20XX-01-XXXX

Improving Feature Learning in Deep Neural Networks through Multi-Class Pretraining and Fine-Tuning

Author: Mr. Sudhanshu Gupta, Johndeere

Abstract

Deep neural networks have achieved remarkable success in various domains by effectively learning feature representations. In this paper, we propose a novel approach to enhance feature learning in deep neural networks by leveraging a multi-class pretraining and fine-tuning strategy. Our key idea is to train the base model on a larger number of classes, allowing it to learn more generalized feature vectors from a diverse set of data. By exposing the model to a broader range of visual patterns and concepts, we aim to facilitate the extraction of robust and discriminative features. Following the multi-class pre training phase, we perform fine-tuning on a smaller subset of classes relevant to the specific application at hand. This fine-tuning process enables the model to specialize and adapt its learned features to the target domain, enhancing its performance in specific tasks. Furthermore, we analyze the learned feature representations using visualization techniques and provide insights into how the pretraining and fine-tuning stages contribute to feature enrichment and transferability. Overall, our proposed approach of training a base model on a larger number of classes followed by fine-tuning on a specific application's classes offers a practical and effective framework to enhance feature learning in deep neural networks. The experimental results demonstrate the potential for improved performance and generalization capabilities, making it a valuable technique for various real-world applications.

Introduction

DNNs have developed into extremely complex architectures over the last decade that can extract complex representations from vast amounts of data. These multi-layered networks are sufficiently advanced to capture hierarchical information, which makes them very useful for handling complicated jobs. One specialized class of DNNs, known as CNNs, has emerged as a game-changer in the domain of computer vision. CNNs are designed to process and analyze visual data, making them particularly effective for image-related tasks. In the last two years, deep convolutional networks have outperformed the state of the art in many visual recognition tasks, e.g. [15]. Their architecture incorporates convolutional layers that enable the network to automatically and adaptively learn spatial hierarchies of features. CNNs differ from traditional neural networks through the inclusion of convolutional layers. These layers convolve over input data using filters or kernels so that the network can recognize spatial dependencies and local patterns. Later pooling layers preserve important features while shrinking spatial dimensions [22]. CNNs excel in feature extraction, automatically learning hierarchical representations of visual elements. This property makes them well-suited for tasks such as image recognition, object detection, and, particularly relevant to this paper, image segmentation. Image segmentation is very essential and critical to image processing and pattern recognition. Despite the successes of DNNs and CNNs, image segmentation presents unique challenges. Overcoming issues such as boundary ambiguity, occlusion, and variability in object appearance requires tailored solutions[12]. Semantic segmentation and instance segmentation are two advanced strategies that try to solve these problems. Applications for image segmentation can be found in a wide range of fields, from autonomous vehicles to medical imaging to tumor detection. The effectiveness of these applications is directly impacted by the accuracy and efficiency of segmentation algorithms.

As per [7], Segmentation approaches are categorized into four classes: pixel based segmentation, area based segmentation, edge based segmentation and physics based segmentation. In this paper we focus on pixel based segmentation and how to increase the feature vector learning of image segmentation networks. Converting any required information into feature vectors is essential because machine learning models can only handle numerical values. Feature vectors are multi-dimensional numerical values that represent features utilized by machine learning algorithms. An ordered list of the numerical characteristics of things seen is called a feature vector. It symbolizes input features for a prediction-making machine learning model. As mentioned in [9] It has been noted that convolutional kernels in a conventional CNN have a tendency to produce activation maps pertaining to specific object properties after training. Considering the nature of activations, segmentation masks of features particular to an object can be understood. Hence, this output activation matrix already has the key to creating requirement-specific segmentation. This CNN feature is used by the majority of image segmentation algorithms to build the segmentation masks that are needed to solve the problem.

Our goal in this research is to improve the quality of the feature vectors, or activation matrix, that the image segmentation model learns. The first stage is to determine the cause of the activation matrix or polished feature vector’s learning constraint. We rigorously investigate, quantitatively assess, and conclude that the model cannot learn ideal feature vectors if there is data imbalance in our training datasets. As discussed in [16], class imbalance poses a challenge for developing unbiased, accurate predictive models. Specifically, poor generalization may result from image segmentation neural networks overfitting to foreground samples from small structures, which are frequently severely underrepresented in the training set. The figure 3.1.1 gives a rough estimate of class imbalance that exists in our training dataset. The graph clearly shows that the obstruction class is n times larger than the other classes causing the model to overfit on some of these classes. A key contribution as represented in [16] is the

However, we do not address the issue of data imbalance in this paper. By training the model on a variety of classes and transferring the significant weights from the trained model to a model whose output is fewer classes than the original one, as required by the application, we attempt to make the model learn the feature vectors more robustly. Through the process of transfer learning, which involves optimizing pre-trained models on massive datasets for particular tasks, CNN efficacy is further increased. This method reduces the requirement for enormous volumes of annotated data and speeds up the creation of high-performing models.

We conduct extensive experiments on various datasets and demonstrate that our approach improves feature learning and boosts the overall performance of image segmentation.

2. Literature Review

2.1 Convolutional Neural Network for image segmentation.

Convolutional Neural Networks (CNNs) have made significant strides in image segmentation, a crucial problem in computer vision. The goal of this literature review is to present a thorough summary of research using CNNs for image segmentation applications. The overview looks at how CNN designs have changed over time, how they are used in different fields, and what advances and problems are motivating researchers in this area.

LeNet-5 [37] laid the foundation for CNNs. Although initially designed for handwritten digit recognition, its convolutional and pooling layers became fundamental for later segmentation architectures. [24] U-Net revolutionized medical image segmentation. Its encoder-decoder architecture, with skip connections, allows precise localization and segmentation of structures in medical images. [19]FCN, proposed by Long et al. in 2015, extended CNNs for end-to-end segmentation. FCNs introduced the concept of pixel-wise classification, enabling segmentation maps with dense predictions. The application of CNNs are progressing, especially in the fields of Medical Image Segmentation, Remote Sensing, Autonomous Vehicles, Natural Scene Segmentation, Industrial Inspection etc. As CNN architectures evolve, there is a need for dynamic visualization techniques that adapt to the increasing complexity of models. Exploring real-time and interactive visualization methods is an avenue for future research.

2.2 Transfer Learning

In the field of deep learning, transfer learning has become a key technique that offers a strong framework for using information from one domain to improve performance in another. This review of the literature is on the use of transfer learning, particularly on how it advances image segmentation tasks. [8]Pre-training a model on a source task and then modifying it for a target task constitutes transfer learning. The incentive is to apply the information gained from a related field to enhance performance on an entirely new, maybe data-poor, or distinct task. Image segmentation faces challenges such as limited labeled data, class imbalance, and domain shifts. Transfer learning aims to mitigate these challenges by leveraging knowledge from well-labeled and diverse source domains.

There are different ways to utilize and implement the transfer learning techniques such as Feature Extraction and Fine-Tuning, Domain Adaptation Techniques: where domain-specific normalization methods aim to align feature distributions across domains.

2.3 Visualizing feature maps

The visualization of feature vectors in Convolutional Neural Networks (CNNs) plays a crucial role in understanding the representations learned by these complex models. This literature review explores the methodologies, techniques, and applications of visualizing feature vectors in CNNs, with a particular focus on insights gained in the context of image segmentation. [32, 25]CNNs consist of layers that progressively extract hierarchical features from input data. Feature vectors represent these learned features, capturing both low-level and high-level visual information. [2, 3] explore the study of the feature vectors, kernels, analyzing the weights during the training cycle. Kernels, known for their ability to implicitly represent high-dimensional spaces, are employed to enhance the efficiency and effectiveness of feature vector operations. The primary objectives include selecting relevant features and projecting them onto a lower-dimensional space. [25] explores methodologies for gaining insights into the internal mechanisms of DNNs. Techniques for analyzing activation patterns, feature representations, and model behavior are discussed, providing a foundation for understanding the learned representations.

2.4 Challenges and Innovations

Data Imbalance: [11]Data class imbalance is a common challenge in image segmentation tasks, where certain classes may be underrepresented, leading to biased model training. It's still difficult to address imbalances in training data. Various methods have been suggested to address this problem, including class weighting, data augmentation, and customized loss functions.The review discusses how class imbalance can result in models favoring dominant classes, causing misclassification and poor segmentation of minority classes[16]. This understanding forms the basis for the need to address imbalance in segmentation datasets. There are different ways to tackle the imbalance problems, such as

- Class weighting involves assigning different weights to classes during training to give more emphasis to underrepresented classes.
- Resampling techniques, such as oversampling minority classes or undersampling majority classes, are discussed.
- Pre-trained models on large datasets can be fine-tuned on imbalanced segmentation datasets, leveraging knowledge from diverse sources.

3. Methodology

We wanted an encoder-decoder architectural model for this hypothesis. [10]. Our architecture is a modified version of U-net [24]. Because the bottleneck in the architecture symbolizes or provides a complete conversion of the picture to the feature vector or activation matrix, it is particularly suitable for the hypothesis that we put forward. We can better comprehend the alterations brought about by the varied amount of training classes with the aid of this architecture. The model architecture in its simplified form is depicted in Figure 3.4.1. A contracting path (encoder) is followed by an expansive path (decoder) in the U-Net design. While the expansive path recovers spatial information, the contracting path captures context and abstract features. [35] There features are described in short below:

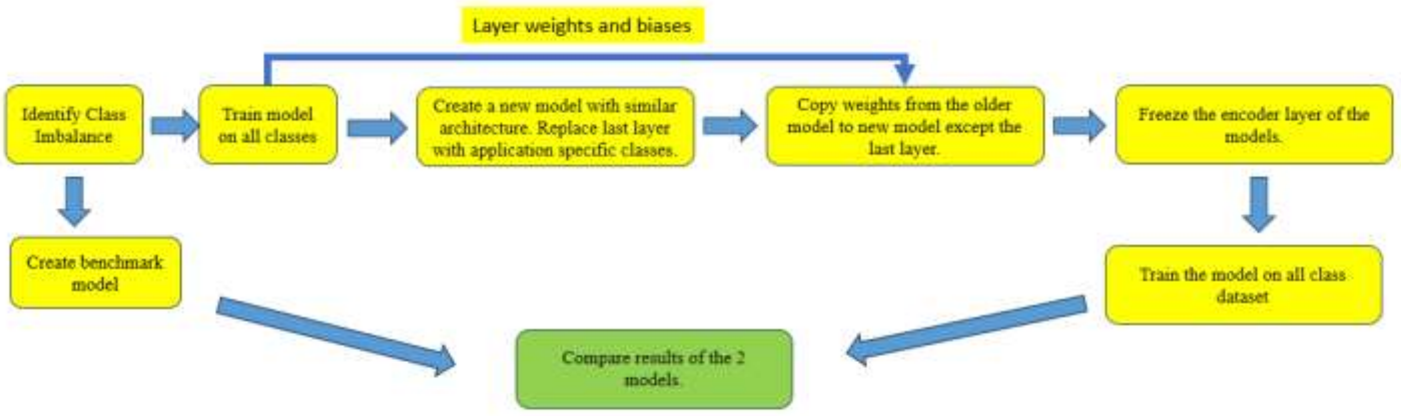
Layer weights and biases
Identify Class Imbalance
Train model on all classes
Create a new model with similar architecture. Replace last layer with application specific classes.
Copy weights from the older model to new model except the last layer.
Freeze the encoder layer of the models.
Create benchmark model
Train the model on all class dataset
Compare results of the 2 models.

Figure 3.1 : Shows a basic chart of the steps that were carried out for this research

Contracting Path (Encoder):

- To minimize spatial dimensions, use convolutional layers using max-pooling procedures.
- With every stage, more abstract feature vectors are captured.

- Bottleneck:
  - A bottleneck layer that retains high-level features.
- Expansive Path (Decoder):
  - To improve spatial resolution, use up sampling techniques or transposed convolutions.
  - Skip connections that help with accurate localization by concatenating characteristics from the respective encoder layers.

This study will not discuss the training dataset because it is proprietary. We use the SODA dataset and reference [3] to review and assess this work. These unlabeled images are utilized for validation. In this section we will describe the whole process of the experiment. For simplifying the process, we will divide it into the following steps.

3.1. Identifying class imbalance

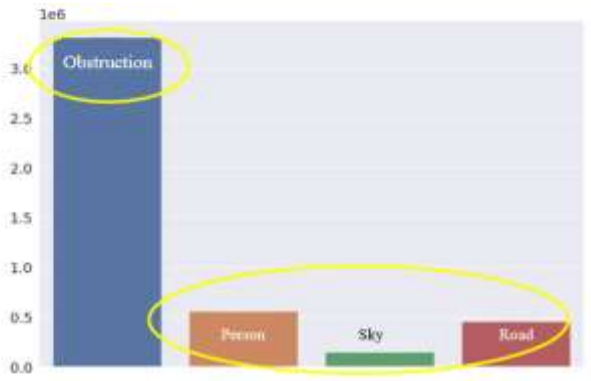

Figure 3.1.1: The Figure explains the class imbalance in our training dataset. The graph shows the distribution of classes when we combine them as required by the application.

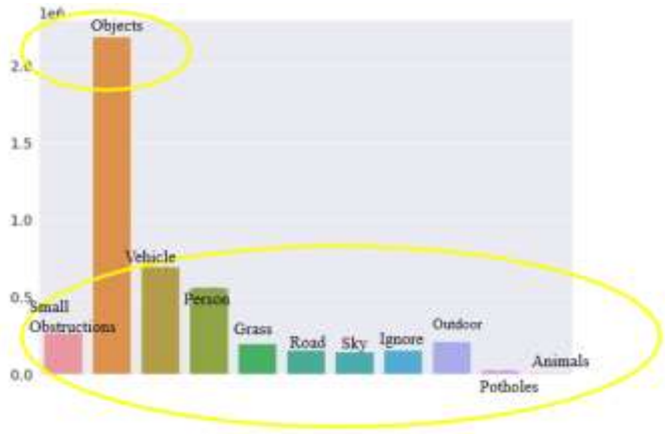

Figure 3.1.2: The graph shows 11 levels of classes available in the dataset. It clearly denotes the class imbalance that occurs.

We needed to know why there were additional false positives and false negatives in the model before we jumped to a solution. The class imbalance for obstacles is far greater than those of the other classes, as Figure 3.1.1 illustrates. Compared to other classes, the Obstruction class is over five times larger. This imbalance, which occurs at the instance level, will undoubtedly lead to some overfitting during model training.
As stated in [16], we train our model using focus loss. In the event of a data imbalance, focal loss can assist in learning the feature vectors of a class with additional weighted loss. For the relevant classes, applying additional multiplication factors (such as gamma) can improve feature learning.
From Figure 1 and 2, we can get an idea on the imbalance ratio of the input training dataset.

3.2. Create benchmark model: Train the model with application specific dataset!

We build our own benchmark model by training it on the application-specific labeled dataset to gauge the progress of feature vector interpretation. We merge several classes into this one. The labels that were combined for this training experiment are displayed in Figure 3.2. We use the application-specific labeled dataset to train our model architecture and evaluate its performance to produce a benchmark model. After the training is complete, this can be compared to the newly created final model. We will use the name "benchmark model" for the current model throughout this paper.

Label	Class Name
1.	Person
2.	Objects
3.	Outdoor
4.	Sky
5.	Potholes
6.	Road
7.	Animals
8.	Grass, bush, trees
9.	Small obstructions
10.	Vehicle

Labeled classes available in the dataset

Label	Super Category	Merged with
1.	Person	Person
2.	Obstructions	Vehicle, Objects, Outdoor, Grass, Animal
3.	Road	Road, small obstructions, Potholes
4.	Sky	Sky

Application wise labeling, derived from all available classes

Figure 3.2 : The table on the left shows all 11 classes(11th : ignore) that are labeled and available. The table on the right shows combined labels which are required by application.

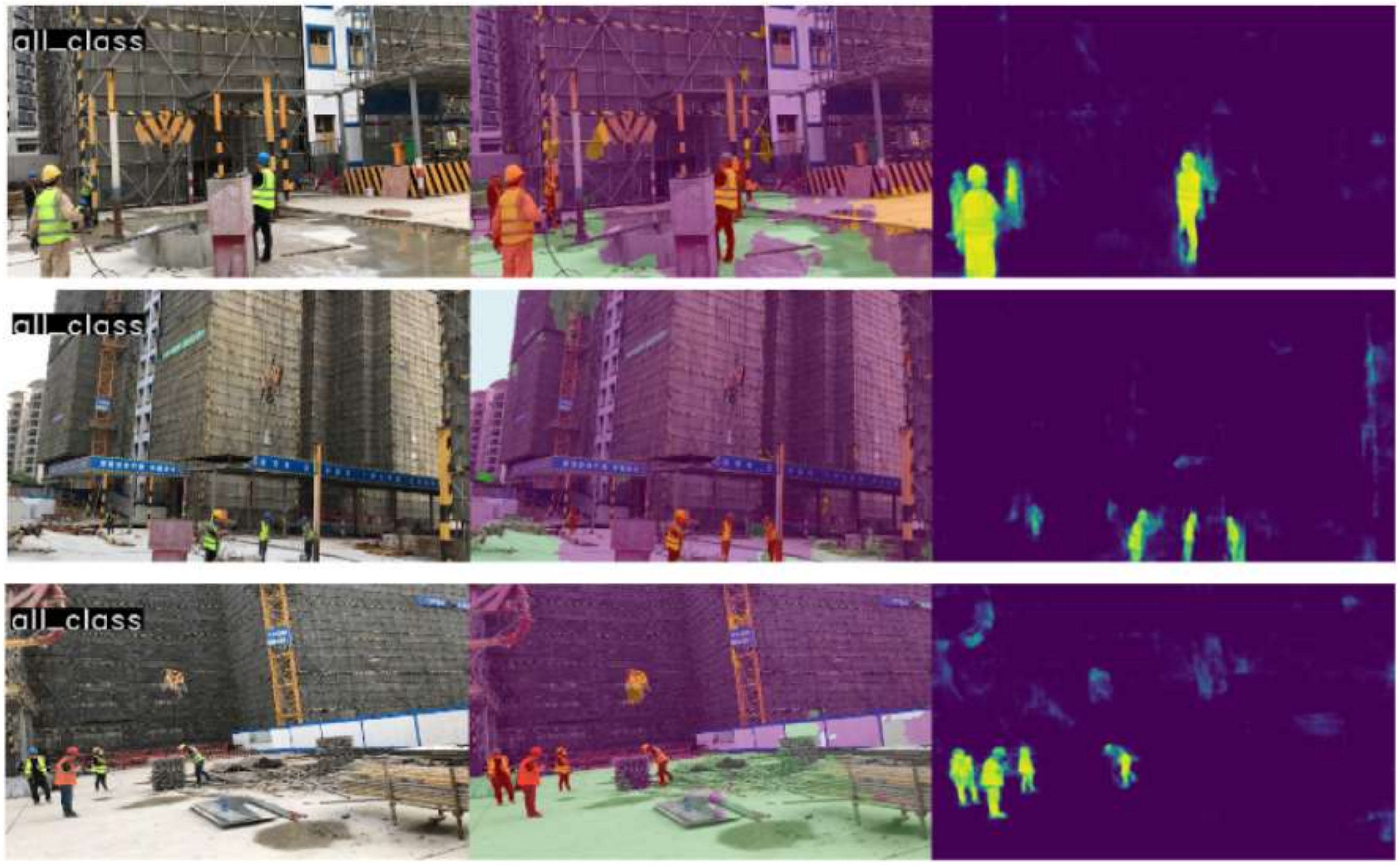

Figure 3.3.1: The original image can be seen in the image on the left. The output of the detections superimposed on the original image is seen in the middle image. The confidence(output) in human segmentation at each pixel position is shown in the image.

3.3. Train the model with all class labeled dataset.

The goal of using all labels is to provide the model the opportunity to train on a very diverse dataset, which will produce a unique activation matrix for every labeled obstacle that can be identified in the pictures. We use a dataset (labels explained in Fig. 3.2) to train the model without combining the classes. We'll refer to this dataset as the "all class dataset" for simplicity's sake. To comprehend the state of the model, we also monitor the logit output from the model and visualize it. The output that the model predicts after being successfully trained is depicted in the figure 3.3.1.

3.4. Fine-tune all class models to give application specific output.

Now that we have a model that has been trained on all classes, the weights and biases of the model layers have been learned in a way that produces a feature vector map that is detailed for 11 classes.
The following steps define the most important steps of this research.

- We replace the last layer of the model so that it can be adjusted for application specific output.
- Then we freeze the encoder side of the model so that in back-propagation, the encoder should not adapt to the application specific labeled feature maps. By doing so, we can achieve the state where the activation map presented is detailed, diverse for 11 classes.
- Now, by using transfer-learning and training the model for a few more steps, we get a model with a decoder which has learnt to map the 11-class activation map to the application specific 4 classes.

By doing so, we can get the encoder's feature map which relates to diverse labels, but the decoder can map it back to application specific features. This provides a diverse segmentation space threshold where the space for every label has widened and chance of overfit has reduced. Throughout the paper we will refer to this model as "Fine-tuned model"

4. Results

4.1 Visualizing the output of the benchmark model and fine-tuned model..

The outcome of improving the model's feature vector knowledge is evident in the above figure. When compared to the benchmark model, the fine-tuned model exhibits comparatively higher accurate segmentations.
It should be noted that after training the model on a variety of obstruction kinds and fine-tuning the decoder, we are achieving higher detection performance since the abundance of the obstruction class led to overfitting of the obstruction class. To provide readers with a more thorough overview of the enhancement, we have included additional specific images in the Appendix.

4.2 Visualizing the kernels/weights.

We have changed our code to capture the activation feature map after each layer, as shown in figure 4.2, to better comprehend the improvements in the feature vectors or activation maps.

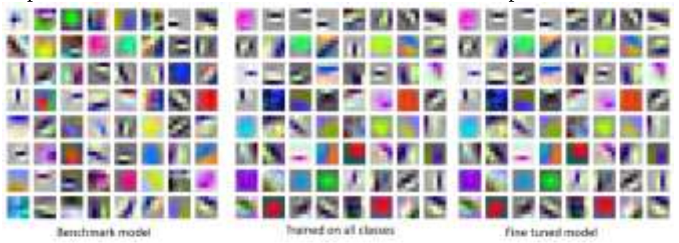

Figure 4.2.1: The images are a representation of the kernels (weights) in the first layer of our trained models. Left most images is the visualization of weights in the first layer of our benchmark model.
Middle image represents the weights of the model initially trained on all classes. Rightmost image represents the weights of the fine-tuned model.

It should be noted that the weights/kernel represented for the "Trained on all class" model and the "Fine-tuned model" are identical because we fixed the encoder side of the model's weights. We depict the feature vector maps for the first layer, as shown in the image, in order to develop an understanding at the feature level.
The results are from the kernel that are achieved from the fine-tuned model. The same kernel can be found on the 4th row, 2nd column of the kernel matrix from figure above.
This figure demonstrates how we map the output and assess feature vector improvements by contrasting the outcomes with the benchmark model. Since it will be confusing to compare these hidden layers visually, we have included the visualization of the intermediate activation maps in the appendix. In order to compare the outcomes, we will take the initial two layers and the last two layers.

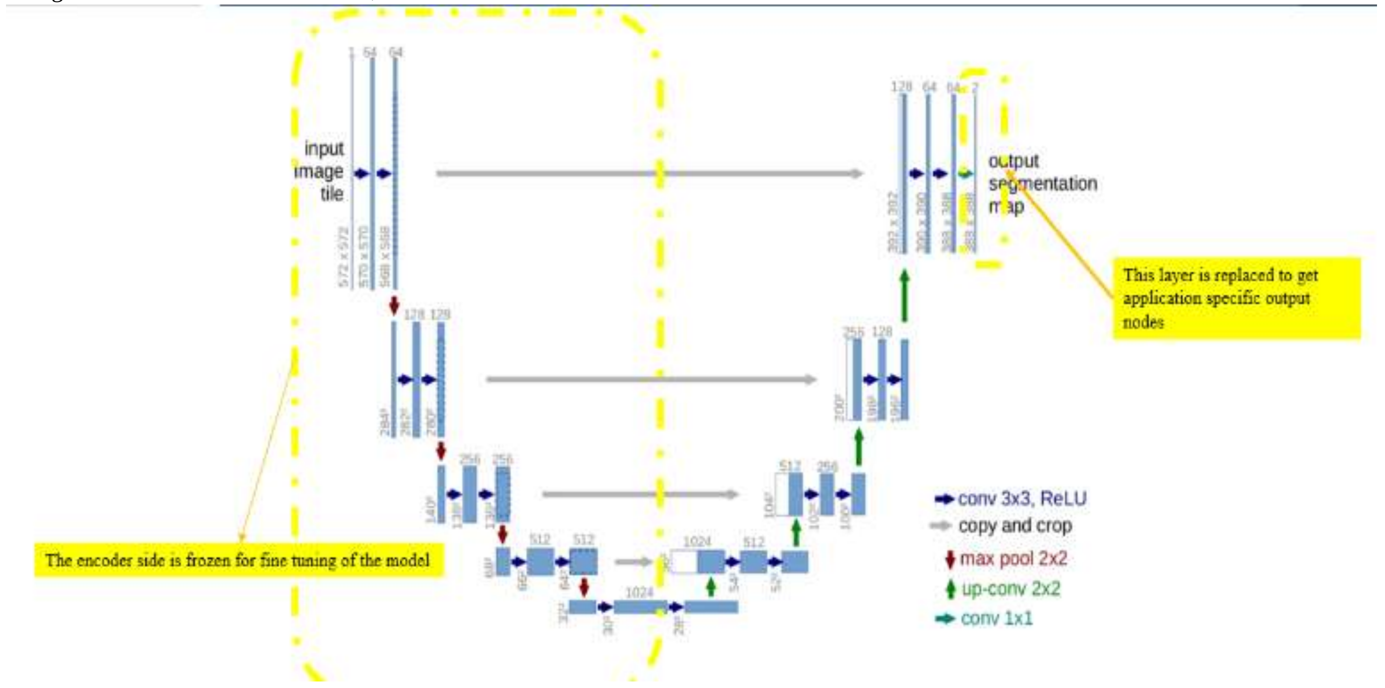
input image tile
output segmentation map
This layer is replaced to get application specific output nodes
The encoder side is frozen for fine tuning of the model
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

Figure 3.4.1: The image shows a rough estimate of the [24] U-net architecture that we modify and use for our experiment. The layers marked in yellow dashed box represent the encoder of the model and the remaining represents the decoder of the model.

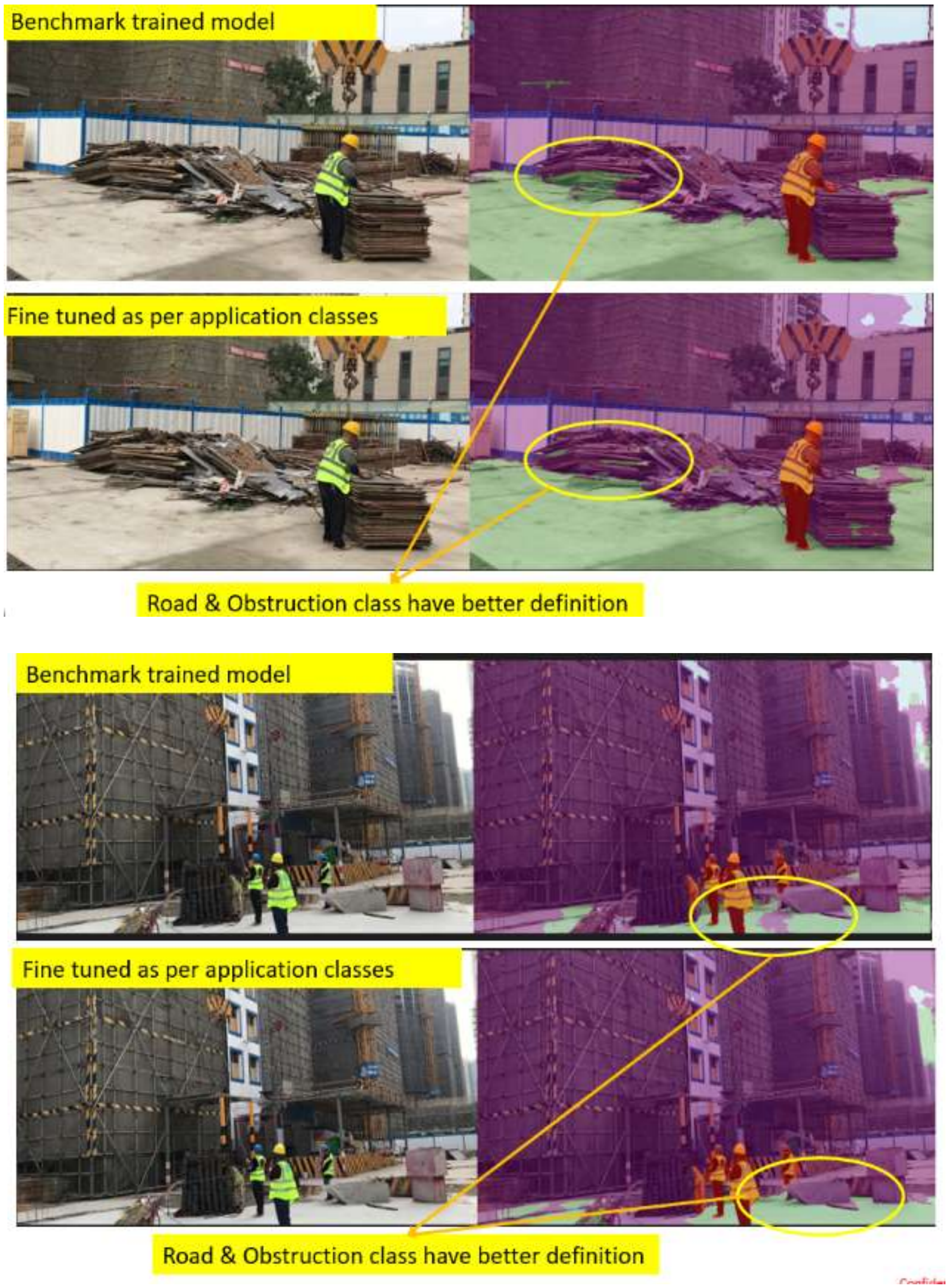
Benchmark trained model
Fine tuned as per application classes
Road & Obstruction class have better definition
Benchmark trained model
Fine tuned as per application classes
Road & Obstruction class have better definition

Figure 4.1.1 : The images show output of the 2 models on a test dataset. Red denotes person detection. Green denoted Road, purple denotes obstruction.

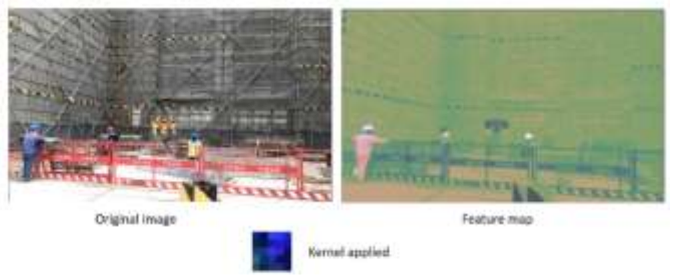

Figure 4.1.2 : The image shows the original image(on the left) and the output feature map(on the right). The middle image shows the filter(kernel/weights) applied to get the output.

.

4.2 Analyzing quantitative outputs

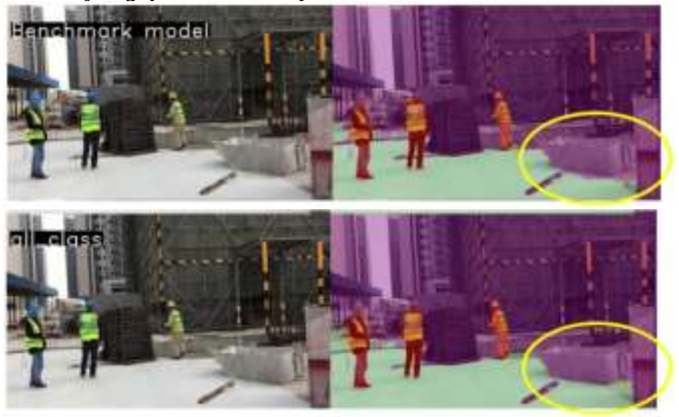

Figure 4.2.1: shows the output from the 2 models, which have similar semantic detections, but the benchmark model has a disputed obstruction detection on the object highlighted on the right.

The figure shows semantic output of an image from a construction site [36]. The comparison shows a similar output but has a dispersed detection on the obstacle on the right. Throughout the activation maps, we consider that region and visualize how the feature understanding of the object has improved.

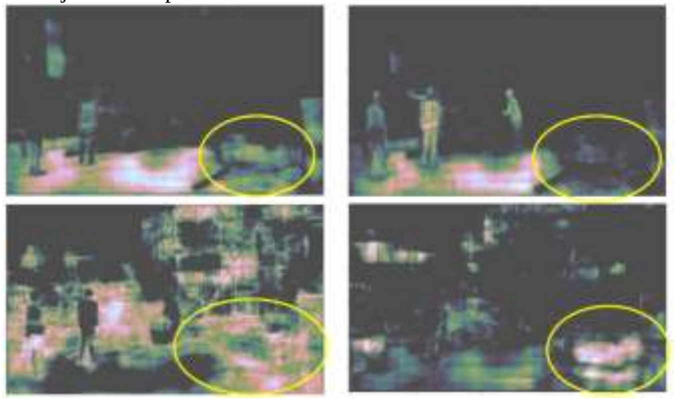

Figure 4.2.2 : Represents activation maps from the benchmark model. The explanation for the scattered obstacle detection in figure 4.2.2 is suggested by the activation maps in the image below. The image displays a few pixel locations that belong to the obstacle class's false positive pixel regions and are likewise activated.

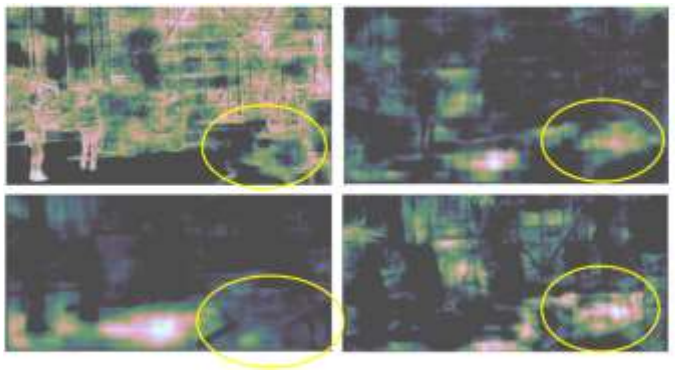

Figure 4.2.3 : Represents activation maps from the fine tuned model.

The figure's activation maps provide an indication of why the obstruction class's semantic segmentation has improved. The distinction between the benchmark model and an improved and enhanced feature vector learning is displayed in the two activation maps above. This provides credence to the idea that we can improve the feature vector learning of a segmentation model by first training it on a variety of labeled datasets and then applying transfer learning approaches.

5. Summary and conclusion

Improved Segmentation Performance:

The fine-tuning of the decoder side resulted in improved image segmentation performance, particularly for the 4 classes relevant to the application of interest. The model exhibited better discrimination and accuracy in identifying and segmenting instances of these specific classes.

Enhanced Feature Vectors:

The feature vectors extracted from the model demonstrated improvement, indicating that the fine-tuning process allowed the network to learn more discriminative and contextually relevant features for the targeted application.

Visualization of Feature Vectors:

Feature vectors from various layers of the network were visualized to gain insights into the representation learned by the model. This visualization provided a deeper understanding of how the features evolved across different layers, highlighting the hierarchical and abstract nature of the learned representations.

Transfer learning has a tremendous potential in the future since it saves training time and resources. Here we offer additional benefits of transfer learning such that features learnt can be tweaked and controlled in order to gain improved performance on image segmentations. The findings demonstrate that by first training the model on a variety of labels and then fine-tuning it according to the needs of the application, we can improve the feature maps of a model. This may contribute to enhancing the model's overall performance. The figure 4.2.2 & 4.2.3 clearly showed the difference in the performance and an improvement in feature mapping.

6. Future Scope

The ideology put forth can be applied not only to image segmentation but various other networks. This ideology might have even significant results if the class imbalance of the training dataset is reduced. Increasing the diversity in labels can increase the feature learning even further.

References

1. Agarwal, N. et al. (2021) “Transfer learning: survey and classification,” in Tiwari, S. et al. (eds.) Smart innovations in communication and computational sciences: proceedings of ICSICCS 2020. Singapore: Springer Singapore (Advances in intelligent systems and computing), pp. 145–155. doi: 10.1007/978-981-15-5345-5_13.

2. Baudat, G. and Anouar, F. (2003) “Feature vector selection and projection using kernels,” Neurocomputing, 55(1–2), pp. 21–38. doi: 10.1016/S0925-2312(03)00429-6.

3. Bellet, A., Habrard, A. and Sebban, M. (2013) “A survey on metric learning for feature vectors and structured data,” arXiv preprint arXiv:1306.6709.

4. Buda, M., Maki, A. and Mazurowski, M. A. (2018) “A systematic study of the class imbalance problem in convolutional neural networks.,” Neural Networks, 106, pp. 249–259. doi: 10.1016/j.neunet.2018.07.011.

5. Chen, Z. et al. (2022) “Class Re-Activation Maps for Weakly-Supervised Semantic Segmentation,” in 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, pp. 959–968. doi: 10.1109/CVPR52688.2022.00104.

6. Deng, W. et al. (2019) “Brain Tumor Segmentation Based on Improved Convolutional Neural Network in Combination with Non-quantifiable Local Texture Feature.,” Journal of medical systems, 43(6), p. 152. doi: 10.1007/s10916-019-1289-2.

7. Deng, Y., Manjunath, B. S. and Shin, H. (1999) “Color image segmentation,” Proceedings. 1999 IEEE ….

8. Furht, B., Villanustre, F. and Weiss, K. (2016) “Transfer learning techniques,” Big data technologies ….

9. Ghosh, S. et al. (2019) “Understanding deep learning techniques for image segmentation,” ACM Computing Surveys, 52(4), pp. 1–35. doi: 10.1145/3329784.

10. Haralick, R. M. and Shapiro, L. G. (1985) “Image segmentation techniques,” Computer Vision, Graphics, and Image Processing, 29(1), pp. 100–132. doi: 10.1016/S0734-189X(85)90153-7.

11. Johnson, J. M. and Khoshgoftaar, T. M. (2019) “Survey on deep learning with class imbalance,” Journal of big data, 6(1), p. 27. doi: 10.1186/s40537-019-0192-5.

12. Kervadec, H. et al. (2021) “Boundary loss for highly unbalanced segmentation.,” Medical Image Analysis, 67, p. 101851. doi: 10.1016/j.media.2020.101851.

13. Korsch, D., Bodesheim, P. and Denzler, J. (2021) “End-to-End Learning of Fisher Vector Encodings for Part Features in Fine-Grained Recognition,” in Bauckhage, C., Gall, J., and Schwing, A. (eds.) Pattern recognition: 43rd DAGM german conference, DAGM GCPR 2021, bonn, germany, september 28 – october 1, 2021, proceedings. Cham: Springer International Publishing (Lecture notes in computer science), pp. 142–158. doi: 10.1007/978-3-030-92659-5_9.

14. Krähenbühl, P. and Koltun, V. (2011) “Efficient inference in fully connected crfs with gaussian edge potentials,” Advances in neural information ….

15. Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2012) “ImageNet classification with deep convolutional neural networks,” Communications of the ACM, 60(6), pp. 84–90. doi: 10.1145/3065386.

16. Li, Z., Kamnitsas, K. and Glocker, B. (2021) “Analyzing overfitting under class imbalance in neural networks for image segmentation.,” IEEE transactions on medical imaging, 40(3), pp. 1065–1077. doi: 10.1109/TMI.2020.3046692.

17. Lin, T.-Y. et al. (2014) “Microsoft COCO: common objects in context,” in Fleet, D. et al. (eds.) European Conference on Computer Vision (ECCV). Cham: Springer International Publishing (Lecture notes in computer science), pp. 740–755. doi: 10.1007/978-3-319-10602-1_48.

18. Lin, T.-Y. et al. (2017) “RetinaNet - Focal loss for dense object detection,” in 2017 IEEE International Conference on Computer Vision (ICCV). 2017 IEEE International Conference on Computer Vision (ICCV), IEEE, pp. 2999–3007. doi: 10.1109/ICCV.2017.324.

19. Long, J., Shelhamer, E. and Darrell, T. (2015) “Fully convolutional networks for semantic segmentation,” in 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, pp. 3431–3440. doi: 10.1109/CVPR.2015.7298965.

20. Minaee, S. et al. (2022) “Image segmentation using deep learning: A survey.,” IEEE transactions on pattern analysis and machine intelligence, 44(7), pp. 3523–3542. doi: 10.1109/TPAMI.2021.3059968.

21. Noh, H., Hong, S. and Han, B. (2015) “Learning deconvolution network for semantic segmentation,” in 2015 IEEE International Conference on Computer Vision (ICCV). 2015 IEEE International Conference on Computer Vision (ICCV), IEEE, pp. 1520–1528. doi: 10.1109/ICCV.2015.178.

22. Pan, T. et al. (2017) “Fully Convolutional Neural Networks with Full-Scale-Features for Semantic Segmentation,” Proceedings of the AAAI Conference on Artificial Intelligence, 31(1). doi: 10.1609/aaai.v31i1.11217.

23. Pinheiro, P. O. and Collobert, R. (2015) “From image-level to pixel-level labeling with Convolutional Networks,” in 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, pp. 1713–1721. doi: 10.1109/CVPR.2015.7298780.

24. Ronneberger, O., Fischer, P. and Brox, T. (2015) “U-Net: Convolutional Networks for Biomedical Image Segmentation,” in Navab, N. et al. (eds.) Medical Image Computing and Computer-Assisted Intervention (MICCAI). Cham: Springer International Publishing (Lecture notes in computer science), pp. 234–241. doi: 10.1007/978-3-319-24574-4_28.

25. Shahroudnejad, A. (2021) “A survey on understanding, visualizations, and explanation of deep neural networks,” arXiv preprint arXiv:2102.01792.

26. Simonyan, K. and Zisserman, A. (2014) “Very deep convolutional networks for large-scale image recognition,” arXiv preprint arXiv:1409.1556.

27. Suranaree University of Technology, Thailand et al. (2020) “Improving the representation of CNN based features by autoencoder for a task of construction material image classification,” Journal of Advances in Information Technology, 11(4), pp. 192–199. doi: 10.12720/jait.11.4.192-199.

28. Vinogradova, K., Dibrov, A. and Myers, G. (2020) “Towards Interpretable Semantic Segmentation via Gradient-Weighted Class Activation Mapping (Student Abstract),” Proceedings of the AAAI Conference on Artificial Intelligence, 34(10), pp. 13943–13944. doi: 10.1609/aaai.v34i10.7244.

29. Wang, D., Hu, G. and Lyu, C. (2021) “FRNet: an end-to-end feature refinement neural network for medical image segmentation,” The Visual computer, 37(5), pp. 1101–1112. doi: 10.1007/s00371-020-01855-z.

30. Weiss, K., Khoshgoftaar, T. M. and Wang, D. (2016) “A survey of transfer learning,” Journal of big data, 3(1), p. 9. doi: 10.1186/s40537-016-0043-6.

31. Xia, X. and Kulis, B. (2017) “W-net: A deep model for fully unsupervised image segmentation,” arXiv preprint arXiv:1711.08506.

32. Zeiler, M. D. and Fergus, R. (2014) “Visualizing and Understanding Convolutional Networks,” in Fleet, D. et al. (eds.) Computer Vision – ECCV 2014. Cham: Springer International Publishing (Lecture Notes in Computer Science), pp. 818–833. doi: 10.1007/978-3-319-10590-1_53.

33. Zeiler, M. D., Taylor, G. W. and Fergus, R. (2011) “Adaptive deconvolutional networks for mid and high level feature learning,” in 2011 International Conference on Computer Vision. 2011 IEEE International Conference on Computer Vision (ICCV), IEEE, pp. 2018–2025. doi: 10.1109/ICCV.2011.6126474.

34. Zhou, B. et al. (2016) “Learning deep features for discriminative localization,” in Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, pp. 2921–2929. doi: 10.1109/CVPR.2016.319.

35. Zhou, Z. et al. (2018) “UNet++: A Nested U-Net Architecture for Medical Image Segmentation.,” Deep learning in medical image analysis and multimodal learning for clinical decision support : 4th International Workshop, DLMIA 2018, and 8th International Workshop, ML-CDS 2018, held in conjunction with MICCAI 2018, Granada, Spain, S..., 11045, pp. 3–11. doi: 10.1007/978-3-030-00889-5_1.

36. Duan, R. et al. (2022) “SODA: A large-scale open site object detection dataset for deep learning in construction,” Automation in Construction, 142, p. 104499. doi: 10.1016/j.autcon.2022.104499.

37. Zhang, J. et al. (2022) “A novel deep LeNet-5 convolutional neural network model for image recognition,” Computer Science and Information Systems, 19(3), pp. 1463–1480. doi: 10.2298/CSIS220120036Z.

Contact Information

Mobile number: +91 9730043246

Email Id : GuptaSudhanshu@johndeere.com

Appendix

A. Implementation Detail

This section contains additional implementation details for our experiment, including data validation, network architecture, and inference procedure.

A.1 Validation dataset

The dataset used for training has proprietary concerns hence , we did not discuss the training dataset. But for the validation of this experiment, we have used SODA open source dataset. Site Object Detection Dataset (SODA), which contains 15 object classes categorized by the worker, material, machine, and layout. >20,000 images were collected from multiple construction sites in different situations, weather conditions, and construction phases, covering different angles and perspectives.

We have to resize the input images from the dataset since the training dataset is of different shape. The reason is:

- Batch normalization normalizes the activations in a layer by adjusting and scaling them. It relies on the assumption that the input data has a consistent shape across the batch. If you pass images of different shapes, it might lead to errors during the batch normalization process.
- To address this, you should ensure that the input images have the same dimensions. We have to either resize or crop the images before passing them through the network.
- Max pooling is a down-sampling operation that reduces the spatial dimensions of the input data. Like batch normalization, max pooling layers also expect a consistent input shape.
- If you provide images with different shapes, it can result in incompatible dimensions during the max pooling operation, potentially causing errors.
- Similarly, you should preprocess your images to have consistent dimensions before passing them through the network.

A.2 Network Architecture

We have built our own custom version of U-net architecture but the ideology still remains the same. Figure 3.4.1 shows a simplified version of the U-Net model. We have on purpose chosen this architecture because the bottleneck and skip connections give out feature vectors which are easy to record and use for understanding the functionality. Let's delve into the details of Figure 3.4.1, a simplified representation of your custom U-Net model:

1. Encoder (Contracting Path): Convolutional Blocks:
    - The contracting path consists of a series of convolutional blocks, each followed by activation layers (commonly ReLU) for feature extraction.
    - These blocks reduce spatial dimensions while capturing hierarchical features.
2. Bottleneck Layer: Central Bottleneck:
    - The bottleneck layer acts as the central hub, capturing the most abstract features from the encoder.
    - It maintains contextual information crucial for accurate feature representation.
3. Decoder (Expansive Path): Transposed Convolutional Blocks:
    - The expansive path uses transposed convolutions to upsample feature maps.
    - Skip connections are concatenated to recover spatial details lost during downsampling.
4. Skip Connections: Role in Feature Preservation:
    - Skip connections link corresponding layers between the encoder and decoder.
    - These connections enable the model to preserve fine-grained details and facilitate gradient flow during backpropagation.
5. Output Layer: Segmentation Map:
    - The final layer produces a segmentation map representing the predicted class labels for each pixel in the input image.
    - We use softmax Activation functions for multiclass segmentation.
6. Feature Vectors: Extraction from Bottleneck and Skip Connections:
    - The bottleneck and skip connections provide distinct feature vectors.
    - These vectors capture both low-level and high-level features, aiding in the interpretability of the model.
7. Training Procedure:
    - The model is trained using a focal loss function for image segmentation tasks
    - During training, the network learns to adjust its parameters to minimize the discrepancy between predicted and ground truth segmentation maps.

8. Inference Procedure:
    - During inference, input images are passed through the trained network to obtain segmentation predictions.
    - Feature vectors from the bottleneck and skip connections can be recorded and analyzed for insights into model decisions.
    - In order to capture the activation map for every layer, we had to modify the model code. We had to make sure the weights and kernel were being correctly returned and captured.

A.3 Transfer learning

The process of transfer learning in our case was not too complicated. We had to disable the learnable configuration of the encoder side of the model in order to freeze the training on these layers. We also had to remove the last layer of the all-class model since the output layer of this model was 10 nodes (classes) whereas the application required only 4 classes. We created a state_dict of the trained model and loaded it into a new model with all the right architecture for application specific requirements. Then with the application specific labeled dataset, we ran training for some epochs.

The purpose of the second iteration training is for the decoder to be able to map the 10 class feature vector to 4 class output or application specific output.

B. Visualizing outputs

We were returning every layer output in order to obtain the activation maps. In addition, we do iterations over every model kernel in order to incorporate all the refined and learned weights. Figure 4.2.2 & Figure 4.2.3 are just handpicked visualized feature maps since visualizing other intermediate layers will not make any sense. To give a broader aspect, we have attached a few examples where we can spot almost all the learned weights. These weights will be in the order of the forward propagation. For these examples, we have taken the fine-tuned model to infer and visualize the vector maps.

- We can visualize the first layer of the network since the first layer adapts to learn information about the edges, color, shapes etc.
- The figure 4.2.1 shows all the kernels that have developed after training the model. This layer is on the encoder side of the model.
- We pick some distinct kernel from the figure that is highlighted and visualize the activation map of those kernels.

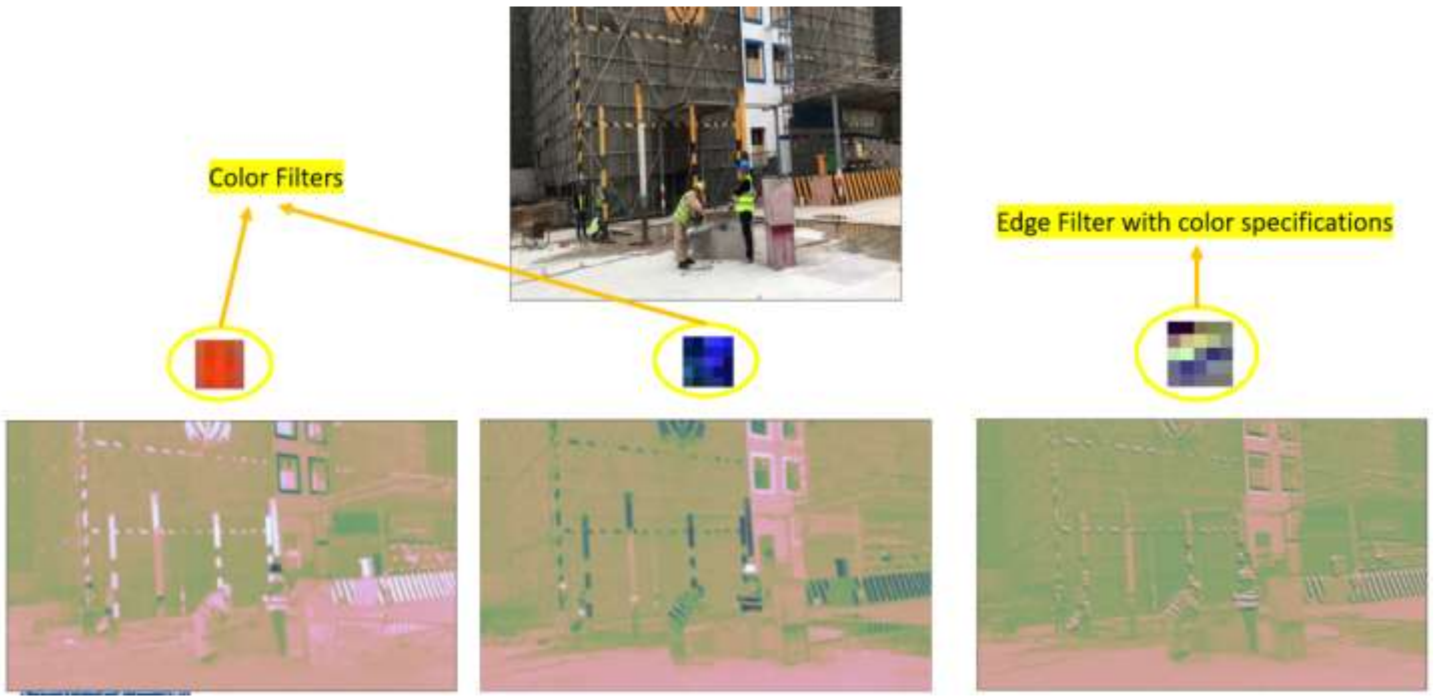

Figure 1: The image on the top represents the original image. The small boxes in between are the kernels selected for visualization. The 3 images below are the visualization of activation maps from the respective kernels.

From the figure, we can clearly see how the initial layer has been trained so as to learn specific edges, color based information and activate those specific instances. The leftmost kernel represents an orangish filter. The purpose of these filters is to highlight the features in the image which resonate with the kernels characteristics. In the leftmost case, from the activation map, we can see that all orangish, yellow features have been activated. Hence showing how the feature vector slowly builds to a complex array of numbers.

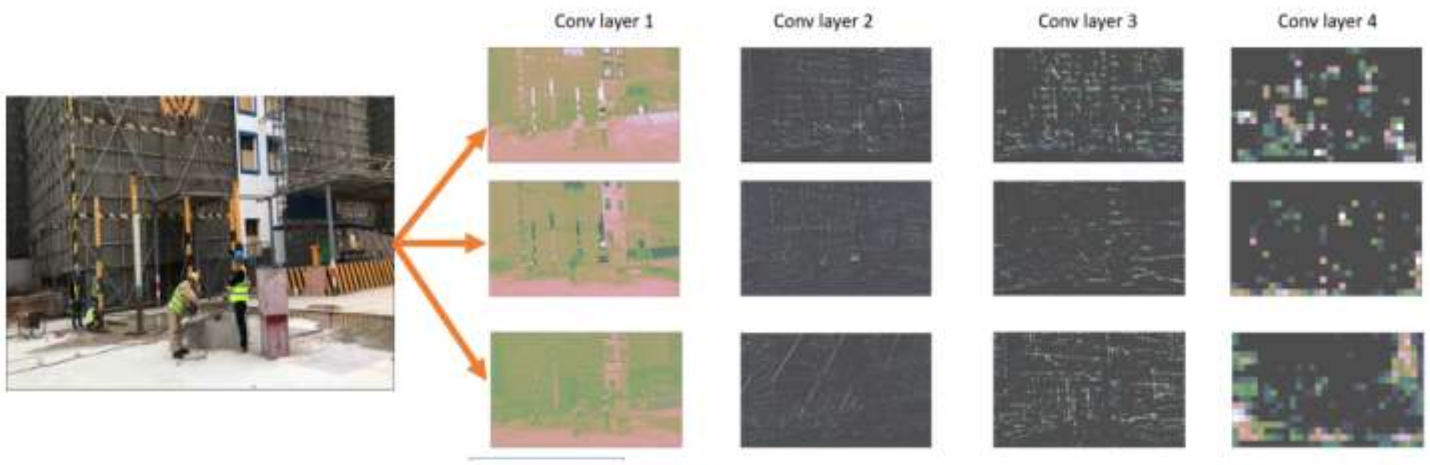

Figure 2: The above image describes the conversion of activation map and learning of features as the forward propagation progresses. The above layers are a part of the encoder side of the model.

In the figure , we can clearly see the conversion of the feature vector into something that is visually confusing. Visually we can make sense of the initial layers but as the network goes deep, the feature vector becomes an array of multidimensional numbers. For this reason, we track the first layer of encoder and 2 last layers of decoder.

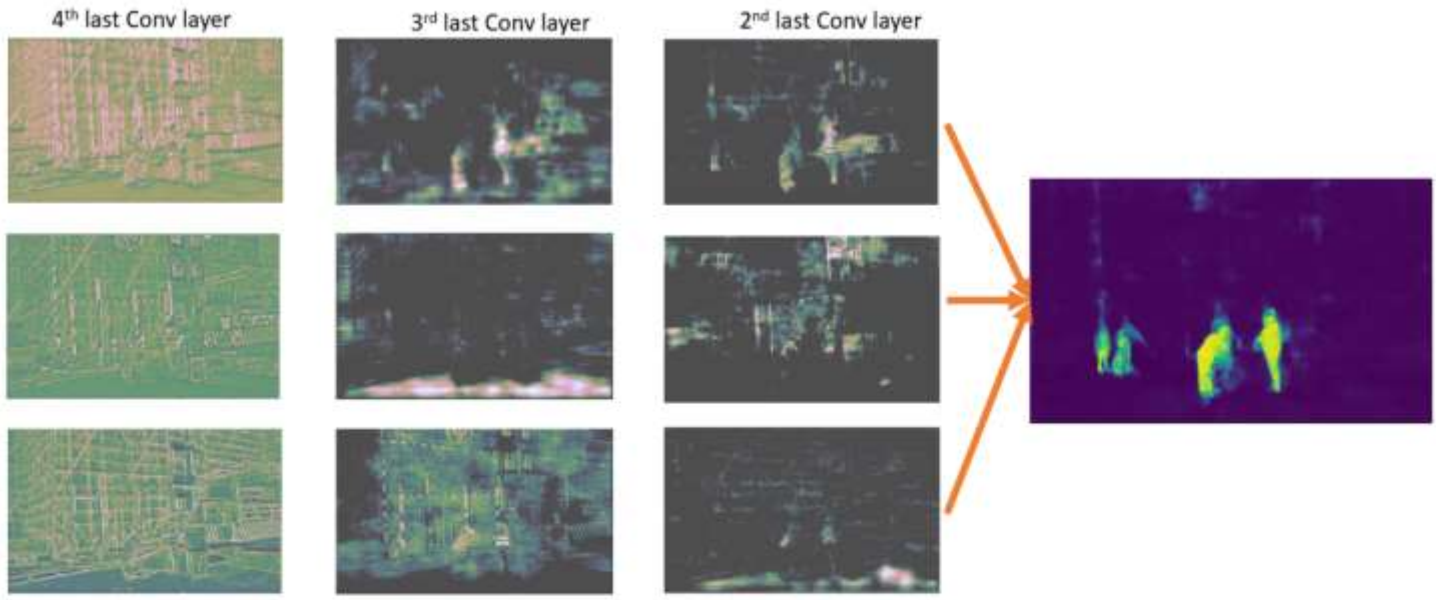

Figure 3: The above image describes the decoder side of the model. The order of forward propagation is followed from left to right . The rightmost image is the final inference after getting the logits from the last layer.

In the figure we can see how the feature vector slowly builds up the information and segmenting the pixels to their respective classes. Skip connection also plays an important role since it is responsible to pass on the initial important information.