core and ARM® CortexTM-A53 Quad-core and 4 GB RAM) board to compute the inference time of the base and pruned models which are reported in Table 2. Pruning the filters using the proposed filter pruning method and finetuning the pruned models with custom regularizer ($C_2$) reduces inference time from 198ms to 96 ms. Similarly, the memory requirement is reduced from 22 MB to 13 MB. We have also employed the pruning methods discussed in Section 2.2 on InceptionResNetv1 [22] based face recognition model and observe similar pruning results.

## 4. CONCLUSION AND FUTURE WORK

The insignificant and redundant filter pruning methods are employed widely to prune the filters from CNNs such as VGG-16, ResNet-50, and so on in which standard convolution operations are used. However, we can not achieve a similar amount of compression for embedded friendly models like MobileNetv1 using these pruning methods. In this paper, we proposed a hybrid filter pruning method in which both insignificant and redundant filters are pruned at the same time (in the same pruning iteration). We also introduced two custom regularizers that help to prune additional filters from MobileNetv1. We have reduced 67.03% of computational and 45.45% of memory requirement of MobileNetv1 for the face detection problem. On an ARM-based device, the inference time is reduced by 57%. From our experiments, we have obtained the pruned models that perform much better (28.00, 28.58) than the existing knowledge distillation methods on MS COCO. We believe that utilizing such models for knowledge distillation may produce even better results.

## 5. REFERENCES

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

[2] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[3] Song Han, Jeff Pool, John Tran, and William J Dally, "Learning both weights and connections for efficient neural networks," *arXiv preprint arXiv:1506.02626*, 2015.

[4] Yann LeCun, John S Denker, and Sara A Solla, "Optimal brain damage," in *Advances in neural information processing systems*, 1990, pp. 598–605.

[5] Hui Miao, Ang Li, Larry S Davis, and Amol Deshpande, "Towards unified data and lifecycle management for deep learning," in *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. IEEE, 2017, pp. 571–582.

[6] Pravendra Singh, Vinay Kumar Verma, Piyush Rai, and Vinay Namboodiri, "Leveraging filter correlations for deep model compression," in *The IEEE Winter Conference on Applications of Computer Vision*, 2020, pp. 835–844.

[7] SH Basha, Mohammad Farazuddin, Viswanath Pulabaigari, Shiv Ram Dubey, and Snehasis Mukherjee, "Deep model compression based on the training history," *arXiv preprint arXiv:2102.00160*, 2021.

[8] Yawei Li, Shuhang Gu, Luc Van Gool, and Radu Timofte, "Learning filter basis for convolutional neural network compression," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 5623–5632.

[9] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf, "Pruning filters for efficient convnets," *arXiv preprint arXiv:1608.08710*, 2016.

[10] Wanwei Wang, Wei Hong, Feng Wang, and Jinke Yu, "Gan-knowledge distillation for one-stage object detection," *IEEE Access*, vol. 8, pp. 60719–60727, 2020.

[11] Jongsoo Park, Sheng Li, Wei Wen, Ping Tak Peter Tang, Hai Li, Yiran Chen, and Pradeep Dubey, "Faster cnns with direct sparse convolutions and guided pruning," *arXiv preprint arXiv:1608.01409*, 2016.

[12] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally, "Eie: Efficient inference engine on compressed deep neural network," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 243–254, 2016.

[13] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang, "Soft filter pruning for accelerating deep convolutional neural networks," *arXiv preprint arXiv:1808.06866*, 2018.

[14] Jingfei Chang, Yang Lu, Ping Xue, Yiqun Xu, and Zhen Wei, "Acp: Automatic channel pruning via clustering and swarm intelligence optimization for cnn," *arXiv preprint arXiv:2101.06407*, 2021.

[15] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[16] Kun Ren, Long Huang, Chunqi Fan, Honggui Han, and Hai Deng, "Real-time traffic sign detection network using ds-detnet and lite fusion fpn," *Journal of Real-Time Image Processing*, pp. 1–11, 2021.

[17] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[18] Jürgen Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[19] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[20] Shuo Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang, "Wider face: A face detection benchmark," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[21] Vidit Jain and Erik Learned-Miller, "Fddb: A benchmark for face detection in unconstrained settings," Tech. Rep., UMass Amherst technical report, 2010.

[22] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-first AAAI conference on artificial intelligence*, 2017.