Improving Efficiency of DeconvNet without sacrificing accuracy

T. Renaud, C. Nivelet Etcheberry
Department of Computer Science, Departement of Electrical Engineering
POSTECH University, Pohang, South Korea
Email: {thomasrenaud, corentin}@postech.ac.kr

***Abstract*—Semantic segmentation is an important part of computer vision research, and since this domain is becoming a more central part of our lives, we need these technologies to be able to perform on even the smallest devices. In this work, we present a new approach to enhance the semantic segmentation efficiency of DeconvNet. Our main contribution is the replacement of the VGG16 backbone with MobileNetV3_small components. With this adjustment, the number of parameters in the model is reduced from 252 million to 12 million, and a non-proportional Intersection over Union (IoU) of about 50% is achieved. However, we also note difficulties in integrating MobileNetV3_small with the DeconvNet design.**

I. Introduction

Semantic segmentation is a branch of computer vision where the objective is to classify each pixel in an image with its corresponding object category. While recent advances in the domain have significantly improved segmentation accuracy, there is still work to do to improve inference time, accuracy, efficiency,... In this context, our work focuses on optimizing the DeconvNet architecture for semantic segmentation by replacing the VGG16 backbone of the original architecture with parts of MobileNetV3_small. The first objective is to do that but preserving the balance between model efficiency and segmentation accuracy.

Traditional approaches to semantic segmentation often involve large parameter counts, leading to resource-intensive models. Our research aims to address this issue by leveraging the efficiency benefits of MobileNetV3_small within the DeconvNet framework.

II. Related Work

A. VGG16

VGG16, proposed by Simonyan and Zisserman [1], is a deep convolutional neural network architecture known for its simplicity and effectiveness in image classification tasks. Its design comprises consecutive convolutional layers with small receptive fields and max-pooling layers, leading to a deep and expressive model. Despite its success, the high parameter count in VGG16 makes it computationally demanding, hence the need of alternative architectures for more efficient solutions.

B. MobileNetV3

MobileNetV3 is a family of lightweight neural network architectures optimized for mobile and edge devices [2]. It employs a combination of depthwise separable convolutions, linear bottlenecks, and inverted residuals to achieve a good balance between model size and accuracy. MobileNetV3 variants, especially the small version, have demonstrated state-of-the-art performance in various vision tasks while being computationally efficient.

C. DeconvNet

DeconvNet, short for Deconvolutional Network, is a convolutional neural network architecture designed for end to end semantic segmentation [3].It utilizes deconvolutional layers to upsample feature maps and refine segmentation boundaries. DeconvNet is used for dense prediction tasks due to its ability to capture fine details in the upsampling process.

III. Methodology

A. Dataset Presentation

The VOC (Pascal VOC) dataset is a widely used benchmark for semantic segmentation tasks. It spans multiple object categories, including but not limited to aeroplanes, bicycles, birds, and more. [1] Each image in the dataset is accompanied by detailed pixel-level annotations, making it a valuable resource for training and evaluating semantic segmentation models.

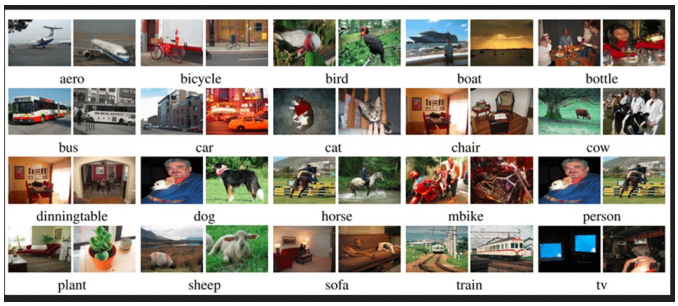


Fig. 1. Some examples of images from the VOC dataset.

B. Dataset Preprocessing and Augmentation

Effective preprocessing and augmentation are essential for training robust semantic segmentation models on the VOC dataset, due to to the limitation in extensive labeled dataset. The implemented pipeline includes several techniques to enhance model generalization.

- **Random Crop:** Randomly crops the input image and mask to introduce spatial variability during training.
- **Random Horizontal Flip:** Applies horizontal flips with a certain probability to augment the dataset's diversity.
- **Resize:** Resizes both input images and masks to a specified size to fit model input size.

These preprocessing and augmentation strategies aim to expose the model to various perspectives and variations within the dataset, facilitating improved performance and generalization.

C. Training pipeline

We present a description of our training loop :

1) **Initialization:** Initialize the model, dataset, loss function, optimizer, and other parameters.
2) **Training Loop:** Iterate through X epochs:
   a) Set the model to training mode.
   b) Iterate through batches in the training dataset:
      - Forward pass: Calculate model predictions.
      - Loss : Compute the loss between predictions and ground truth.
      - Backward pass: Compute gradients and perform optimization.
   c) Calculate average training loss for the epoch.
   d) Set the model to evaluation mode.
   e) Iterate through batches in the validation dataset:
      - Forward pass: Calculate model predictions.
      - Loss computation: Compute the loss between predictions and validation ground truth.
      - Update counters for true positives, false positives, and false negatives for IoU calculation.
   f) Calculate average validation loss and IoU.
   g) Early stopping check: If IoU loss does not improve, stop training.
   h) Save model and optimizer checkpoint if it is the best so far (with respect to IoU loss).
3) **Finalization:** Log final results and clean up.

D. Architecture

TABLE I
OUR MODEL SUMMARY

Layer	Output Size	Composition
Conv1	[batch, 16, 112, 112]	1 Conv2d layer
Conv2	[batch, 24, 28, 28]	3 Inverted Residuals
Conv3	[batch, 40, 14, 14]	3 Inverted Residuals
Conv4	[batch, 96, 7, 7]	3 Inverted Residuals
Conv5	[batch, 576, 7, 7]	3 Inverted Residuals
Conv67	[batch, 1000, 1, 1]	1 Conv2d + 1 Conv2d
Deconv67	[batch, 576, 7, 7]	1 ConvTranspose2d
Deconv5	[batch, 96, 7, 7]	3 ConvTranspose2d
Deconv4	[batch, 40, 7, 7]	3 ConvTranspose2d
Deconv3	[batch, 24, 14, 14]	3 ConvTranspose2d
Deconv2	[batch, 16, 28, 28]	3 ConvTranspose2d
Deconv1	[batch, 21, 112, 112]	3 ConvTranspose2d
Mask	[batch, 21, 224, 224]	Interpolated Deconv1

E. Data Logging

To ensure reproducibility and facilitate analysis, we logged all experimental results using the Weights and Biases (W&B) platform [10]. W&B provides a centralized and collaborative environment for experiment tracking, visualization, and comparison.

The logged data includes:

- **Training and Validation Loss:** Evolution of the loss during training for both the training and validation datasets.
- **IoU (Intersection over Union):** Performance metric measuring the overlap between predicted and ground truth masks.
- **Model Architecture and Parameters:** Detailed information on the model architecture and hyperparameters.
- **Training Configurations:** Settings such as learning rate, batch size, and augmentation techniques.
- **Visualizations:** Examples of predicted masks on input images for qualitative assessment.
- **System Metrics:** GPU usage, memory consumption, and other system-related metrics during training.

This comprehensive logging approach facilitates easy comparison between different experiments and aids in identifying patterns or anomalies in the training process. The integration with W&B allows for efficient collaboration among team members and provides a clear overview of the entire experimentation lifecycle.

IV. EXPERIMENTS

This section first describes our experiment setup. Then, we analyze and evaluate the proposed solution in various aspects.

A. First Experiment

We provide insights into key hyperparameters (see Table II) such as the number of epochs, optimizer, loss function and scheduler. We present quantitative results from the first experiment (see Figure 2), including training and validation loss, mean Intersection over Union (IoU), and training time per epoch.

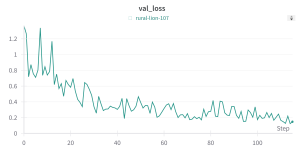
TABLE II
FIRST TRAINING HYPERPARAMETER SPECIFICATION

Parameter/Variable	Description
Number of Epochs	115
Optimizer	Stochastic Gradient Descent
Loss Function	Cross Entropy Loss
Scheduler	-

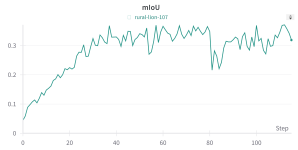
B. Second Experiment

We provide insights into key hyperparameters (see Table III) such as the number of epochs, optimizer, loss function and scheduler. We present quantitative results from the first experiment (see Figure 3), including training and validation loss, mean Intersection over Union (IoU), and training time per epoch.

(a) Training Loss



(b) Validation Loss



(c) Mean Intersection over Union

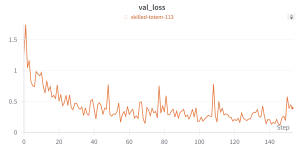
(d) Training time (s)

Fig. 2. Quantitative results from the 1st Experiment

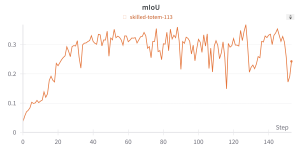
TABLE III
SECOND TRAINING HYPERPARAMETER SPECIFICATION

Parameter/Variable	Description
Number of Epochs	150
Optimizer	Adam
Loss Function	Cross Entropy Loss
Scheduler	Cosine Scheduler

(a) Training Loss



(b) Validation Loss



(c) Mean Intersection over Union

(d) Training time (s)

Fig. 3. Quantitative results from the 2nd Experiment

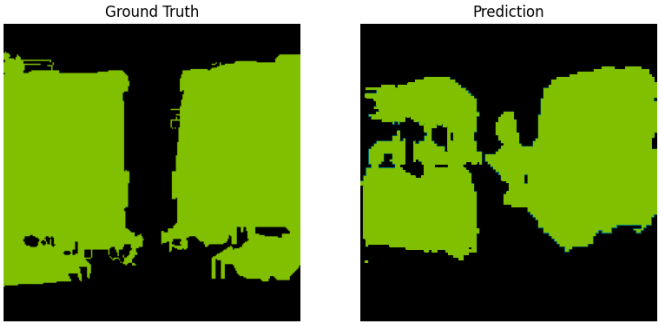
V. DISCUSSION

In this section, we delve into the acquired results and the challenges encountered throughout our project.

A. Parameter Reduction and Challenges

We achieved a remarkable reduction in parameters, exceeding a tenfold decrease. However, the highest attained result, with a Mean IoU of 0.40 (see Table IV), fell below our expectations. Our quest for an improved IoU led us to explore various architectures. Adapting MobileNet to a DeconvNet configuration, especially incorporating new inverted residuals and linear bottlenecks, posed distinct challenges compared to the VGG16 backbone architecture.

(a) Segmentation of a plane



(b) Segmentation of cable cars

Fig. 4. Segmentation results during the second experiment. Inputs have been cropped for data augmentation and normalized.

TABLE IV
EVALUATION RESULTS ON PASCAL VOC 2012 TEST SET

Method	Mean Intersection Over Union (%)
Hypercolumn [9]	59.2
DeconvNet [3]	**69.6**
OURS	40.0

B. Conv67-Deconv67 Transition and Resource Constraints

The pivotal Conv67-Deconv67 layers played a crucial role. Our modified version featured significantly fewer parameters in these layers compared to the original VGG16-based architecture, potentially contributing to the observed performance limitations. To meet our goal of reducing computational complexity, we also reduced several kernel sizes, especially on the Conv67-Deconv67 layers that contained nearly 70 million parameters.

C. Time and Resource Constraints

Resource limitations, notably restricted GPU availability, constrained our experimentation. Despite utilizing GPUs from both laptop and Google Colab resources, their inherent limitations restricted the extent of our exploration. Given the one-month timeframe and this being our first Computer Vision project, the project's full potential could not be fully realized.

VI. CONCLUSION

This project is partly successful. We were able to propose a less computationally complex working model for semantic

segmentation. We demonstrated that it is possible to work to get alternatives to DeconvNet with a VGG-16 backbone. Future plans could involve refining our model, with more time and computational resources.

REFERENCES

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

[2] A. G. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching for mobilenetv3," *arXiv preprint arXiv:1905.02244*, 2019.

[3] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," pp. 1520–1528, 2015.

[4] L. Jing and Y. Tian, "Self-supervised visual feature learning with deep neural networks: A survey," *DOI.org (Datacite)*, 2019. [Online]. Available: https://doi.org/10.48550/ARXIV.1902.06162

[5] J. Dai, K. He, and J. Sun, "Convolutional feature masking for joint object and stuff segmentation," 2015.

[6] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[8] B. Hariharan, P. Arbelaez, L. Bourdev, S. Maji, and J. Malik, "Semantic contours from inverse detectors," 2011.

[9] B. Hariharan and et al., "Hypercolumns for object segmentation and fine-grained localization," 2014, dOI.org (Datacite), https://doi.org/10.48550/ARXIV.1411.5752.

[10] Weights and Biases, "Weights and biases."