Semantic Segmentation with DenseASPP

Ivan Tepliashin
ivan.tepliashin@studenti.unipd.it

Nikita Paltov
nikita.paltov@studenti.unipd.it

Ilia Smirnov
ilia.smirnov.1@studenti.unipd.it

Abstract

Semantic segmentation is a basic task in many spheres, where each pixel in a high resolution image is categorized into a set of semantic labels. The high-level feature representation in this task is sensitive to scale changes, thus multi-scale informaation is required to be captured by the model. To solve this problem, DenseASPP was introduced in [9]. It makes use of atrous convolutions to concatenate multiple atrous-convolved features using different dilation rates into a final feature representation. In this architecture, atrous convolutional layers are connected in a dense way, allowing to cover the scale range densely, without significantly increasing the model size. In this work, we introduce our implementation of a DenseASPP model, carry out experiments to better understand its advantages and study its performance on the Cityscapes dataset.

1. Introduction

Semantic segmentation is a fundamental problem in the field of computer vision, aiming to assign a meaningful class label to every pixel in an image. This task is critical for enabling machines to interpret and understand visual data with human-level precision. Semantic segmentation provides a pixel-wise understanding of the scene, making it indispensable in applications requiring spatially detailed insights.

The significance of semantic segmentation spans a wide range of domains. It can be used in autonomous driving, medical imaging, robotics, augmented reality, and other fields where precise contextual information can be extracted from visual inputs.

In this project, we utilized the DenseASPP (Dense Atrous Spatial Pyramid Pooling) architecture [9], a deep learning model designed to tackle the challenges of semantic segmentation. DenseASPP enhances the classic approach of deep convolutional neural networks (CNNs) by employing dense connectivity and multi-scale feature extraction using atrous (dilated) convolutions. To extract high level information, convolutional networks use multiple pooling layers to increase the receptive field size of an output neuron. However, increased number of pooling layers leads to reduced feature map size, which poses serious challenges to upsample the segmentation output back to full resolution. On the other hand, the last layers of such a network are absolutely necessary to capture higher level semantics, and we cannot get remove them without losing the quality. Atrous convolution [3] is proposed to resolve the contradictory requirements between larger feature map resolution and larger receptive fields. An atrous kernel can be dilated in varied rates by inserting zeros into appropriate positions in the kernel mask. Atrous convolution is able to achieve a larger receptive field size in comparison to a traditional convlolution, without increasing the numbers of kernel parameters. A feature map produced by an atrous convolution can be as the same size as the input, but with each output neuron possessing a larger receptive field, and therefore encoding higher level semantics. However, all neurons in the atrous-convolved feature map share the same receptive field size, which means the process of semantic mask generation only makes use of features from a single scale. We know that multi-scale information would help resolve ambiguous cases and results in more robust classification. To this end, ASPP [2] was proposed to concatenate feature maps generated by atrous convolution with different dilation rates, so that the neurons in the output feature map contain multiple receptive field sizes, which encode multi-scale information and eventually boost performance. Still, there were limitations even with ASPP approach. Specifically, input images under the autonomous driving scenarios are of high resolution, which requires neurons to have even larger receptive field. ASPP requires a large enough dilation ratio to achieve a large enough receptive field. However, the atrous convolution becomes more and more ineffective with the increase of the dilation rate (specifically, when it becomes greater than 24). To solve this problem, Dense Atrous Spatial Pyramid Pooling (DenseASPP) [9] was introduced. Such an architecture is able to encode multi-scale information, and si-

multaneously achieves a large enough receptive field size. DenseASPP consists of a base network followed by a cascade of atrous convolution layers. It uses dense connections to feed the output of each atrous convolution layer to all unvisited atrous convolution layers ahead. By the series of atrous convolutions, neurons at later layers obtain larger and larger receptive field without suffering from the kernel degradation issue in ASPP. And by the series of feature concatenations, neurons at each intermediate feature map encode semantic information from multiple scales, and different intermediate feature maps encode multi-scale information from different scale ranges. Therefore, the final output feature map in DenseASPP not only covers semantic information in a large scale range, but also covers that range in a very dense manner.

In this project we implemented the DenseASPP neural network following the architecture from [9], trained it on the Cityscapes dataset [4] and performed some experiments trying to reach better results. With our best model we managed to achieve performance on the validation part of the dataset with a mean Intersection-over-Union (mIoU) score of 54.0%.

2. Related Work

Semantic segmentation has been extensively studied in computer vision, particularly for street scenes, where accurate segmentation is critical for applications such as autonomous driving. Numerous architectures have been proposed to tackle the inherent challenges of this task.

The key work that this project is based upon is "DenseASPP for Semantic Segmentation in Street Scenes" by Yang et al [9]. In this paper, the authors propose the DenseASPP architecture, which integrates dense connectivity into the Atrous Spatial Pyramid Pooling (ASPP) framework. By stacking multiple ASPP layers with varying dilation rates in a densely connected manner, DenseASPP enables effective multi-scale feature extraction while maintaining high spatial resolution. The paper demonstrates that DenseASPP outperforms many prior methods, especially in handling small and distant objects, by leveraging dense connections to reuse features and improve gradient flow during training. This architecture showed state-of-the-art performance on benchmarks like Cityscapes, making it highly relevant for urban street scene segmentation.

In comparison to traditional approaches, such as Fully Convolutional Networks (FCNs), which laid the groundwork for pixel-level segmentation, DenseASPP provides a more refined multi-scale representation. DenseASPP maintains high-resolution feature maps, mitigating the problem of losing finer features due to downsampling.

The paper of DenseASPP makes use of ASPP, introduced in [2]. This approach uses concatenated features from multiple atrous convolution layers with different dilation rates arranged in parallel. The proposed DenseASPP combines the advantages of using atrous convolution layers in parallel and in cascade, and generates features of more scales in a larger range. Furthermore, DenseASPP improves ASPP by densely connecting multiple atrous convolution layers with different dilation rates. This allows for a more comprehensive multi-scale feature representation without excessive computational overhead. Dense connections improve feature reuse and reduce the risk of gradient vanishing during training. DenseASPP is named after DenseNet [7] which can be viewed as a special case of DenseASPP by setting dilation rate as 1. Thus, DenseASPP also shares the advantages of DenseNet including alleviating gradient-vanishing problem and substantially fewer parameters.

We follow [9] to implement our model. Unlike the original DenseASPP implementation, we used a pre-trained DenseNet-121 as our feature extractor instead of training a new backbone from scratch, due to computational constraints.

3. Dataset

In our project we are using the Cityscapes dataset [4]. It is a large-scale benchmark dataset widely used for semantic urban scene understanding tasks, including semantic segmentation, instance segmentation, and depth estimation. It contains 5,000 finely annotated images with pixel-level labels and an additional 20,000 coarsely annotated images. The images are captured from 50 cities across Germany, Switzerland, and Austria, representing diverse urban environments, seasons, and weather conditions. Each image has a resolution of 2048x1024 pixels, providing high-quality data for fine-grained analysis. The dataset is divided into training set (2975 images) and validation set (500 images). Figure 1 from [9] depicts an example of images and their ground truth segmentations.

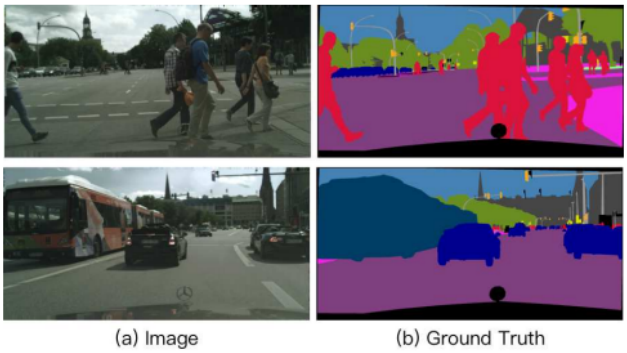

Figure 1. Example from Cityscapes dataset. In this illustration we can see scale variations of different objects caused by the distance to the camera.

The dataset images and annotations undergo preprocess-

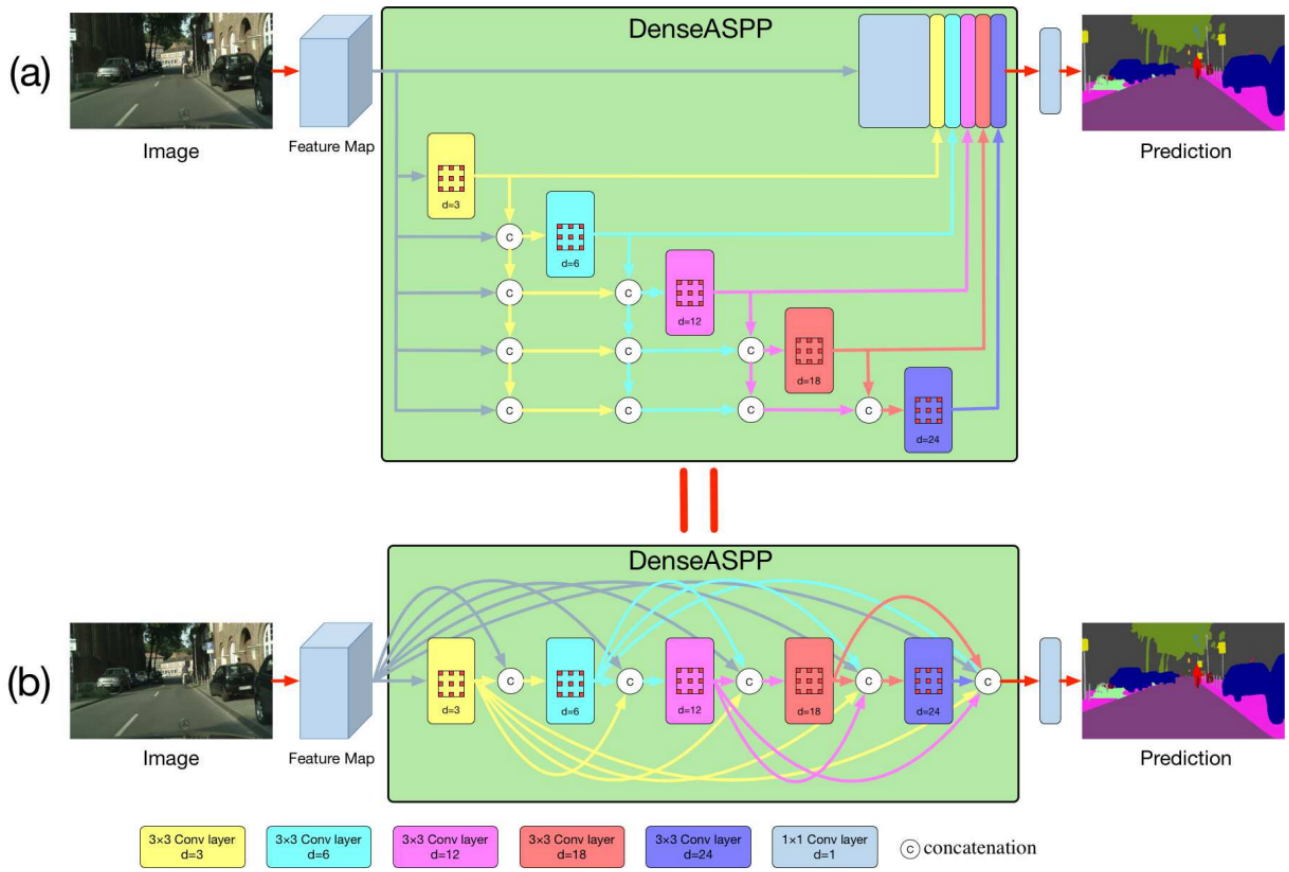

Figure 2. The structure of DenseASPP. (a) illustrates DenseASPP in detail, the output of each dilated convolutional layer is concatenated with input feature map, and then fed into the next dilated layer. Each path of DenseASPP composes a feature representation of correspond scale. (b) illustrates this structure in a more concrete version

ing transformations before being used for training and validation. The transformations applied include resizing and normalization. The original images are of size 1024×2048 (here and further in the report we describe a resolution of an image as $\text{height} \times \text{width}$). To maintain aspect ratio while reducing computational demands, we resized them to 512×1024. We also experimented with 224×224 and 512×512, but these led to lower segmentation accuracy. Images were normalized using ImageNet [5] mean and standard deviation values to align with the pre-trained DenseNet-121 feature extractor. Cityscapes provides labels for 35 classes, but it is more the high-level view of the dataset and reflects the number of manually labeled classes. A common practice for this dataset is to use only 20 main classes (including "background" class), so we map original 35 classes to 20. This approach ensures that the dataset is prepared consistently and efficiently for the semantic segmentation task, enabling effective model training and evaluation. In the training process we load the preprocessed dataset in batches of size 8, with shuffling enabled for the training set and disabled for the validation set.

4. Method

4.1. Dense Atrous Spatial Pyramid Pooling

First, we would like to describe the DenseASPP method itself. This description is mostly taken from [9]. Atrous convolution is first introduced in [1]. It allows to increase receptive field while keeping the feature map resolution unchanged. Atrous convolution is equivalent to convolving the input with up-sampled filters produced by inserting zeros between two consecutive filter values. Thus, a large dilation rate means a large receptive field. In street scene segmentation, objects usually have very different sizes. To handle this case, the feature maps must be able to cover different scales of receptive fields.

The structure of DenseASPP is illustrated in Figure 2, taken from [9]. The atrous convolutional layers are organized in a cascade fashion, where the dilation rate of each layer increases layer by layer. Layers with small dilation rates are put in the lower part, while layers with large dilation rates are put in the upper part. The output of each atrous layer is concatenated with the input feature map and all the

outputs from lower layers, and the concatenated feature map is fed into the following layer. The final output of DenseASPP is a feature map generated by multi-rate, multi-scale atrous convolutions. Thus, the obtained output is indeed a sampling of the input with different scales of receptive fields.

DenseASPP is an effective architecture to sample inputs at different scales. A key design of DenseASPP is using dense connections to enable diverse ensembling of layers with different dilation rates. Each ensemble is equivalent to a kernel in different scale, i.e. different receptive field. Dilation is able to increase receptive field of a convolution kernel. For an atrous convolutional layer with dilation rate d and kernel size K, the equivalent receptive field size is:

$$R = (d-1) \times (K-1) + K$$

Stacking two convolutional layers together can give us a larger receptive field. Suppose we have two convolution layers with the filter size K_1 and K_2 respectively, the new receptive field is:

$$K = K_1 + K_2 - 1$$

Another benefit brought by DenseASPP is the larger receptive field. Atrous convolutional layers in DenseASPP share information through skip connections. Let R_{max} denote the largest receptive filed of a feature pyramid, and function $R_{K,d}$ means that of a convolutional layer with kernel size K and dilation rate d. Thus, the largest receptive field of of DenseASPP(3, 6, 12, 18, 24) is:

$$R_{max} = R_{3,3} + R_{3,6} + R_{3,12} + R_{3,18} + R_{3,24} - 4 = 127$$

Such a large receptive field can provide global information for large objects in high resolution images.

4.2. Model Architecture

In this part we describe the architecture of our best model. Our model architecture consists of three components: the feature extractor, the DenseASPP module, and the classifier.

As a feature extractor we use DenseNet-121, pre-trained on ImageNet. Standard pretrained DenseNet-121 architecture downsamples the input 32 times, which is unsuitable for our purposes, because small features are harder to recognize in this case. To resolve this problem, following the article on DenseASPP, we remove the last two pooling layers and the last classification layer, and set the dilation rates of the convolution layers after the two removed pooling layers to be 2 and 4 respectively to be able to reuse the pre-trained weights. Thus, the feature extractor outputs a feature map with resolution of 1/8 of the input image.

The main part of our network is a DenseASPP module. It consists of a cascade of atrous convolutions with dilation rates of [3, 6, 12, 18, 24], as these provide an optimal balance of feature diversity and receptive field size. Each atrous convolution block consists of a 1×1 convolutional layer for dimension reduction, followed by the main atrous convolutional layer. A 1×1 convolutional layer before every dilated layer reduces feature map's depth into $1/4$ of its original size. The output of each layer is concatenated with the outputs of all preceding layers, enhancing the model's ability to capture multi-scale context information. Batch normalization is used in these layers to stabilize and accelerate training.

Suppose every atrous layer outputs n feature maps, DenseASPP have c_0 feature maps as input, and the l-th 1×1 convolutional layer before l-th dilated layer have c_l input feature maps. Then:

$$c_l = c_0 + n \times (l-1)$$

According to the paper, every 1×1 convolutional layer before the dilated layer is supposed to reduce the dimension into $c_0/2$ channels. The authors set $n = c_0/8$ for all atrous layers in DenseASPP. However, in our model, we set $n = c_0/16$ (we call n growth rate), due to the limitations of our computational powers. The feature map of DenseNet-121 has 1024 channels. Before every atrous layer, the number of channels in a feature map is reduced into 256, which is $c_0/4$ in this case. This reduction allows us to control model size and prevent the network from growing too wide.

Next, we have a classifier head: the final output feature map is passed through a 1×1 convolutional layer with 20 filters, generating a low-resolution segmentation map. This is then upsampled by a factor of 8 to match the original input size.

Overall, our model has 2252820 trainable parameters.

4.3. Training Parameters

The network was trained on P100 (on Kaggle resources) for 60 epochs using the Adam optimizer, with the initial learning rate of 0.0003 and weight decay of 0.00001. These optimizer parameters are taken from the original paper.

We experimented with different growth rates (amount of feature maps output by a single atrous layer) and found that a growth rate of 64 performed best, while a lower growth rate of 32 led to reduced accuracy.

5. Experiments

To begin with, our network architecture has undergone changes during our attempts to find the best model configuration. Different models, that we trained along the way, are reported in Table 1. Our base feature extractor was DenseNet-121. In out first models we used a feature extractor with no modifications and input images of resolution 224×224 in the training set. DenseNet outputs a feature map of 1/32 of original image size with 1024 channels.

Model	DenseASPP	Max scale	Growth rate	# parameters	image size	# epochs	mIoU train	mIoU val
#1	(3, 6, 9, 12)	61	32	662420	224×224	50	51.4%	31.6%
#2	(3, 6, 9, 12)	61	32	1266580	224×224	60	54.9%	29.9%
#3	(6, 12, 18, 24)	121	32	1266580	512×512	80	63.3%	41.4%
#4	(3, 6, 12, 18, 24)	127	64	2252820	512×1024	60	**73.5**%	**54.0**%
#5	(3, 6, 9, 12, 18, 24)	145	64	2748179	512×1024	60	55.0%	39.5%

Table 1. Comparison of our models trained in the process of pursuing the highest performance.

Since our classifier upsamples its input only by a factor of 8, and we are still required to get back to the original image size in the output, in our early models we had to use an extra upsampling layer by a factor of 4 after the feature extractor. Due to this the network could not learn fine details. To mitigate this problem, we tried removing a couple of last blocks in the feature extractor to avoid extra downsampling. In model #1 from 1 we use DenseNet-121 without two last dense blocks. This allows us to avoid extra upsampling, but this also did not help us reach higher quality, because a significant number of features was learned in the dense blocks that we removed. Next, we tried to leave one more dense block and thus also introduce upsampling by factor of 2 after the feature extractor. This combined approach was used in the model #2 from 1. As we can see from the resulting mIoU scores, it did not result in an improvement. In the next step, we decided to increase the resolution of images in the training set to 512×512 and simultaneously move from basic DenseASPP configuration of [3, 6, 9, 12] to another configuration with larger receptive field, [6, 12, 18, 24]. This model (#3) reached 41.4% on validation set, but it was still underperforming. Next, in the model #4 we added 1×1 convolution before each atrous layer, moved to a higher growth rate (size of an output of a single atrous layer) of 64 instead of previously used 32 and added another atrous layer with the dilation of 3. Thus, we tried to implement a model carefully following the one reported in the original paper on DenseASPP. Also, we decided to increase the resolution of input images up to 512×1024 to preserve original scale. This approach allowed us to achieve the best performance of 54.0% on validation set. We tried to improve the model further, adding more atrous layers, but this did not result in any improvement. The receptive field provided with such a configuration is too big, which cannot enhance the performance of a model, but only makes it more complex. Also we tried to exclude 'background' class from training and train our model only on 19 classes instead of 20, but this resulted in the fall of mIoU in the model #5.

After we have reached the best model, we conducted various experiments in attempts to improve the model's performance and better understand the significance of different modules.

Feature extractor	# epochs	mIoU train	mIoU val
Densenet-121	60	73.5%	54.0%
Densenet-201	60	**73.9**%	**54.2**%
MobileNet V3	40	67.7%	41.9%
ConvNeXt	30	65.9%	53.0%

Table 2. Feature extractor comparison, performed on the best model.

5.1. Feature Extractor Variation

After finding the best model configuration, we experimented with different backbones. The results of these experiments are reported in the Table 2. First, we tried DenseNet-201 [7]. This variant of a DenseNet has more layers than a DenseNet-121 and therefore it is a more complex model It is supposed to achieve higher accuracy in classification, but this configuration is heavier. It did improve the mIoU of our best model, but insignificantly. Next, we also tried MobileNetV3 [6] and ConvNeXt [8]. These two models have modern CNN architectures, which were introduced later than the DenseNet, in 2019 and 2022 correspondently. MobileNet V3 is a lightweight CNN optimized for mobile and edge devices. It combines depthwise separable convolutions, squeeze-and-excitation (SE) blocks, and the h-swish activation function to improve efficiency. ConvNeXt is another modern CNN architecture inspired by Transformer-based models. It reintroduces pure CNNs with ResNet-like architecture but incorporates design principles from Vision Transformers. These models are both heavier than the DenseNet, so training a network which uses them as a backbone requires more time. This is the reason why we could not train them for the same amount of epochs as we did with DenseNets. The networks featuring these feature extractors still could not outperform DenseNet-121 within our computational constraints. However, we believe that with more computational resources they are capable of outperforming DenseNets in this task. Both ConvNeXt and MobileNetV3 output a feature map of 1/32 of input size, thus we require additional upsampling before DenseASPP module. This problem may be potentially mitigated with some improvements to the architecture of these models that would allow to reuse pre-trained weights without aggresive downsampling.

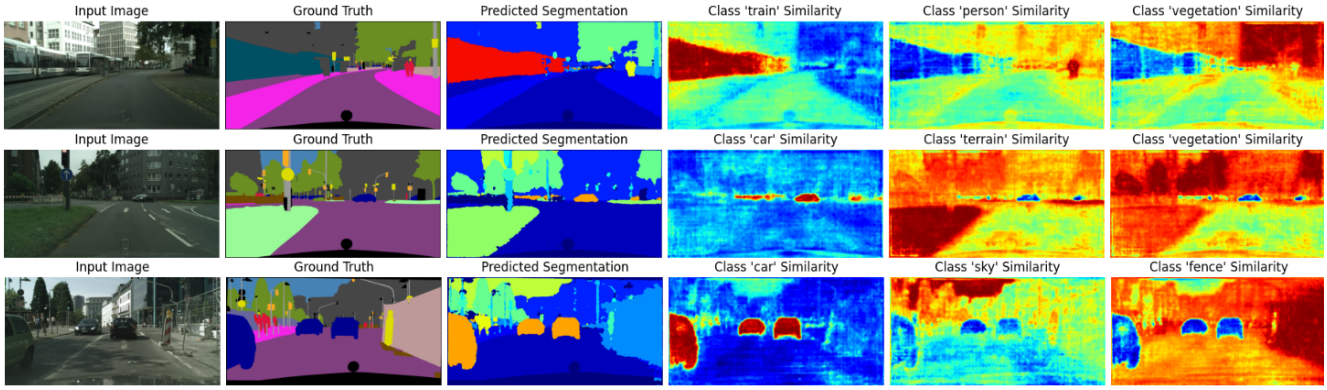

Figure 3. Illustration of the results of our model and feature similarities between a fixed class and all the pixels in an image. Hotter color means higher cosine similarity in the feature space.

5.2. Ablation Studies

All the ablation studies we performed on our best model, e.g. with DenseASPP(3, 6, 12, 18, 24) and the architecture following the original article [9].

First, we performed feature similarity analysis. Context information is of great significance for distinguishing confusing categories and classifying large objects. Some features from the Cityscapes dataset are harder to distinguish due to their similarity (such as 'bicycle' and 'motorcycle', 'bus' and 'car' or 'truck' and 'train'), making certain classes more challenging to segment. For these categories, sufficient context information is critical. Fig. 3 illustrates three examples of results of our best model along with feature similarities of some better recognized classes. Cosine similarity is used to measure the similarities between features of pixels and a fixed class. The heat-maps depict how much each pixel in the feature map is similar to the chosen class. They show that most of the pixels in a continuous region with the same label have similar features. DenseASPP uses large context information, which theoretically allows it to better classify pixels in comparison to other approaches.

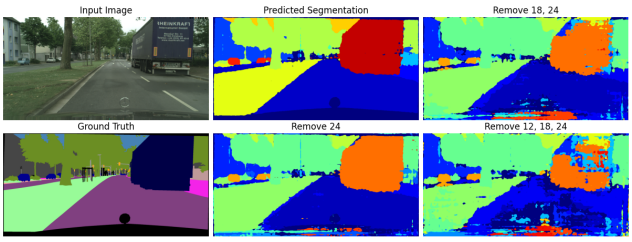

Figure 4. Effects of removing top levels in the DenseASPP module.

Furthermore, we tried removing some of the upper DenseASPP blocks to study how this would influence the results. Larger objects were still recognizable when removing 1–2 DenseASPP blocks, but further they suffered significantly. An example is portrayed in Fig. 4. In this image we can see a large truck, which is still classified correctly, when first two upper DenseASPP blocks are removed, but with further removal model becomes incapable of properly segmenting it. Smaller objects, such as cars in that image, are still recognized more or less correctly even with the 3 top blocks removed.

5.3. Additional Remarks

Overall, we noticed that there is some class-specific performance. The size of an object in an image plays a significant role in the ability of a model to recognize it. Objects like cars were segmented with high accuracy, whereas small objects like traffic signs were often misclassified due to their small size.

Also, during our pursue of the best model, we found out that larger input images resulted in better segmentation. We believe this is due to higher preservation of details before downsampling. Original images from the Cityscapes dataset have resolution of 1024×2048, we resize them to 512×1024, and the feature extractor downsamples them by factor of 8. This can result in losing of a substantial part of information related to a small object, thus making it harder for the model to learn it.

6. Conclusion

In this project we implemented a DenseASPP network. Our best model configuration achieved an mIoU of 54.0% on the Cityscapes validation set. Given our limited resources, this performance is reasonable but could be improved with some improvements and modifications.

First of all, a larger dataset could ensure that the model better learns all the features. It could be enriched using standard augmentation techniques, such as random crops, flips and scaling. Also, using random crops instead of resizing could help preserve more spatial information. Next, the training could be extended to 80 epochs, as done in the original paper. This would likely improve results. Further-

more, higher resolution inputs could be used. Training with original-sized images (1024×2048) instead of resized images could capture finer details. Different backbones could be used to improve the feature extraction. Our experiments show that ConvNeXt is potentially capable of outperforming DenseNets with proper modifications and longer training.

Despite the constraints, our results demonstrate the effectiveness of atrous convolutions and DenseASPP in semantic segmentation. With sufficient computational resources, DenseASPP remains a powerful method for handling multi-scale variations in segmentation tasks.

References

[1] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs, 2016.

[2] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs, 2017.

[3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2018.

[4] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[6] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3, 2019.

[7] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2018.

[8] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s, 2022.

[9] Maoke Yang, Kun Yu, Chi Zhang, Zhiwei Li, and Kuiyuan Yang. Denseaspp for semantic segmentation in street scenes. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3684–3692, 2018.