Comparing Transfer Learning approaches for Neural Networks in Semantic Segmentation

Auriane Mahfouz, Inês Jesus, Camilo Betancourt Nieto
Student IDs: 2072042, 2073570, 2087189
{auriane.mahfouz, ines.caeironevessilveirajesus, camilo.betancourtnieto}@studenti.unipd.it

Abstract

Semantic segmentation is a key technique in computer vision that enables the identification of specific objects within images and is a crucial task across various applications, from medical imaging to autonomous driving. Our focus is on comparing the performance of several deep learning models on the Oxford IIIT Pet Dataset. Apart from fine-tuning the FCN-ResNet50 and the DeepLabV3-ResNet50, we implemented some instances of the DenseASPP and a simple architectural variation of it. Having these set of network designs and varying the datasets on which the models were trained allowed us to evaluate which transfer learning approach works best for our specific task. Our best model, an instance of DenseASPP, obtained a pixel accuracy of 92% and a weighted mIoU of 85% on the test set. Additionally, we performed a qualitative assessment of the models' predictions, taking into account their key characteristics, the way they were trained, and the challenges they faced.

1. Introduction

Semantic segmentation is a crucial process in computer vision that involves breaking down images into segments or groups of pixels to identify distinct objects or areas. This technique is fundamental across a wide array of applications. In medical imaging, it enables doctors to pinpoint and analyze specific regions for more accurate diagnoses. In the automotive industry, image segmentation is essential for developing autonomous vehicles that can navigate safely by recognizing various elements like roads, pedestrians, and obstacles. Also, it enhances augmented and virtual reality experiences by seamlessly integrating digital elements with the real world. Moreover, it's a game-changer in digital editing, offering sophisticated tools for tasks such as object removal and background changes. Pet image segmentation can have applications in smart home systems, allowing to track pets to ensure their safety or monitoring their activities. More generally, animal image segmentation is also used in CAPTCHA (a test that allows systems to tell humans apart from computers).

This task faces several difficulties, including the wide variety of object appearances, the complexity of backgrounds, and the need to adapt to different scales within images. Another significant challenge is the need for real-time processing in applications like autonomous driving, which demands both speed and accuracy. However, there have been previous efforts in the literature that tackle these problems effectively. This report builds upon previously defined models for semantic segmentation (more on this in Sec. 2): Fully Convolutional Networks (FCNs) [8], DeepLabV3 [1], and DenseASPP [12], applied to the Oxford-IIIT Pet Datase [9].

We aimed to compare how different architectures adapt when using transfer learning on a different dataset. Specifically, we fine-tuned various pretrained versions of DenseASPP, including an architectural modification we made and labeled as CutDenseASPP, and compared their performance against FCN and DeepLabV3 models on the Oxford-IIIT Pet Dataset. We evaluated the models based on pixel accuracy, weighted mean Intersection over Union (mIoU), and prediction time. According to these metrics, DenseASPP demonstrated the best overall performance. Its variant, CutDenseASPP, also outperformed the other models and highlighted the flexibility of its parent architecture.

2. Related Work

The paper by Long et al. [8] introduced Fully Convolutional Networks (FCNs) for semantic segmentation, demonstrating how CNNs can be adapted to output spatial maps for pixel-wise classification. This adaptation allows preexisting models, originally used for image classification, to be trained end-to-end for semantic segmentation, effectively leveraging the features learned from their initial tasks.

Additionally. the DeepLab series introduced by Chen et al. [1] [2] made significant advancements in using atrous (dilated) convolutions for semantic segmentation, improving the ability to capture multi-scale information and enhancing segmentation accuracy. These atrous convolutions allow to expand the receptive field of the filters without increasing the number of parameters or the amount of computation required.

Expanding on this, Yang et al. [12] introduced the Dense Atrous Spatial Pyramid Pooling (DenseASPP), an architecture that uses atrous convolution layers in parallel and in cascade, integrating the advantages that both of these approaches offer. In cascade mode, each upper atrous layer receives the output from a lower atrous layer, which allows the network to efficiently produce progressively larger receptive fields. On the other hand, in parallel mode, multiple atrous layers process the same input simultaneously and their results are concatenated, resulting in an output that samples the input at various scales of receptive fields.

In the aforementioned work, the authors implemented the DenseASPP architecture as a head on top of a pretrained backbone, specifically DenseNet [6], which shares similarities with the DenseASPP core idea but was originally designed for classification and does not incorporate atrous convolutions. This implementation was tested on the Cityscapes dataset [3].

Extending these approaches, we fine-tuned these models on a simple but different dataset. For some models, we kept the architecture unchanged, while for others, we made slight modifications to the architecture and examined how these changes affected the results.

3. Dataset

For this project, we used the Oxford-IIIT Pet Dataset [9] which is a widely recognized dataset used for image classification, object detection, and semantic segmentation.It consists of 7,349 images spanning 37 different breeds of cats and dogs, with a roughly equal distribution between the two species. Each breed is represented by approximately 200 images, providing a balanced dataset.

It is also worth mentioning the three datasets employed for pretraining the models we used:

- **ImageNet [4]**: A large-scale dataset containing over 14 million images across 1,000 object categories, used extensively in computer vision.
- **Pascal VOC [5]**: A dataset with around 11,000 images and 20 object classes including people, animals (birds, cats, cows, dogs, etc.), vehicles, and indoor elements. It includes annotations for object detection, classification, and segmentation tasks.
- **ADE20K [13]**: A diverse dataset with over 20,000 images annotated with 150 object categories, covering objects, nature, walls, floor, sky, vehicles, animals (dogs, cats, horses, etc.).
- **COCO [7]**: A comprehensive dataset containing over 330,000 images with annotations for object detection, segmentation, and captioning, covering 80 object categories including everyday objects, animals (dogs, cats, horses, sheep, etc.), and people.

3.1. Data Pre-Processing

The data was downloaded directly from the Torchvision library, which splits the data into training and testing according to the strategy employed in the original paper [9]. In this paper, the split was designed to create a perfect balance between breeds, which is not so important in this case since the purpose of the model is to segment any pet, independently from the original category. Our split still considers the original split structure but to accommodate for more training instances the validation is split from the test set, which is originally set to be 50% of the total data. As such, the proportions for the training, validation and test sets are 50-25-25, respectively.

The data is resized to a size of (256,256) to have uniform samples to feed into the models and then normalized by channel according to the mean and standard deviation of the training set.

3.2. Data Augmentation

As a way to make the model more robust, we applied the following data augmentation techniques to every sample of the training data:

1. `RandomRotation` with a range from -90 to 90 degrees,
2. `RandomHorizontalFlip` with a probability of 0.5,
3. `ColorJitter` with brigthness, contrast and saturation from 0 to 2 and hue of 0.1, and
4. `GaussianBlur` with kernel size of 3 and sigma between 0.1 and 2.0.

These data augmentation techniques doubled the size of the training dataset with the purpose of helping the model generalize better to different lighting and image quality conditions and animal positions.

4. Method

We approached this task in two different ways.

First, we used pretrained models for semantic segmentation, imported directly from the PyTorch model zoo [10], and adapted them for our specific dataset to establish a common benchmark. This approach was used for the FCN and the DeepLabV3.

The backbone used in both these models was a ResNet50 with weights pretrained on the COCO dataset [7]. The

ResNet50 is a deep convolutional network with 50 layers that include shortcut connections that help to mitigate the vanishing gradient problem, enabling to train very deep networks. Here it acts as a feature extractor.

Then, by modifying a public implementation of the DenseASPP [11], we had more opportunities to customize the model in terms of its pretraining options and its architecture, tailoring it more precisely to our specific task.

We now briefly describe the main features of each model, giving some insights about their implementation.

4.1. FCN

The first of the considered models is the Fully Convolutional Network (FCN) with a ResNet50 backbone (FCN-ResNet50).

FCNs are designed for dense prediction tasks like semantic segmentation since they replace the traditional fully connected layers used for classification with convolutional layers in their architecture, allowing the network to output a spatial map of class predictions. The feature maps outputted by the ResNet50 are upsampled using transposed convolutions, the opposite of convolution, thus being able to retrace the output back to the the size of the input sample.

4.2. DeepLabV3

We also trained the DeepLabV3 model with a ResNet50 backbone (DeepLabV3-ResNet50). DeepLabV3 distinguishes itself with its adept use of Atrous Convolution and the Atrous Spatial Pyramid Pooling (ASPP) module, designed to capture contextual information at multiple scales effectively. This approach is particularly beneficial for segmenting objects of varying sizes within the same image, addressing one of the common challenges in semantic segmentation.

4.3. DenseASPP

This model connects a set of atrous convolutions in a dense way, meaning that each layer receives inputs from all preceding layers and passes its own outputs to all subsequent layers. Apart from helping the gradient flow [6], this approach enables the network to effectively capture multi-scale features by leveraging the strengths of both cascading and parallel strategies when implementing atrous convolutions. This aspect is important because, as noted by the authors, the aim is that the resulting multi-scale features not only span a wide scale range but also cover it densely, which is key for effective image segmentation applications. More specifically, this design uses four atrous convolutional layers with dilation rates of 3, 6, 12, 18 and 24.

The DenseASPP is built on top of a backbone model, functioning as a head specifically suited for image segmentation. In our project, we aimed to assess the impact of network pretraining on results. We tested two versions of DenseASPP: `DenseASPP1`, with its head pretrained on the augmented Pascal VOC Dataset [5], and `DenseASPP2`, with its head pretrained on the ADE20K Dataset [13]. Both versions used the dilated DenseNet121 as the backbone. DenseNet121 is a variation of DenseNet that incorporates atrous convolutions with progressively wider dilation rates, similar to the DenseASPP head. In both cases, the backbone was pretrained on ImageNet.

4.3.1 CutDenseASPP

To evaluate the flexibility of the DenseASPP architecture when modified, and to assess the impact of both the backbone and the head on model performance, we truncated the DenseASPP design. Specifically, we altered the architecture by retaining only the layers with dilation rates of 3, 6, and 12, and adjusted the classification layers accordingly. To ensure a fair comparison and compensate for the reduced number of parameters, we replaced the original backbone with a larger network, the dilated DenseNet161 (which was also pretrained on ImageNet). This adjustment enabled us to compare the performance between a model with a smaller backbone and a larger head (the standard DenseASPP) versus a model with a larger backbone and a smaller head (the CutDenseASPP). We tested two versions of the CutDenseASPP: `CutDenseASPP1`, with its head pretrained on the augmented Pascal VOC Dataset, and `CutDenseASPP2`, with its head pretrained on the ADE20K Dataset.

For all DenseASPP networks (including the CutDenseASPP), the optimizer implementation follows the same procedure as in the original paper.

Model	Backbone	Head pretraining dataset	Trainable par.
FCN	ResNet50	COCO	9,439,747
DeepLabV3	ResNet50	COCO	16,126,211
DenseASPP1	Dilated DenseNet121	Pascal VOC	2,220,675
DenseASPP2	Dilated DenseNet121	ADE20K	2,220,675
CutDenseASPP1	Dilated DenseNet161	Pascal VOC	2,197,347
CutDenseASPP2	Dilated DenseNet161	ADE20K	2,197,347

Table 1: Model details and number of trainable parameters

5. Experiments

Given our task and dataset, our models were trained to classify between the background, the foreground, and the

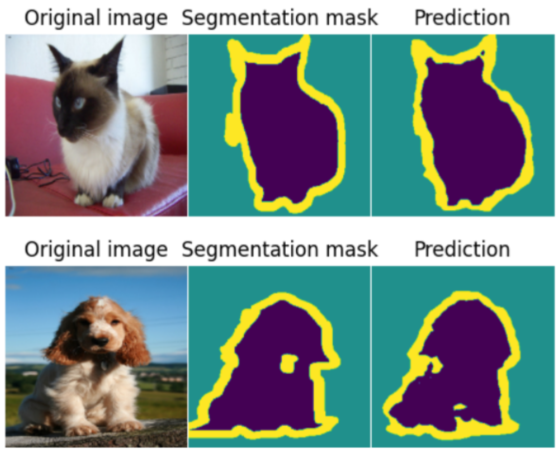


Figure 1: Test images, ground truth, and predictions made by DenseASPP1

boundary of pets in images containing cats and dogs. To illustrate the kind of images in the dataset, the ground truth (labeled as "segmentation mask"), and the capabilities of our models, Fig. 1 shows two examples of the predictions made by the DenseASPP1 on the test set.

For all networks, we froze the backbone weights and trained only the heads of the models. This way, the backbone served as a fixed feature extractor. Tab. 1 shows the number of trainable parameters for each model (after the freezing step), as well as the backbone network used and the dataset on which the heads were pretrained.

All models were trained for a maximum of 10 epochs, but an early stopper was incorporated to finish training if the validation loss was not improving. As mentioned before, after freezing the backbone, we trained the last part of the models (the head) and computed the pixel accuracy and the weighted mIoU, which are shown in Tab. 2.

From these metrics, we can observe that the DenseASPP1 beats the rest of the models in all the aspects that we took into account, although the difference with respect to DenseASPP2 is not substantial. This suggests that, in our specific setting, the architecture has a greater impact on performance than the dataset used during the pretraining stage. We can also observe that the CutDenseASPP variants also outperform the other two models regarding pixel accuracy and weighted mIoU. This behavior showcases the flexibility that the DenseASPP has, as it manages to achieve competitive metrics even if its design is truncated. However, this also shows that, for this particular case, having a well-suited head is more effective than having a heavy backbone but an overall model that is not as tailored for the task at hand. This becomes more evident when looking not only at the trainable parameters, but also at the total parameters (which are around 33M, 39M, 10M, and 30M for the FCN, DeepLabV3, DenseASPP, and CutDenseASPP, respectively).

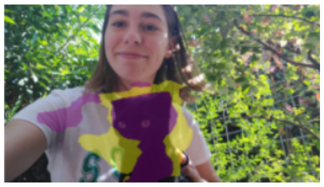
Model	Pixel accuracy	Weighted mIoU	Pred. time (s)
FCN	0.887	0.805	36.784
Deeplabv3	0.894	0.817	47.600
DenseASPP1	**0.917**	**0.854**	**27.626**
DenseASPP2	0.916	0.853	28.567
CutDenseASPP1	0.902	0.831	57.061
CutDenseASPP2	0.906	0.838	56.600

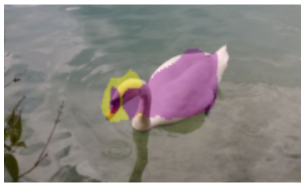
Table 2: Results on the test set

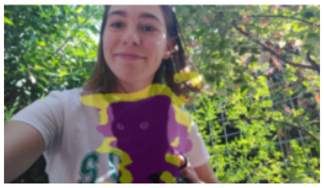
Apart from the typical objective metrics we computed, we also performed a simple demonstration by predicting examples taken from our own photos to provide a qualitative assessment of the models' performance when computing the segmentation of the pet and its corresponding boundary.

Two selected predictions of our best and worst models are shown in Fig. 2. We selected these specific images to observe how the models behave when tested against examples that slightly differ from those in the Oxford-IIIT Pet Dataset, which was used for fine-tuning. In one case, we tested an image containing a pet alongside another entity, specifically a human face. The other image features a swan, an animal not included in the dataset, which only contains dogs and cats. Our best model demonstrated more precise predictions, particularly regarding boundary accuracy, while our worst model failed to enclose the animal properly. In the swan image, even though this kind of animal was not present in the fine-tuning dataset, our models were able to identify the animal's coarse shape. This could be due to the abstraction capabilities achieved by pretraining the models on larger, more general datasets. Still, as we would expect, the prediction seems to be more accurate in the cat's example. Furthermore, both models, although not perfectly, seemed able to distinguish the cat's face from a human face, with DenseASPP1 showing a more accurate prediction.

Another interesting experiment we conducted is shown in Fig. 3. This time, we challenged the models by presenting two different perspectives of the same animals (which, again, are neither cats nor dogs) in the exact same context. In this example, it seems that our best model has more flexibility to identify the animals, despite of the differences in perspective. This result is reasonable, taking into account that this model's architecture aims to capture a wide range of scales in its features. However, there are some imperfections in the predictions that might be attributed to the fact that animal in question is not included in the fine-tuning dataset.


(a) Prediction 1: FCN


(b) Prediction 2: FCN


(c) Prediction 1: DenseASPP1

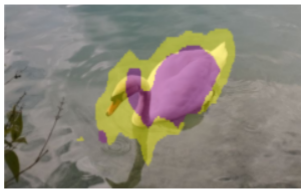
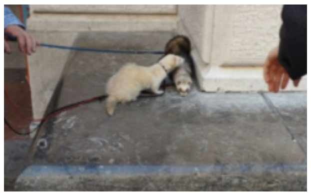
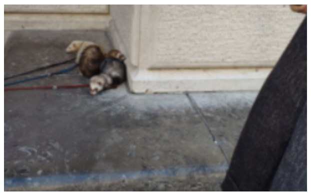
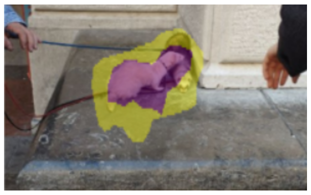
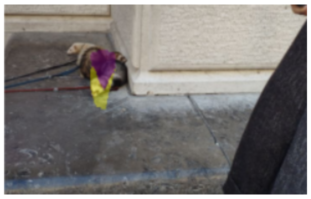

(d) Prediction 2: DenseASPP1

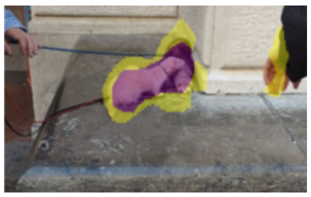
Figure 2: Selected examples of semantic segmentation for animals and their boundaries


(a) Original image - Perspective 1


(b) Original image - Perspective 2


(c) FCN - Perspective 1


(d) FCN - Perspective 2


(e) DenseASPP1 - Perspective 1

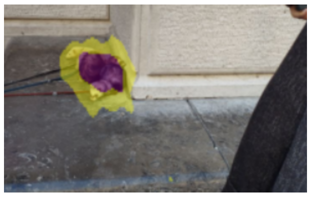

(f) DenseASPP1 - Perspective 2

Figure 3: Semantic segmentation for different perspectives of the same animals

6. Conclusion

After fine-tuning various models for the semantic segmentation task on the Oxford-IIIT Pet Dataset, we found that the DenseASPP was the best performing one in terms of pixel accuracy, weighted mIoU, and prediction time. We also proved this model's robustness, even when its architecture is slightly modified.

The set of experiments we conducted showed that, when using transfer learning techniques, neural networks are able to generalize their learned knowledge, even in cases where the test data has small difference with the fine-tuning dataset. However, having relevant data at hand and applying models that capture multi-scale features effectively are key for computer vision and, specifically, semantic segmentation.

References

[1] 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.

[2] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation, 2017.

[3] 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, 2016.

[4] 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.

[5] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

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

[7] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[8] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation, 2015.

[9] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[10] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[11] Tramac. Awesome semantic segmentation using pytorch. `https://github.com/Tramac/awesome-semantic-segmentation-pytorch/blob/master/README.md`, 2023. Accessed: 2024-07-10.

[12] 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.

[13] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.