University of Padova

Department of Information Engineering

Master's degree in Computer Engineering

Incremental Learning Techniques for Part-Based Semantic Segmentation

Supervisor

Pietro Zanuttigh

Co-supervisor

Umberto Michieli

Candidate

Luca Rossi

(ID: 1141231)

14 October 2019

Academic Year 2018/2019

I would like to express my gratitude to my supervisor Professor Pietro Zanuttigh for allowing me to work in his Lab.
I am particularly grateful for the assistance given by my advisor PhD student Umberto Michieli who supported and bore me during the development and writing of this thesis.
I would also like to thank all the LTTM lab for the good times spent together, the laughs and all the adventures faced... a broken motherboard, two faulty power supplies, a Linux Partition erased by Windows Update and other maintenance stuffs.
Finally, I must express my very profound gratitude to my girlfriend and my family for the continuous support and encouragement throughout my years of study. This accomplishment would not have been possible without them.
Thank you

Abstract

Semantic Image Segmentation is a computer vision task in which we label each single pixel in an image according to the semantic content. This task is of essential importance for a wide range of applications as: autonomous vehicles, robot perception, human-machine interaction, medical image segmentation, image editing tools, image search engines and augmented reality to name a few. However some application tasks need more accurate scene understanding, for robot perception a human body part segmentation can be a very valuable tool in order to identify parts such as hands and interact with them, in this case it is possible to talk of Part-Based Semantic Segmentation. Although many works have studied this topic and the Pascal-Part dataset has been created specifically for this scope, most of these works concern only parts of few classes as human, animals, cars.

One of the biggest problems is to find a more general model capable of learning at the same time a huge number of parts from different classes.

In this work we use an Incremental Learning approach to try to develop such model. Our framework uses two networks where the second one is fed with the output of the first.

The first network performs the Semantic Segmentation task with 21 classes from the Pascal-VOC dataset, while the second one performs the Part-Based Semantic Segmentation task with 108 classes\parts on the Pascal-Part dataset. We achieve this goal using the features extracted by a custom CNN from the softmax (or argmax) output of the first network to constraint and condition the prediction of the parts.

Additionally, we have studied a new loss, with the aim of relating different parts of the same class throught an Adjcency-matrix. Then, a distance function is applied to minimize discrepancies between the predicted matrix and the ground-truth one, this should teach to the network to penalize parts that should not be close.

Comparing our model results with a simple network for Semantic Segmentation adapted for our task, we were able to increase the Mean Intersection over Union from 29.9% to 35.1%.

Contents

List of Figures

List of Tables

1 Introduction

Semantic Image Segmentation is computer vision task which describes the process of associating each pixel of an image region with a class label according to the semantic content i.e. what is being represented. Since it makes a prediction for every pixel in the image it is commonly referred to as dense prediction. Nowadays the increasing number of applications with benefits from inferring knowledge from imagery had made this topic one of the key problems in the field of computer vision as it is crucial to have a complete scene understanding of a given image.

A wide range of application use this process as for example: road segmentation for autonomous vehicles, scene segmentation for robot perception, human-machine interaction, medical image segmentation, computational photography - image editing tools, image search engines and augmented reality to name a few. Some of these applications need a more accurate scene understanding, for robot perception a human body part segmentation can be a very valuable tool in order to identify parts, such as hands, and interact with them to perform some tasks. In this case it is possible to talk of Part-Based Semantic Segmentation.

Despite the availability of datasets with part-annotations for a large number of different classes, this topic has been addressed in the past using various traditional computer vision and machine learning techniques but for only for the human body or for some specific objects. In this work we do not consider only a few classes but we try to find out a more general method that can be used for all kind of classes belonging to a dataset, in this case the Pascal-Part dataset.

The main idea of this thesis is connected to the Incremental learning problem, which is defined as the capability of a CNNs architectures to continuously extend the existing model's knowledge without forgetting previous classes and losing performance on old data. In the same way we would not lose information from a previous network which performs Semantic Segmentation for the 21 classes from

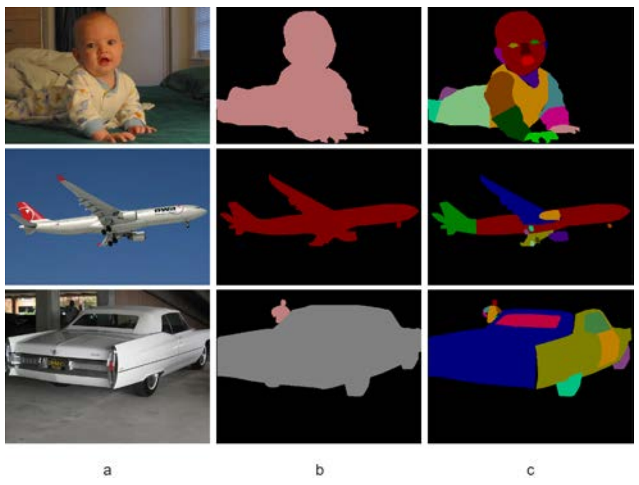


Figure 1.1: Some example from Pascal-VOC and Pascal-Part dataset. a) RGB images b) shows some ground truth from Pascal-VOC for Semantic Segmentation while c) shows some annotations from Pascal-Part for Part-Based Semantic Segmentation.

Pascal-VOC dataset. Our idea assumes that an output from this kind of network could give useful information for a second network performing Part-Based Semantic Segmentation. We evaluate this approach on Pascal-Part dataset which extends the Pascal-VOC with annotation parts for each one of the 21 classes, Figure 1.1. In this work we start implementing a State-of-the-Art CNNs as DeeplabV2 with a ResNet101 encoder. The network is trained on a refined Pascal-Part dataset to obtain a baseline results and on Pascal-VOC to generate additional data that we want use to improve the baseline. In this thesis we want to investigate new methods to extract higher level features from the output of the first network that can be useful to improves lower level feature for part-annotations in order to increase the accuracy on the Part-Based Semantic Segmentation task. The output of the first network will be concatenated with the RGB input image and fed as input to the second neural network.
Moreover we studied a new loss to associate each other parts of the same class. The network should learn during the training phase how to discriminate parts that do not belong to the same class and that should not be neighbour. In order define this loss we built an Adjacency matrix from Graph Theory, where rows and columns index are related to class-parts. So each matrix element will be different from zero only if the related parts belong to the same class or appear close on the image.

The remainder of this thesis is organized as follows. Chapter 2 introduces the problem of semantic segmentation and describes the state-of-the-art frameworks that use deep learning techniques to solve this problem. Chapter 3 describes the framework utilized for this work and the proposed techniques that have been developed. Chapter 4 defines the methods we have used to reduce and refine the Pascal-Part dataset. Chapter 5 reports the main results obtained from the experiments. Chapter 6 outlines and discusses future works.

2

Deep Learning for Image Understanding

Image segmentation is the process of partitioning an image into multiple segments that have similar properties and features. Semantic Segmentation is an extension of this technique and its challenge consists in labelling each pixel of an image into a category corresponding to an object or part of the image. A simple formulation: given an RGB image or a gray scale image, we want to find a way to assign a label l_i from a label space $L = \{l_0, ..., l_k\}$ to each pixel. In this work we will always assume that the label l_0 is associated with the background class for convenience.

Semantic segmentation is one of the key problems in computer vision and its importance is strictly related to scene understanding problem which is essential for an increasing number of application, as autonomous driving, human-machine interaction,computational photography, images search engines and augmented reality.

In order to understand how deep learning has approached this task, it is important to note that semantic segmentation is a natural step in the progression from coarse to fine inference. The origin could be located at image classification, which predicts a list of contained objects on a given image. Object detection is the next step, providing additional information as spacial location for those objects. So semantic segmentation with its dense prediction inferring labels for every pixel is a further extension of these problems, Fig 2.1. Further improvement can be instance segmentation, different labels for different instances of the same class, and part-based segmentation, decomposition of an already segmented class into its sub-parts, that is the topic of this work.

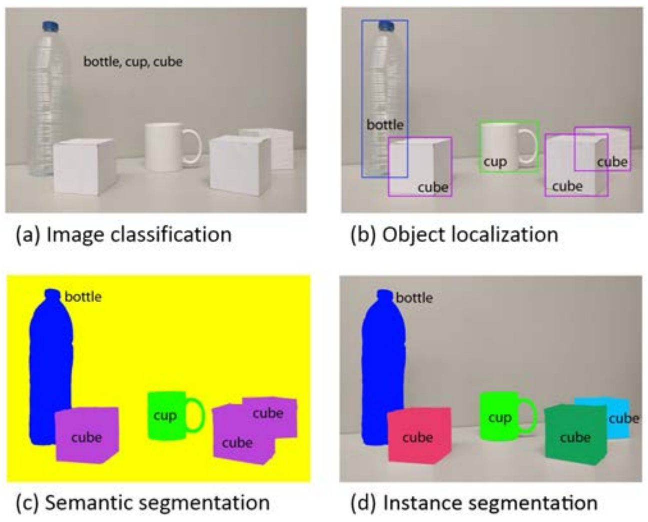


Figure 2.1: Evolution of object recognition or scene understanding from coarse-grained to fine-grained inference: classification, detection or localization, semantic segmentation, and instance segmentation. Figure from [3].

2.1 Convolutional Neural Networks

A convolutional neural network (*CNN*) [1] is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data, and it has widely improved many computer vision fields. The ImageNet challenge (*ILSVRC*), that was dominated by approaches based on feature extraction and traditional ML classifiers until 2012, has showed how neural network could achieve much better results and a gap in performance that would be hard to fill with classical image analysis algorithms.

2.1.1 CNNs for Image Classification

CNNs have made significant contributions to the Image Classification task, their importance is so high that some State-Of-The-Art model are currently being used as building blocks for many segmentation architecture, typically using them for the feature encoding part. In the section we briefly report some of these models.

Alexnet

AlexNet, published by A.Krizhevsky et al. [2], is the first deep CNNs with outstanding performance during 2012 ImageNet competition, its test accuracy of 84.6% outperformed the previous accuracy of 73.8% that was obtained with traditional techniques. This model, that is considered one of the simplest today, consists of: five convolutional filters, max-pooling layers, three fully-connected layers and one dropout, Fig 2.2.

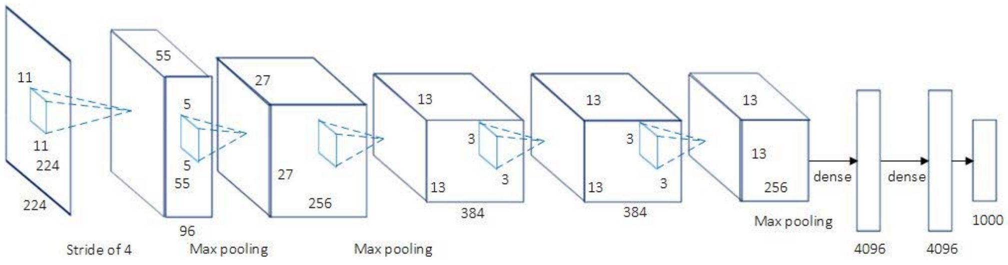


Figure 2.2: AlexNet Convolutional Neural Network architecture. Figure from [2] .

VGG-16

VGG-16, published by Simonyan at al. [6] (2014), won 2013 ImageNet competition with 92.7% accuracy. It is also known as VGG-16 due to the fact it composed of 16 convolutional layers, multiple max-pooling layers and three final fully connected layers, Fig 2.3. Its main difference from previous work are:

- introduction of non-linearities, with ReLU activation function, in order to learn more complex patterns;
- use of convolutional layers with small filters 3x3, due to the fact that allows to have less parameters and with an faster training phase.

GoogLeNet - Inception V1

GoogLeNet, published by Szegedy at al. [5] (2014), won 2014 ImageNet challenge with 93.3% accuracy. It is composed by 22 layers and a newly introduced building block named *inpection module*, for a total of 50 convolutional layers. Each module is composed of 1x1, 3x3, 5x5 convolutional layers and a 3x3 max-pool layer in order to increase sparsity in the model and obtain different type of patterns, Fig 2.4.This approach proved that CNNs layers could be stacked in more ways than a typical sequential manner.

ResNet

Microsoft's ResNet [4] won the 2016 ImageNet challenge with 96.4% accuracy. It is well known due to its 152 layers and the introduction of *residual block*, Fig 2.5. The residual blocks address the problem of training a really deep architecture

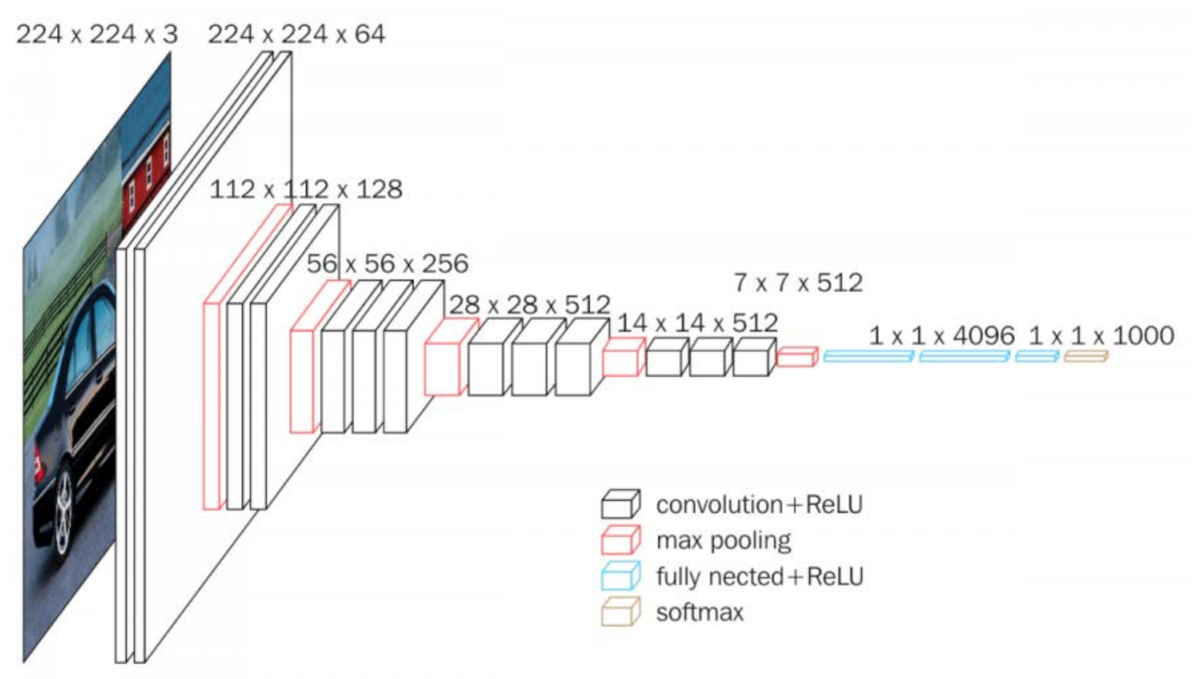


Figure 2.3: VGG-16 CNN architecture. Figure from [6]

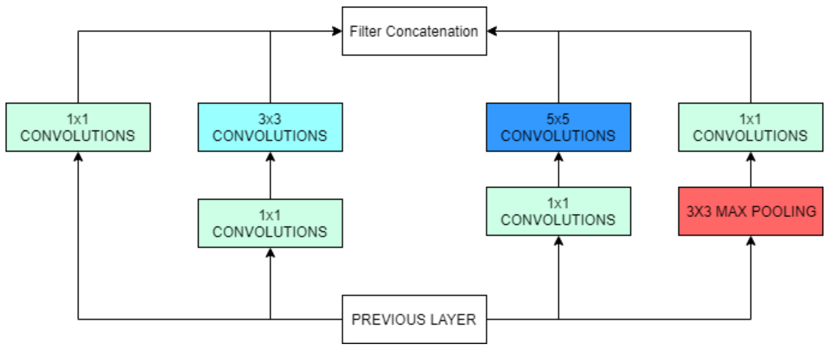


Figure 2.4: Inception module with dimensionality reduction from the GoogLeNet architecture.

by introducing identity skip connections between the output of one or multiple convolutional layers and their original input, this ensures that the next layer learns something new and diffrent from what the input has already encoded. Moreover, this method does not add any additional parameter and does not increase the computational complexity of the model.

2.1.2 CNNs for Semantic Segmentation

Currently, the most successful techniques for Semantic Segmentation are based on the same macro structure that is called *Autoencoder*. These type of models are composed by two main components: the first one, which is called *Encoder*,

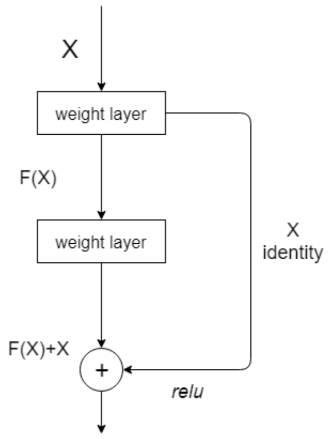


Figure 2.5: Residual block from the ResNet architecture.

is a State-Of-The-Art CNN for Image Classification without its final fully connected layers. The second one, which is called *Decoder*, is the component that reconstructs the prediction starting from the encoder feature maps. Decoders are the components that most determine the difference between those models as they differ in the approaches utilized to upsample the resolution of the feature map.

The possibility to use an Encoder from a state-of-the-art CNN for image classification offers the opportunity to enable *transfer learning.* Training a deep CNN from scratch it could be not feasible because of small size dataset or difficulties on reaching convergence in a reasonable amount of time. Even if large datasets are available and convergence is fast it is still helpful and suggested to start with pre-trained weights instead of random initialization, [9] [10]. Yosinski *et al.* [11] proved that transferring features even from distant task can be better that random initialization. So it is common to use Encoders from classification with their pre-trained weights, even on different dataset, and fine-tuning the Decoder layers with an higher learning rate compared to the Encoder, since the lower part of the network tends to contain more generic features. Here we report some well known CNNs for Semantic Segmentation.

Fully Convolutional Networks

Fully convolutional networks (FCN) is the first model that has adopted an Autoencoder structure, by Long *et. al* [12]. The main idea of this approach was to take advantage of existing CNNs for visual task. They transformed the existing and well-known image classifcation models into fully convolutional ones by replacing the fully connected layers with convolutional ones to output spatial maps instead of classification scores, these maps are upsample to produce dense per-pixel labeled output. This work is considered a milestone since it showed how CNNs can be trained end-to-end for this problem, efficiently learning how to make dense predictions for semantic segmentation with inputs of arbitrary sizes [3].

SegNet

SegNet, by Badrinarayanan *et al.* [13], is a variant of FCN in which the decoder stage is composed by a set of upsampling and convolutional layers which are at last followed by a softmax classier to predict pixel-wise labels, Fig 2.6. Each upsampling layer in the decoder corresponds to a max-pooling one in the encoder. This operation is performed using the max-pooling indices from the corresponding feature maps in the encoder phase. The upsampled maps are then convolved with a set of trainable filters to obtain dense features maps.

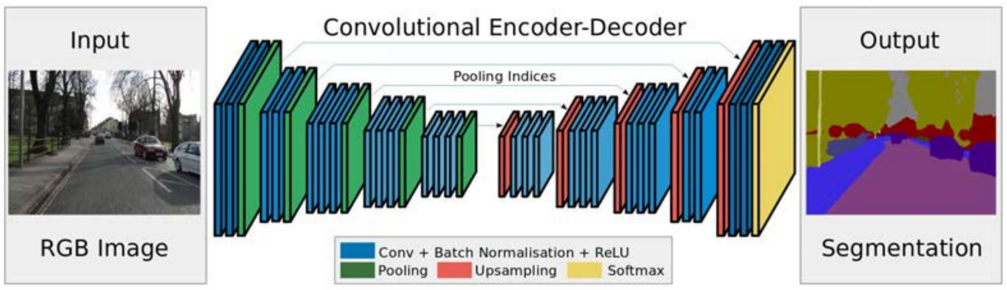


Figure 2.6: SegNet architecture. Figure from [13]

U-net

U-net, by Ronneberger *et al* [14], is an extended version of FCN. They modified the fully convolutional architecture by expanding the capacity of the decoder This model is composed of two parts: a compacting part to compute features and a symmetric expanding part to spatially localise patterns in the image.

FPN

Feature Pyramid Network (FPN), by Lin *et al.* [15], exploits the inherent multi-scale, pyramidal hierarchy of deep convolutional networks to construct feature pyramids with marginal extra cost. A top-down architecture with lateral connections is developed for building high-level semantic feature maps at all scales. This architecture shows significant improvement as a generic feature extractor in several applications.

PSPNet

Pyramid Scene Parsing Network, by Zhao *et al.* [16], was designed to better understanding the global context representation of the scene. Given an input image its features are extracted with a ResNet network modified with the dilated network strategy [19] [20]. Pyramid Pooling Module are fed with the feature maps to distinguish patterns with different scales. They are pooled with four different scales each one corresponding to a pyramid level and processed by a 1x1 convolutional layer to reduce their dimensions. This way each pyramid level analyses sub-regions of the image with different location. The outputs of the pyramid levels are upsampled and concatenated to the inital feature maps to finally contain the local and the global context information. Then, they are processed by a convolutional layer to generate the pixel-wise predictions.

DeeplabV2

DeepLab, by Chen *et al.* [17], has three main characteristics on its architecture. First, down-sampling operator from the last max pooling layers of DCNNs are removed and replaced with atrous convolution layers that up-sample filters inserting holes between non zero filter taps. It also allows us to effectively enlarge the field of view of filters to incorporate larger context without increasing the number of parameters or the amount of computation. Second, in order to handle multiple scale, it is proposed the atrous spatial pyramid pooling (ASPP) technique to robustly segment objects at multiple scales. Finally, they boosted the model's ability to capture fine details by employing a fully connected Conditional Random Field (CRF) [18].

2.2 Part-based Semantic Segmentation

Part-based Semantic Segmentation is a natural extension of Image Semantic Segmentation. Decomposing an object into its semantic parts enables a more detailed understanding of the object. Understanding how different parts are related and located have been an increasingly important topic. This task can provide additional information to benefit some important topics such as pose estimation [21] [22], detection [23] [24] and fine-grained recognition [25]. However object level segmentation over multiple object categories has been extensively studied, while object parsing is addressed only for a few categories as human [26] [27], cars [26] or certain animals.

Human body part segmentation has been considered a very challenging task due to the wide variability of the body parts appearance. There is a large variation due to pose and viewpoint , self-occlusion and clothing. Traditional algorithms [28] find it difficult to handle this large variability in natural images. Even CNN-based approaches [19] [29], although simple and fast, give coarse part details and have local confusion errors, if the person is in a non-typical pose, or when there are some other objects/persons nearby with similar appearance. Other works focus on improving CNNs approaches paying attention to the large scale variation. Chen et al. [30] learn pixel-wise weights through an attention model to combine the part segmentation results of three fixed scales. Xia et al. [31] build an hierarchical model that adapts to object scales and part scales using auto-zoom.

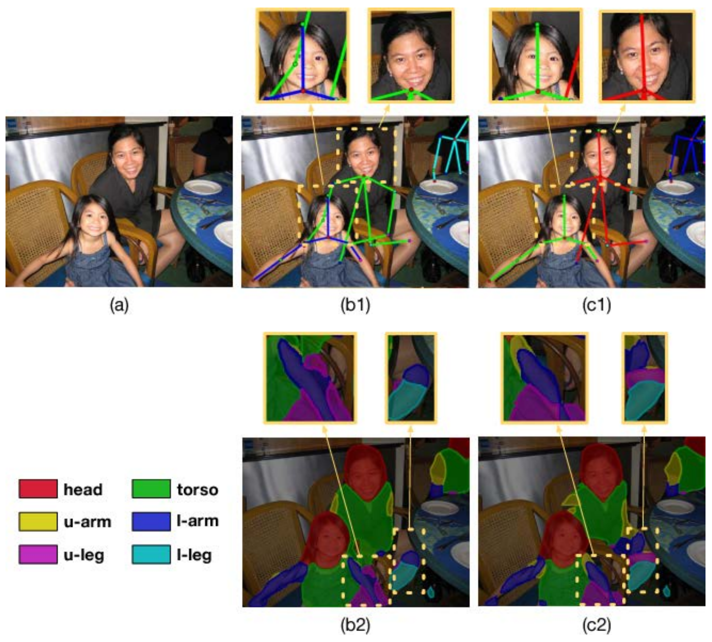


Figure 2.7: Joint human pose estimation and semantic part segmentation improve both tasks. (a) input image. (b) pose estimation and semantic part segmentation results before joint inference. (c) pose estimation and semantic part segmentation results after joint inference. Note that comparing (b) and (c), where (c) are the results from [34], (c1) and (c2) perform better. (c1) recovers the missing forehead joint and corrects the location error of right elbow and right wrist for the woman on the right, while (c2) gives more accurate details of lower arms and upper legs than (b2) for both people. Figure from [34]

Another approach consists on joining semantic part segmentation with a complementary task as pose estimation. The estimated pose provides object-level shape prior to regularize part segments while the part-level segments constrain the variation of pose locations. Yamaguchi et al. [32] perform pose estimation and semantic part segmentation sequentially for clothes parsing, using a CRF with low-level features. Ladicky et al. [33] combine the two tasks in one formulation, using also low level features. However these methods cannot deal with images with large pose variations or multi-person overlapping, mainly due to the less powerful features they use or the poor quality of their part region proposals. Xia et al. [34] exceeds this problem combining FCNs with graphical models, greatly boosting the representation power of models to handle large pose variation. Moreover it introduces a novel part segment consistency terms for pose estimation and novel pose consistency terms for part segmentation, further improving the performance, Figure 2.7.

2.3 Incremental Learning

The Incremental Learning (IL) problem is defined as the capability of machine learning architectures to continuously improve the learned model by feeding new data without losing previously learned knowledge [43]. This is a challenging task since traditional learning models require that all sample the old and the new ones are available during all step of the training stage while an IL model should keep learning without storing old data. This task has been widely studied in the context of problems like image classification and object detection.

Some methods [38], [39] exploit network architectures which grow during the training process as new classes are observed. Istrate et al. in [40] proposed a method that partitions the original network into sub-networks which are then gradually incorporated during the training phase. On the other hand Sarwar et al. in [41] introduce a network that incrementally grows over time while sharing parts of the base network. Alternatively, a different strategy consists in freezing or slowing down the learning process in some parts of the network. Kirkpatrick et al. [42] developed Elastic Weight Consolidation (EWC) to remember old tasks by slowing down learning on certain weights based on how important they are to previously seen tasks.

Papers mentioned before focus on image classification or object detection while only few articles as [43] and [44] investigates on incremental learning for Semantic Segmentation. These works applies the distillation approach, a technique originally proposed in [46], [47] to preserve the output of a complex architecture of networks when adopting a simpler network for more efficient deployment. In [44] the authors applied knowledge distillation to preserve old classes while incrementally enlarge the capabilities of the learning architecture to properly segment and label new classes.

3

Proposed Models

In this thesis we start from the basic idea behind the Incremental Learning approach, the network should not forget what has learned before. From this idea we research how a network should use previous acquired knowledge to learn new task more efficiently. So we use two networks that can be consider as a cascade where the output of the first one will be the input of the second one. The first network is trained on a generic task, Semantic Segmentation on 21 classes from Pascal VOC dataset, while the second network is trained on a more detailed task, Semantic Part Segmentation on Pascal Part dataset which provides sub-classes for the previous 21.

3.1 DeeplabV2

As previously stated, there are not works that perform Semantic-Part-Segmentation with the entire Pascal-Part dataset, so we don't have results to use as basis of comparison. We needed a State-of-Art Networks that allowed us to:

- collect baseline results training a network on a refined version of Pascal-Part dataset;
- obtain additional data useful for our approach from Pascal-VOC dataset.

For this reasons we start with a basic implementation of a State-of-the-Art CNN. Our choice falls on an autoencoder network DeeplabV2 with a ResNet101 with Dilation Rate layers as encoder. It is one of the networks with the higher accuracy on Pascal-VOC challenge.

3.2 Proposed Model

As introduced, we aim to improve the obtained baseline results on Pascal-Part using additional features extracted from the outputs of a CNN trained on Pascal-VOC, so we have investigated two way to represent these outputs:

- *Softmax*: the outputs of the last Activation layer with size $H * W * N$ where N is the number of channels and classes. So each value p on the i-th channel, $H * W * i$ with $i \in \{ 1 \dots \text{N} \}$, is the probability of p to belong to the i-th class. The idea of using the softmax is derived from [45].

- *Argmax*: the result of the *argmax* function applied on the *softmax*. So a matrix with size $H * W * 1$ where each element value $p \in \{ 1 \dots \text{N} \}$ is the index channel of the Softmax with the higher probability.

These types of outputs bring different amount of additional information, on Chapter 5 we analyse how it affects the accuracy.

3.2.1 Model

The DeeplabV2 CNN with ResNet101 as encoder is a quite complicated architecture and its training time is quite significant. Since we don't want to make it more intricate, we have implemented a simple network to process argmax and softmax, usually basic CNNs architecture are made up of *Convolutional, Pooling* and *Fully-Connected* layers. The most common form stacks a few Convolutional with *Relu* activation layers, follows by Pooling layers, and repeats this pattern until the image has been merged spatially to a small size, in our case we use four Conv-Relu layers which applies elementwise non-linearity and one Max Pooling layer.
Through Convolutional operation, a specialized type of linear operation, it's possible to extract feature map from the input tensor, moreover applying Max Pooling layers a downsampling operation is performed which progressively reduces the spatial size of the representation to decrease the amount of parameters and computation in the network, and hence to also control overfitting, it extracts patches from the input feature maps and it outputs the maximum value in each patch and discards all the other values. Once we create a simple architecture to obtain additional data from the argmax\softmax inputs, we must feed them to the main network, so they are concatenated with the encoder output, Figure 3.1. In this

way informations from the Encoder and our simple network are joined and passed to the Decoder. This technique follows the idea of using the features learned our network to improve the performances of the Decoder.

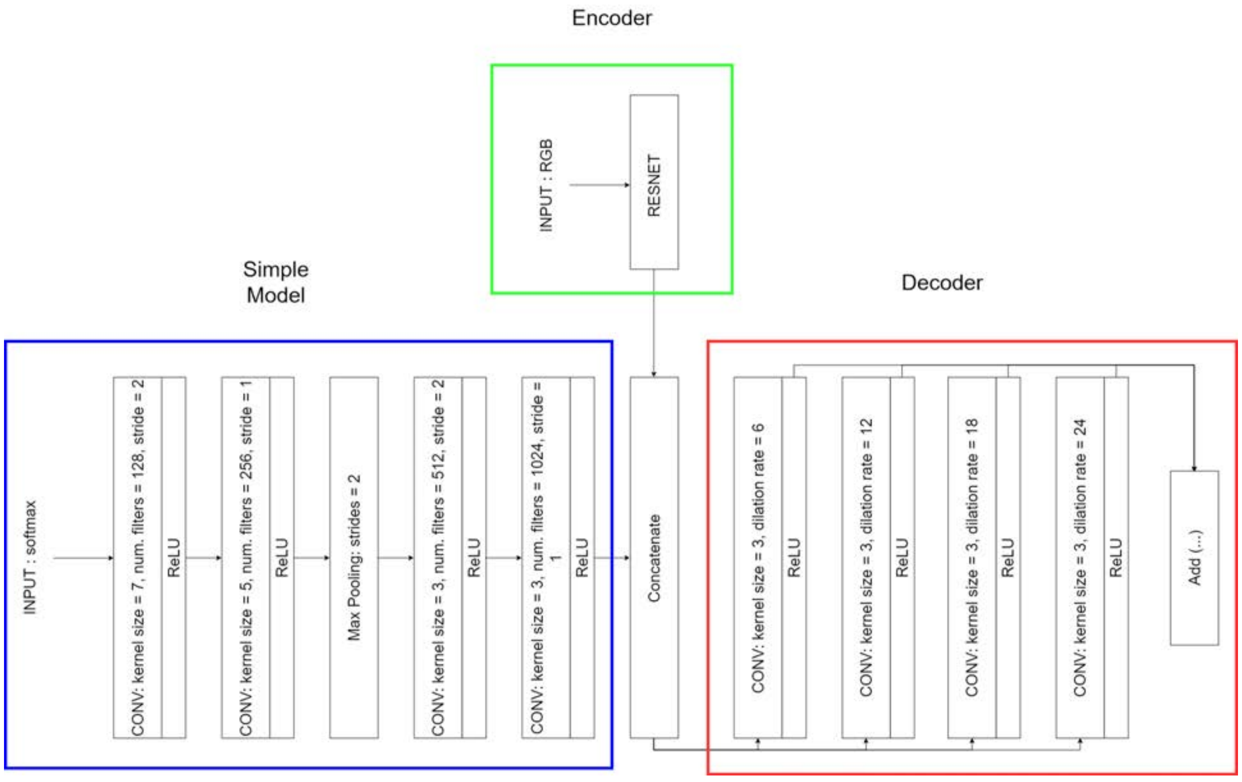

Encoder
INPUT : RGB
RESNET
Simple Model
Decoder
INPUT : softmax
CONV: kernel size = 7, num. filters = 128, stride = 2
ReLU
CONV: kernel size = 5, num. filters = 256, stride = 1
ReLU
Max Pooling: strides = 2
CONV: kernel size = 3, num. filters = 512, stride = 2
ReLU
CONV: kernel size = 3, num. filters = 1024, stride = 1
ReLU
Concatenate
CONV: kernel size = 3, dilation rate = 6
ReLU
CONV: kernel size = 3, dilation rate = 12
ReLU
CONV: kernel size = 3, dilation rate = 18
ReLU
CONV: kernel size = 3, dilation rate = 24
ReLU
Add (...)

Figure 3.1: First proposed model.

A simple variation of 3.1 is showed on Figure 3.2. In this case, the network is simpler, faster and with a reduced memory load. On this variant the output of the *Concatenation* layer is pass only to the first layer of the DeeplabV2, while the others layer have as input only the Encoder output. We can see in Chapter 4, how this choice does not change the accuracy results, probably because useful information can be extracted from the first Decoder layer and the information from other layers are redundant.

3.2.2 Model variation with Dilation-Rate layers

Another variation is shown on Figure 3.4. In this case the output of our simple network for processing softmax is given as input to four dilation rate or *atrous* layers, moreover the outputs are concatenated with the output of the same kind of layer from the decoder. These layers insert “spaces” in the kernel matrix to capture features of the input tensor at a different scale. Compared to the increase

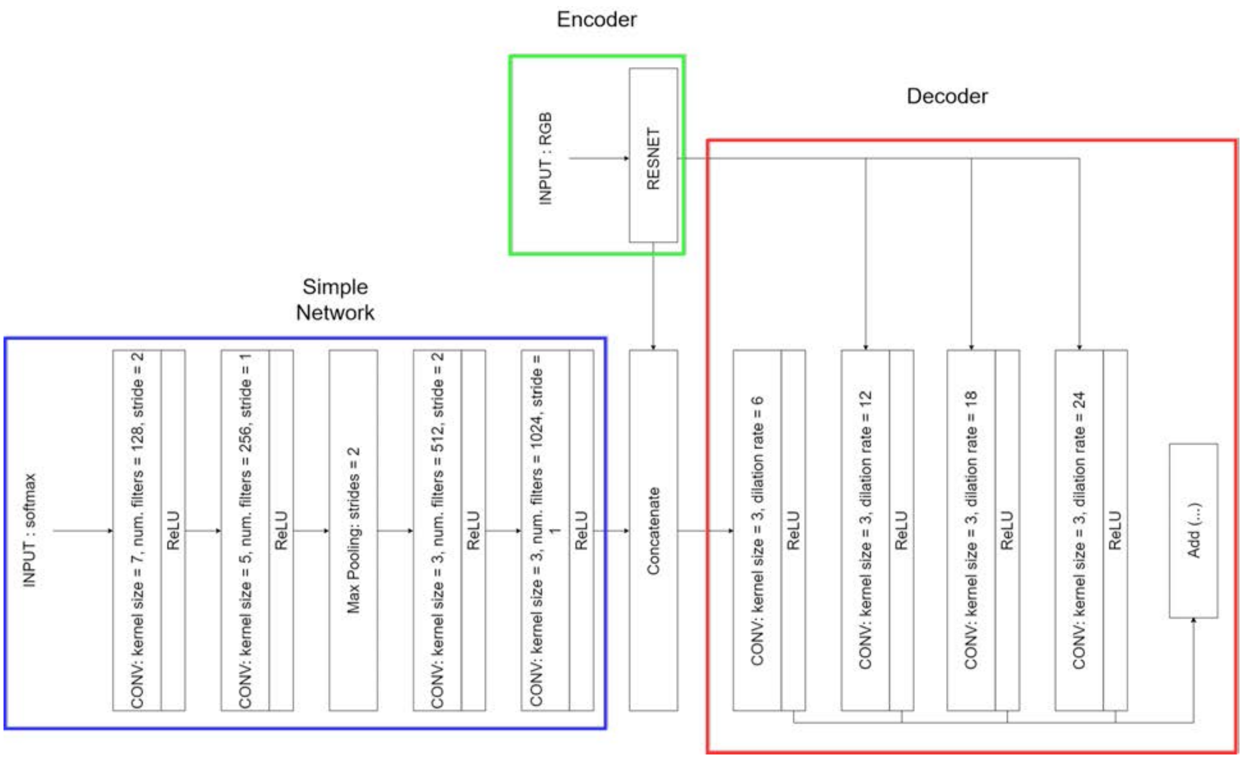

Encoder
INPUT : RGB
RESNET
Decoder
Simple Network
INPUT : softmax
CONV: kernel size = 7, num. filters = 128, stride = 2
ReLU
CONV: kernel size = 5, num. filters = 256, stride = 1
ReLU
Max Pooling: strides = 2
CONV: kernel size = 3, num. filters = 512, stride = 2
ReLU
CONV: kernel size = 3, num. filters = 1024, stride = 1
ReLU
Concatenate
CONV: kernel size = 3, dilation rate = 6
ReLU
CONV: kernel size = 3, dilation rate = 12
ReLU
CONV: kernel size = 3, dilation rate = 18
ReLU
CONV: kernel size = 3, dilation rate = 24
ReLU
Add (...)

Figure 3.2: Variation of the first proposed model.

in the kernel size, these layers are a cheaply way to increase the receptive field of output units, which is especially effective when multiple dilated convolutions are stacked one after another, Deeplab network take advantage of this properties.

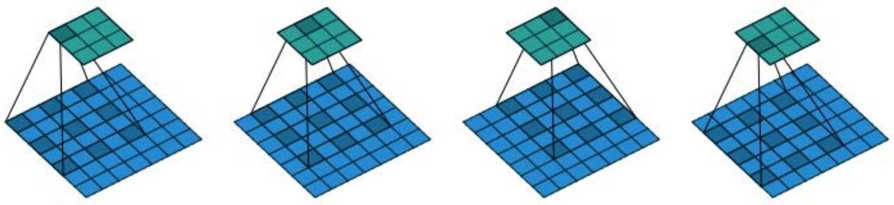


Figure 3.3: Dilated convolution, example convolving a 3x3 kernel over a 7x7 input. Figure from [35] with a dilation factor of 2

Some model variations

On previous model we have made some small changes on the softmax \argmax network to check if it could be possible to have more improvements. We increase and decrease the channels number of the Conv2D layers. We add some layers on the network to extract more features from the softmax \argmax network, we try different kernel initializer and add some BachNormalization ad Dropout layer in order to regularize the output.

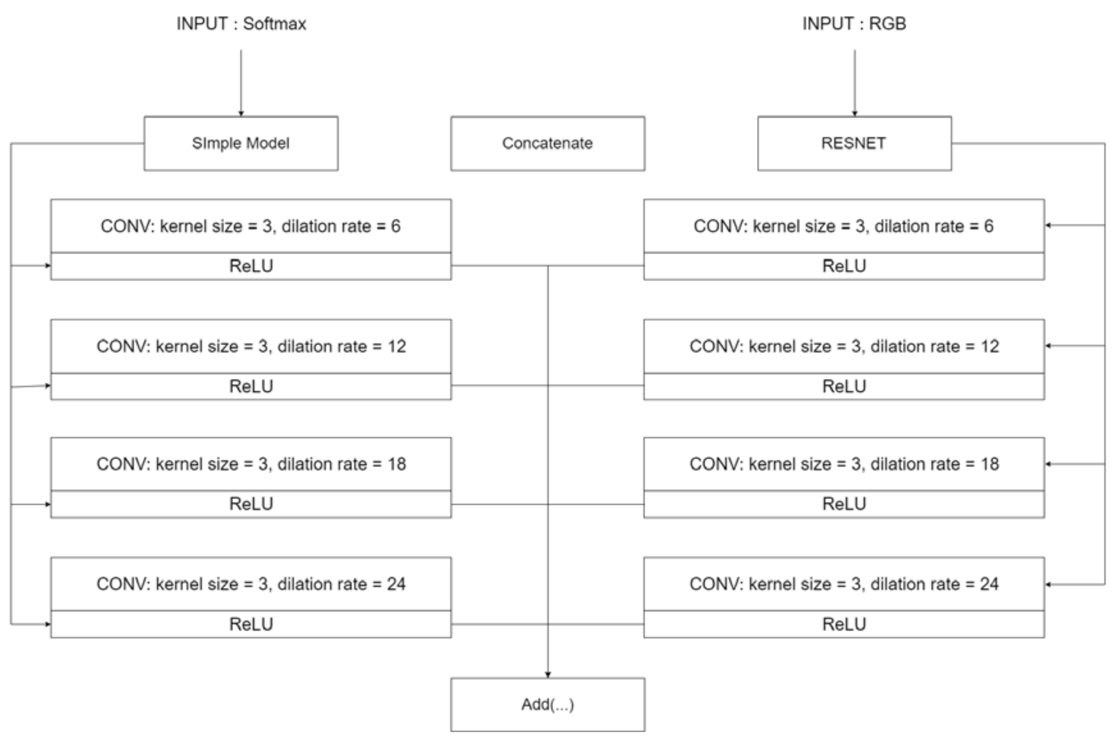

INPUT : Softmax
INPUT : RGB
SImple Model
Concatenate
RESNET
CONV: kernel size = 3, dilation rate = 6
ReLU
CONV: kernel size = 3, dilation rate = 12
ReLU
CONV: kernel size = 3, dilation rate = 18
ReLU
CONV: kernel size = 3, dilation rate = 24
ReLU
Add(...)

Figure 3.4: Proposed model with dilation rate layers.

3.3 Adjacency-Matrix loss

As discuss on Chapter 1, we want to teach to the network how to discriminate parts that do not belong to the same class and that should not be neighbour. For this aim we start from the definition of graph and adjacency matrix.

An undirected graph is a pair G = (V,E) consisting of finite set V of vertices and the finite set E of edges, which are unordered pairs of elements of V. From graph theory we take into account the *adjacency matrix* structure i.e. a symmetric square matrix $|V| * |V|$ used to represent a finite graph. The elements of the matrix indicate whether pairs of vertices are adjacent or not in the graph. Given an adjacency matrix A of G, $\forall q_{ij} \in A$:

$$q_{ij} := \begin{cases} 1 \ \ if[i,j] \in E \\ 0 \ \ if[i,j] \notin E \end{cases}$$

Starting from the idea to define a new loss which goal is teaching to the network which parts are correlated and penalizing cases in which uncorrelated parts appear to be close. We define for each ground truth image its adjacency matrix, where each part that occurs is considered as a graph vertex. So given an image which contains an horse and person its adjacency matrix has a similar appearence to that in Figure 3.5.

During the training, for each predicted batch its adjacency matrix is calculated, one common matrix for all batch images, in this case if we consider a batch size $\geq$2 the matrix could not be a binary matrix since the matrix will be the sum of the adjacency matrices of each batch image and if more images contain the same class with the same parts the final matrix will have entries > 1.

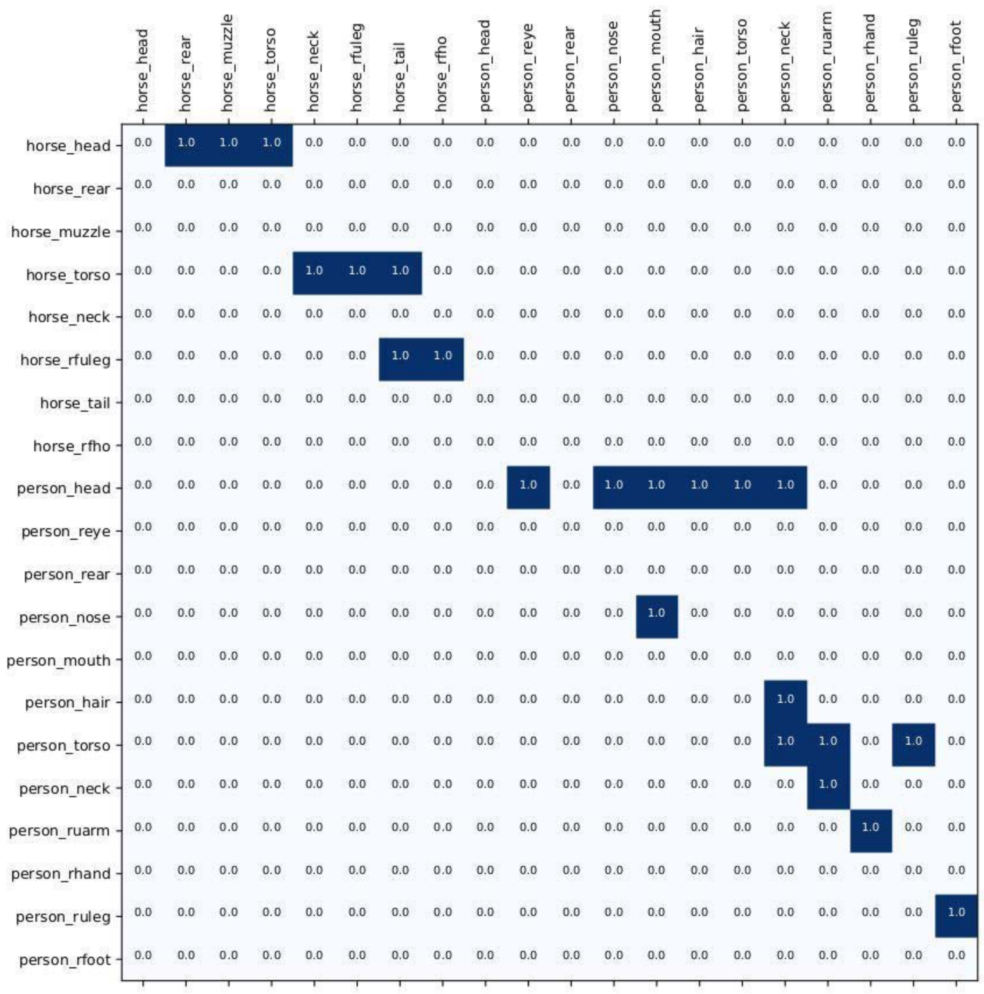
The resulting matrix is then compared to the corresponding matrix from the ground truth with a distance function. It could be an *L1* and *L2* loss, respectively Eq. [3.1] and [3.2]. If the Adjacency matrix is a binary matrix then applying an L1 loss or and L2, the result will be the same. However this will no be true for most of the cases and for the Weighted Adjacency matrix that we will present on Paragraph 3.3.2.

$$L1 = \frac{1}{|V|^2} \sum |q_{ij} - k_{ij}| \quad with \quad q_{ij} \in A_{Adj-ground-Truth}; \quad k_{ij} \in B_{Adj-batch} \tag{3.1}$$

$$L2 = \frac{1}{|V|^2} \sum (q_{ij} - k_{ij})^2 \quad with \quad q_{ij} \in A_{Adj-ground-Truth}; \quad k_{ij} \in B_{Adj-batch} \tag{3.2}$$

3.3.1 Brief implementation description

Given a ground truth gray scale image, we need to find a way to extract information about which parts are close to each others, but it's not easy to determine if two region with different values are adjacent. We can easily extract from an gray scale image patches with the same values creating masks with boolean operators, however it is not so easy to define their closest patches values.
A possible solution could be to find not patch regions but their perimeters, if we find all contour points for all patches, it is possible to compare these contours point to point and find a relation between patches. From the *OpenCV* library for computer vision tasks, we can use the *cv.findContours* method, which returns a list of point for each patch that it finds, however these are not all contour points



	horse_head	horse_rear	horse_muzzle	horse_torso	horse_neck	horse_rfuleg	horse_tail	horse_rfho	person_head	person_reye	person_rear	person_nose	person_mouth	person_hair	person_torso	person_neck	person_ruarm	person_rhand	person_ruleg	person_rfoot
horse_head	0.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
horse_rear	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
horse_muzzle	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
horse_torso	0.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
horse_neck	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
horse_rfuleg	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
horse_tail	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
horse_rfho	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
person_head	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0
person_reye	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
person_rear	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
person_nose	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
person_mouth	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
person_hair	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
person_torso	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	1.0	0.0
person_neck	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
person_ruarm	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
person_rhand	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
person_ruleg	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
person_rfoot	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Figure 3.5: Example of Adjacency matrix for an image with a person and a horse.

but only a simplified description of a perimeter patch so it is not always possible decide if two patches are close looking for the same points on their contours. Starting with a list of points for a first patch A, we evaluate the distance from each point from A and all points from another contour patch. If there is a distance $\leq$1 we set as close the corresponding parts on the Adjacency matrix. All this operations are conducted with matrix-vector operations to be faster.

3.3.2 Weighted Adjacency matrix

Starting from the above proposed loss, we create a simple variation trying to add some information about how much two parts must be close. In this case the adjacency matrix it is not a binary matrix, each element is an approximate value about how many pixels two parts have in common. These could be helpful because we are penalizing firstly part that should not appear closest, secondly parts that should appear close following a proportion rule based on their semantic meaning. So our model should learn that a head and torso person should have in common an approximate number of contour pixels that moreover should be higher than a hand and an arm, while a head and a hand should not have neighboring pixel.

Following the approach described on the previous paragraph with the *Opencv* function, in order to find an approximate number of common pixels, we simply count the number of contour-points for each part, whose distance from another contour-point of another part is less of a distance parameter *pix*, Eq. [3.3].

$$W_{p_i,p_j} = |d(p_{i_k}, p_{j_z}) \leq pix| \tag{3.3}$$

Where:
p_i, p_j = two different parts with: $i \neq j, \quad i, j \in [0, 107]$
pix = pixel distance parameter
k = define the $k-th$ point $\in$ contour of p_i
z = define the $z-th$ point $\in$ contour of p_j

3.3.3 Multiple Weighted Adjacency matrices

Previous losses create a common weighted adjacency matrix for all batch images, this could influence in wrong way the learning phase of our model. If we consider only one Adjacency matrix independently from the batch size, we evaluate a single loss for all batch images. Since the *L2* loss is not a linear function, if we apply it to a common adjacency matrix for all batch images, its result will be different from applying it on a set of adjacency matrices, one for each batch image, and then summing the results. If we evaluate *L2* for each single image we should have a more accurate value to describe the training situation for that batch.

4

Dataset

4.1 PASCAL-VOC 2010

The PASCAL (i.e *pattern analysis, statistical modelling and computational learning*) dataset [7] is a benchmark in visual object category recognition and detection, providing the vision and machine learning communities with a standard dataset of images and annotation. Each given image has an annotation file with object class label for each object in one of the twenty-one classes present in the image. Table 4.2, reports a list of the dataset classes.

Table 4.1: PASCAL classes and their index and parts number.

Index	Class	#Parts	Index	Class	#Parts
1	airplane	5	11	table	1
2	bicycle	4	12	dog	10
3	bird	8	13	horse	8
4	boat	1	14	motorbike	4
5	bottle	2	15	person	12
6	bus	8	16	potted plant	2
7	car	7	17	sheep	8
8	cat	9	18	sofa	1
9	chair	1	19	train	7
10	cow	8	20	tv	1

These data are divided in Train, Validation and Test sets, however for the last one annotations are not provided. In order to have a Test with annotations to evaluate our models, we use the original Validation set as Test set and create a new Validation set randomly taking 500 images from the Training set. These images are taken trying to preserve class distribution over the sets, see Figure 4.1.

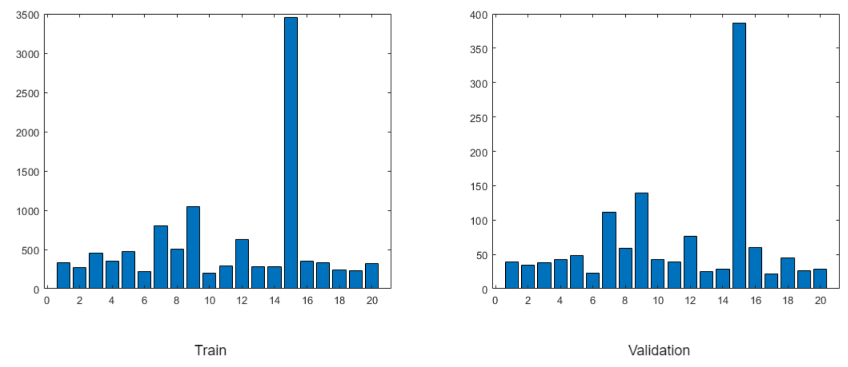


Figure 4.1: Classes distribution over Train and Validation set.

4.2 PASCAL-PART

For our tests, we evaluate the previous models on the Pascal-Part dataset [8], which auguments PASCAL -VOC classes with pixel-wise semantic part annotations, Figure 4.2 shows how each class from Pascal-VOC2010 is split into its sub classes. For example if we consider the first line of the image, it's possible to see how the baby (*person* class) is divided into: head, ears left \right, eyes , hair, nose, neck, mouth, torso, arms and leg lower \upper left \right, hand and foot left \right. We can understand how this partition could be too extreme, the Pascal-Part provides 384 parts that are too many and too detailed, so some refinements are needed in order to reduce this number. From [37] we follow these rules:

1. discard fine-grained labels (e.g 'car wheel front-left' and 'car wheel back-left' are both mapped to *car-wheel*);
2. merge contiguous sub-parts of the same larger part(e.g 'person upper arm' and 'person lower arm' become a single *person-arm*);
3. discard very tiny parts (average of widths and heights over the whole training set <= 15 pixels);
4. discard parts with less than <= 10 samples in train set (like 'bicycle headlight');
5. consider classes without sub-part as a single part object.

The final dataset contains 103 parts and 4 classes with a single part for a total number of 108 classes considering the background. The parts have been mainly reduced following rules 1 and 2, while few parts that are reported on Table 4.2 have been removed due to rules 3 and 4. Table 5.1 reports the number of parts for each class while Figure 4.3 shows all parts for each class. It is important to notice that all the studies on Part-Based Semantic Segmentation are focused on parts of single or few classes, human or animal-parts, so the number of classes is highly reduced in comparison with our case, even for the human parts usually. Only [37] takes into account all parts from the Pascal-Part however its task is Image Classification.

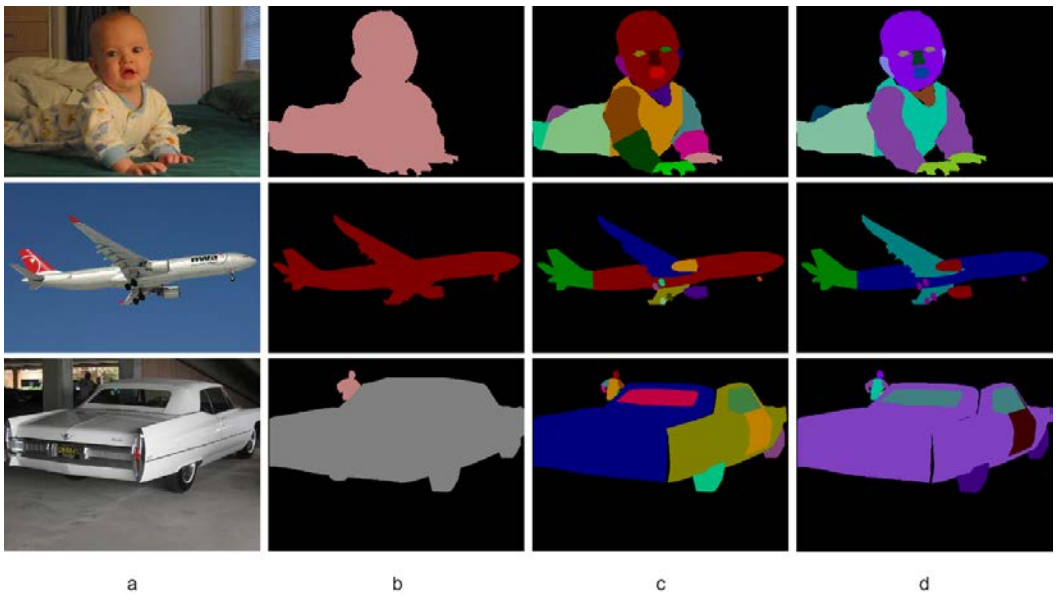


Figure 4.2: a) RGB PASCAL-VOC dataset; b) Labels from PASCAL-VOC; c) Labels from PASCAL-part; Labels from PASCAL-part after refinement.

Table 4.2: Removed parts.

Class	# Parts
aeroplane	tail
bicycle	headlight
bird	r\l eye
car	r\l mirror
cow	r\l eye
horse	r\l eye
person	r\l brow
sheep	r\l eye

For a statistical study we have calculated the pixel distribution of the 107 parts over the entire dataset, we do not consider the background, Figure 4.3 shows this distribution in a logarithmic scale. The values are reported in logarithmic scale because there is a remarkable gap value between classes especially for the little ones that moreover have less occurrences. Since some parts have a small number of occurrences and an object does not always show all its parts on an image, even if the validation set show the same class distribution as the training, the part distribution does not correspond and so the Validation set lacks of two parts. For this reason on this thesis we reports only the results on the Test set. The Validation set was used only to choose the model's parameters.

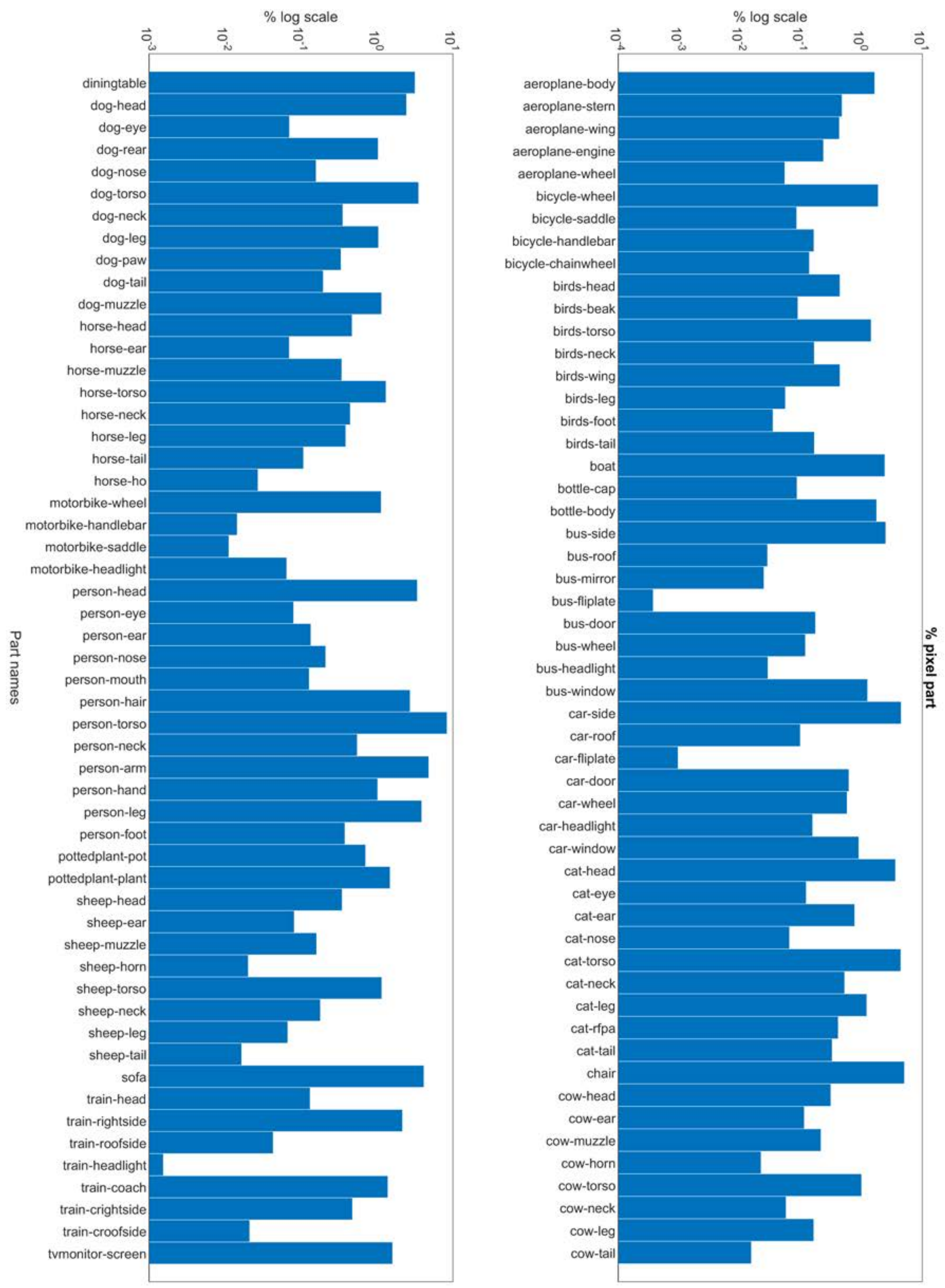


Figure 4.3: Per-class percentage of pixels in logarithmic scale.

5

Results

5.1 Experimental Setup

The original DeeplabV2, which this thesis takes as State-of-the-Art network for semantic segmentation to develop our model, was originally developed using *Tensorflow*. We re-implemented this network with *Tensorflow-Keras* version 1.13 because we want to quickly build and test a neural network with a reduced number of lines of code.

To perform the training procedures we utilized a single Nvidia 1070 GPU, which has 8GB of dedicated memory. The limited amount of available resources forced us to use small batch and resize images to a smaller resolution unless the networks would not fit in memory, as done by all concurrent approaches in Semantic Segmentation to choose training parameters we have performed some empirical trials changing the learning rate, the image size and the number of epochs. We have chosen parameters which lead to an higher *mean intersection over union* (mIoU) as metric. Intersection over union, or *Jaccard coefficient*, measures the similarity between the predicted image and its ground truth. It is defined as Eq. [5.1] where true-positives are those pixels that belong to the class and are correctly predicted, false-negatives are those pixels that belong to the class but are erroneously predicted as a different class and false-positives are those pixels that belong to a different class but are predicted as the considered class. The mIoU is then the mean value across all the classes in the dataset. More intuitively, the IoU can be seen as Eq. [5.2] as the size of the intersection divided by the size of the union of the ground truth and prediction.

$$IoU = \frac{TruePositives}{TruePositives + FalseNegatives + FalsePositives} \tag{5.1}$$

$$IoU = \frac{groundTruth \cap predicted}{groundTruth \cup predicted} \tag{5.2}$$

In particular, we trained the network using the standard technique proposed in [17] with Stochastic Gradient Descent (SGD) as optimizer with momentum set to 0.9. The learning rate policy follows the proposed *polynomial decay* policy, the lr is multiplied by Eq. 5.3 with power 0.9:

$$lr = lr * (1 - (\frac{iter}{maxIter}))^{power} \tag{5.3}$$

Additionally, some analysis will also be conducted using other two metrics namely: the mean Pixel Accuracy (mPA) and the mean Class Accuracy (mCA), which are defined as:

$$mPA = \frac{TruePos. + TrueNeg.}{TruePos. + TrueNeg. + FalseNeg. + FalsePos.} \tag{5.4}$$

$$mCA = \frac{\sum^{\#Classes} \frac{TruePos.}{TruePos.+FalseNeg.}}{\#Classes} \tag{5.5}$$

The mPA, Eq. [5.4], simply reports the percent of pixel in the image which are correctly predicted, while the mCA, Eq. [5.5], evaluates the percentage of correctly classified pixels for each class independently and averaging the values [49] [50].

Tensorflow & Keras

Created by the *Google Brain* team, TensorFlow [48] is an open source library for numerical computation and large-scale machine learning. The biggest benefit it provides for machine learning development is abstraction. Instead of dealing with the practical details of implementing algorithms, or figuring out proper ways to feed the output of one function to the input of another, the developer can focus on the overall logic of the application. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. TensorFlow can train and run deep neural networks for handwritten digit classification, image recognition, word embeddings, recurrent neural networks, sequence-to-sequence models for machine translation, natural language processing. It uses Python to provide a convenient front-end API for building applications with the framework, while executing those applications in high-performance C++.

Keras is a high-level API to build and train deep learning models and is usually used for fast prototyping. It does not do its own low-level operations, such as tensor products and convolutions, it relies on a back-end engine for that. Even though Keras supports multiple back-end engines, its primary backend is TensorFlow. Keras models are made by connecting configurable building blocks together, there are two main types of models available the Sequential model, and the Model class used with the functional API. Neural layers, cost functions, optimizers, initialization schemes, activation functions, and regularization schemes are all standalone modules that you can combine to create new models. Its friendliness offers consistent and simple APIs minimizing the number of user actions required for common use cases.

5.2 State of the Art and Baseline

Firstly, the DeeplabV2 (without CRF) with Resnet101 encoder is trained on Pascal-VOC 2010 dataset to extract additional data that we are going to use on our model during the "incremental" step toward the part segmentation. We fine tune the network starting from weights that were pre-trained on the MSCOCO dataset [36]. During training, images and batch size are set respectively to 321×321 and 2 while the learning rate is set to 10^{-4}. Other simulation with different parameters were made, with image size 256×256 or 512×512 and different learning rate. We chose the resolution achieving the highest accuracy and lower time for convergence. Figure 5.1 shows some predicted images compared to their ground truth.
Since we are going to extract additional information from these results, it must be underlined how bad results could condition adversely our model, some imperfections or errors that can be seen Figure 5.1 could be propagated when performing the incremental step. For example Figures 5.1c and 5.1k show some pixels that are labeled as Person(pink), in spite of the background, and this could have a negative influence. However our baseline network is one of the best performing State-of-the-Art architectures for Semantic Segmentation. Indeed, we can appreciate in Table 5.1 the IoU for each Pascal class and the mIoU is 72.28 %.

The output of the last Activation layer with softmax function is saved after and before the application of the argmax, so we save two types of output, a first one is matrix with size $H\times W\times N$, where N is the number of classes and a second one a simple matrix $H \times W \times 1$.

Table 5.1: Results from DeeplabV2 with Pascal-VOC 2010, 72.28 mIoU with batch size = 2, image size $321 * 321$ and learning rate 10^{-4}.

Class	IoU	Class	IoU
background	90.71 %	-	-
airplane	80.88 %	table	54.49 %
bicycle	71.81 %	dog	83.20 %
bird	85.05 %	horse	75.28 %
boat	46.55 %	motorbike	66.68 %
bottle	74.18 %	person	77.04 %
bus	88.42 %	potted plant	70.25 %
car	85.34 %	sheep	81.06 %
cat	89.10 %	sofa	39.43 %
chair	41.58 %	train	79.03 %
cow	66.57 %	tv	71.27 %

As previously stated in Chapter [3], first of all need to compute baseline results to use a basis comparison. Starting from the previous network we changed channel the number on the last layers from 21 to 108, where 108 is the new number of classes / parts. Considering the amount of classes and parts, for a better understanding of the results each part will be related to a specific color, so the same part on different images will be represent in the same way, this is true also for classes.

We train it on the Pascal-Part dataset for 14 epochs with batch size of 2 and learning rate of 10^{-3} and we obtained 29.9 % of mIoU. As it is clear from the third column of Figure 5.2, which shows some inference images, this model does not perform very well. Comparing the results with the ground truth we can notice different types of errors. First of all, the model does not define correctly the shape of the objects as it is clear from image 5.2*c*. Moreover it commits two types of logic errors, the first one is less relevant and it is showed on image 5.2*g*, here the model tags different parts of the bird with the same label, it recognize the bird but not its parts, the second is more serious and it is shown on 5.2*c* where some pixels on paws, head and body of the cat are predicted as parts of unrelated classes. Some paws pixels are labelled as a person-torso (turquoise), some on the body as bird-body (dark-red) and few pixels on the head as dog-head (beige). This error even is more clear on Figure 5.7 where we compare images predicted with each model proposed, here on Figure 5.7*l* the entire body of the cow (light green) is wrongly labelled as a dog body (light blue).

In order to make a comparison with the "Standard" Semantic Segmentation task, we decided to remap the results from the Pascal-Part into the 21 classes from Pascal-VOC 2010, so each part-label is remapped into its class-label. This mapping is extremely useful because it allows us to understand if our model could improve the Semantic Segmentation task or how much worst it is compared to the State-of-the-Art models for that task. Additionally, when we will condition our architecture with the output of the previous model working on the set of 21 classes we would like not to damage the initial prediction after predicting the part-labels. In this case after remapping the baseline results show 60.20 % of mIoU vs 72.28 %. The fourth column Figure 5.1 shows some examples remapped while Table 5.2 reports the IoU for each class, where we can appreciate that only on class *cow* the Baseline model has a higher IoU, 68.59 vs 66.57. Moreover it can be useful to easily notice which pixels are wrongly predicted from a different class.

Table 5.2: Comparison between results on Pascal-VOC 2010 and results on Pascal-Part after we have merged the resulting parts. The per-class IoU reported below is computed starting from the predictions of the sub-parts.

Class	IoU	Baseline IoU	Class	IoU	Baseline IoU
background	90.71 %	88.59 %	-	-	-
airplane	80.88 %	65.92 %	bus	88.42 %	81.44 %
bicycle	71.81 %	59.77 %	car	85.34 %	71.77 %
bird	85.05 %	69.29 %	cat	89.10 %	76.07 %
boat	46.55 %	43.52 %	chair	41.58 %	21.16 %
bottle	74.18 %	56.77 %	person	77.04 %	72.75 %
table	54.49 %	35.75 %	potted plant	70.25 %	45.93 %
dog	83.20 %	72.89 %	sheep	81.06 %	64.23 %
horse	75.28 %	62.39 %	sofa	39.43 %	36.70 %
motorbike	66.68 %	42.89 %	train	79.03 %	68.86 %
cow	66.57 %	**68.89** %	tv	71.27 %	58.77 %

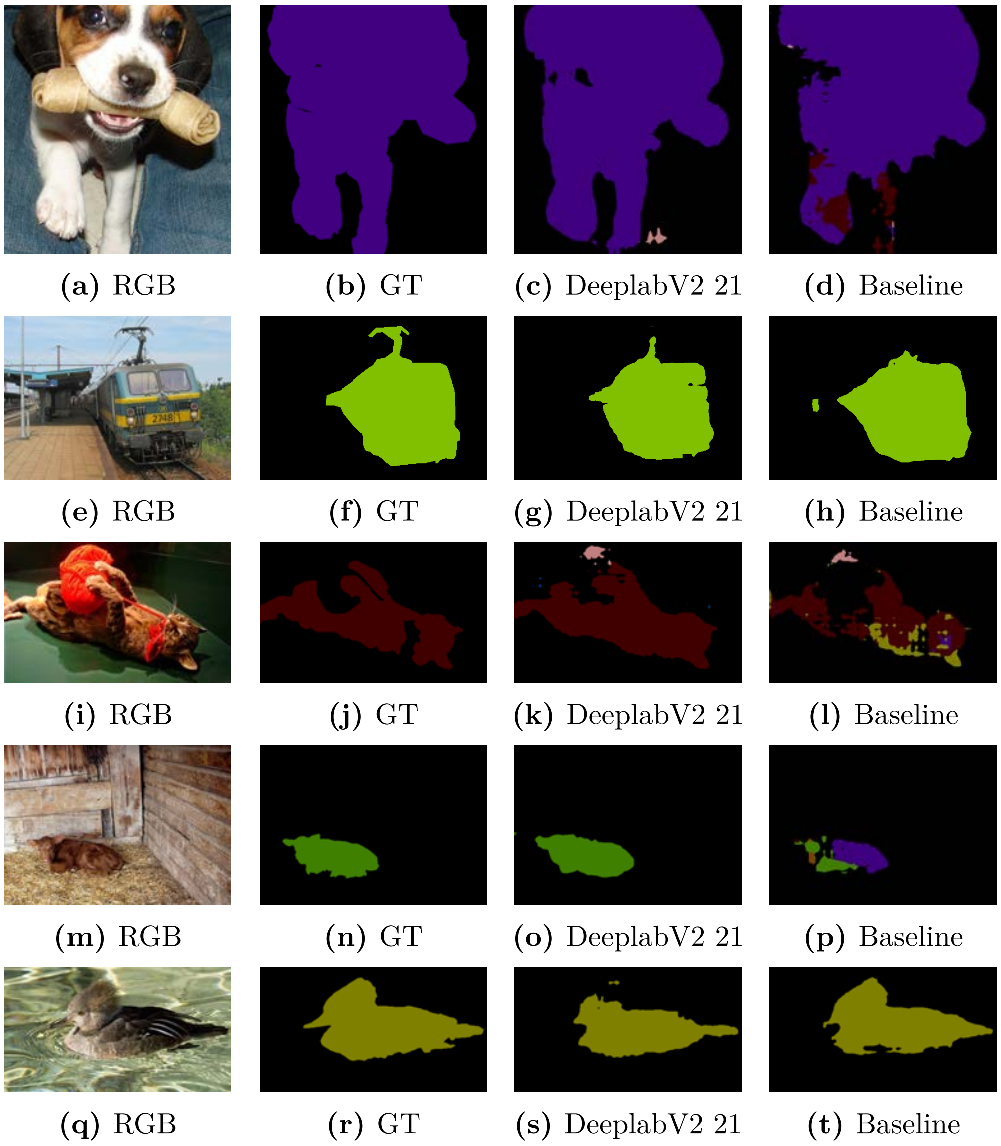


Figure 5.1: From left to right, RGB images, Ground Truth from Pascal-VOC 2010, results from DeeplabV2 and remapped Baseline results from Pascal-Part.

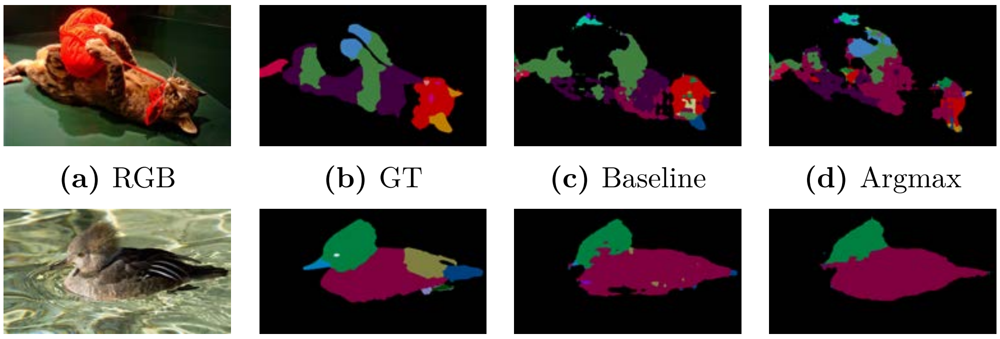
5.3 Conditioning with Argmax

On Chapter 3 we discussed how to extract additional features from the results of the argmax function applied on the network output. We show the results on Table 5.3 where the first line reports the model proposed on Paragraph 3.2.1, the second the variation proposed on Paragraph 3.2.2 while last two lines show simple variations with Batch Normalization layers or with double channels for the Conv2D layers on the custom network that extracts features from the Argmax. As we have supposed, the Argmax does not provide extremely useful additional cues, from Table 5.3 it is possible to notice that the best case improves the mIoU of $\sim$ 0.5 % while the worst case obtains the same result as the Baseline.

Table 5.3: Results from 4 different model with Argmax as additional input.

Baseline	**29.9** %

BatchNormalization	Dilation Rate	Mult. Factor	mIoU
-	-	1	29.9 %
-	x	1	30.3 %
-	-	2	**30.4** %
x	-	1	30.2 %



(a) RGB **(b)** GT **(c)** Baseline **(d)** Argmax

(e) RGB **(f)** GT **(g)** Baseline **(h)** Argmax

Figure 5.2: Some Baseline and Argmax model results on Pascal-Part dataset. From left to right, RGB images, Ground-Truth from Pascal-Part, Baseline and Argmax results.

The third and fourth column of Figure 5.2 compare some results from the Baseline and the Argmax model and it is clear that there are not remarkable improvements. The image 5.2h has a more complete contour shape compared to 5.2g , however 5.2d is far from being similar to its ground-truth, it has too many pixels with wrong labels, the body is almost completely labelled as a bird-body(dark-red). The Figures 5.2 underline how this model does not improve to the Baseline.

As it was made on Paragraph 5.2, we have compared the remapped parts with the results from the Pascal-VOC, Table 5.4 reports this comparison. After remapping, the Argmax results shows 65.8 % of mIoU. The bold type values on Table 5.4 underline classes that have an higher IoU compared to the results from the "Standard" Semantic Segmentation task, so in some cases the Part-Based Semantic Segmentation task could obtain better results and improve the metric, however it is notable that the three classes which gain benefits are classes without sub-parts as the *Sofa*, the *Chair* and the *Boat*. Figure 5.3 compares some results remapped with the Pascal-VOC.

Table 5.4: Comparison between results on Pascal-VOC 2010 and results on Pascal-Part after we have merged the results from the Argmax model. The per-class IoU reported below is computed starting from the predictions of the sub-parts.

Class	IoU	Argmax IoU	Class	IoU	Argmax IoU
background	90.71 %	90.28 %	-	-	-
airplane	80.88 %	68.95 %	bus	88.42 %	83.86 %
bicycle	71.81 %	57.80 %	car	85.34 %	77.08 %
bird	85.05 %	73.35 %	cat	89.10 %	82.41 %
boat	46.55 %	**47.2** %	**chair**	41.58 %	**41.7** %
bottle	74.18 %	66.47 %	person	77.04 %	75.75 %
table	54.49 %	45.60 %	potted plant	70.25 %	59.67 %
dog	83.20 %	76.82 %	sheep	81.06 %	71.96 %
horse	75.28 %	61.66 %	**sofa**	39.43 %	**47.59**%
motorbike	66.68 %	53.57 %	train	79.03 %	75.90 %
cow	66.57 %	65.53 %	tv	71.27 %	59.21 %

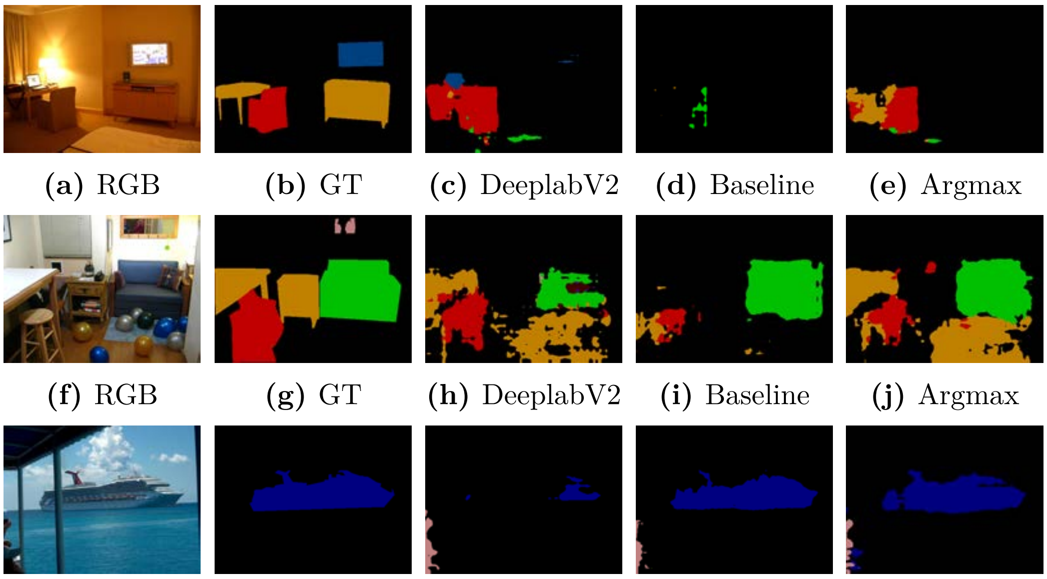
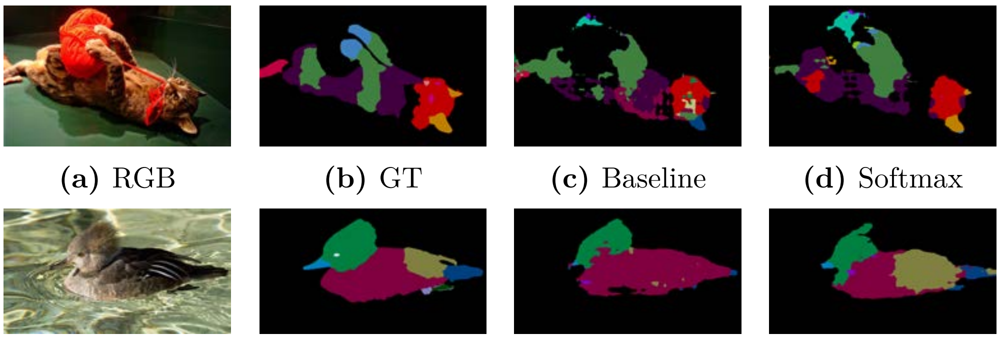

(a) RGB **(b)** GT **(c)** DeeplabV2 **(d)** Baseline **(e)** Argmax
(f) RGB **(g)** GT **(h)** DeeplabV2 **(i)** Baseline **(j)** Argmax
(k) RGB **(l)** GT **(m)** DeeplabV2 **(n)** Baseline **(o)** Argmax

Figure 5.3: Shows some sample images where Sofa (Green), Boat (Blue) and Chair (Red) have an IoU higher with the Argmax model.

5.4 Conditioning with Softmax

From [45] we derived the idea to use the Softmax output to enhance our model, this data should provide better results compared to the Argmax since it brings an higher amount of information. The model proposed on Paragraph 3.2.1 presents an improvement of ~ 3 % with 33.36 % mIoU against the 30.4 % with the Argmax and 29.9 % of our baseline Deeplab v2.. On the third and fourth column of Figure 5.4 the results from the Baseline with the Softmax model are compared. The image 5.4*d* shows some improvements, the body is correctly labelled, pixels predicted as bird-body are gone however there are still wrong labels (turquoise) near the paws. On the other hand, the image 5.4*h* shows a more significant upgrade compared to the Baseline and the Argmax (Figure 5.2*h*), it is quite similar to its ground truth and all its parts have been correctly predicted.


(a) RGB **(b)** GT **(c)** Baseline **(d)** Softmax

(e) RGB **(f)** GT **(g)** Baseline **(h)** Softmax

Figure 5.4: Some Softmax model results on Pascal-Part dataset. From left to right, RGB, Ground-Truth from Pascal-part and Baseline and Softmax results.

On Figure 5.5 we show an histogram with the IoU of each parts for the Baseline and Softmax model. It can be noticed that only few parts have an higher IoU on the Baseline. In both cases some parts are not found as: the *bus-car fliplate*, the *motorbike handlebar* and *saddle*, and the *train headlight*. However these parts are tiny and have few occurrences on the dataset, so probably the network does not have enough examples to learn them. Remapping these results to the 21 classes from the Pascal-VOC leads to a 70.9 % mIoU, still lower compared to the 72.28 % of our baseline model. On Table 5.5 we report IoU for each class, in this case the number of classes with benefits from using additional data is increased. On the previous model, these classes were the ones with no sub-parts, now other classes such as the *Bottle*, *Cow* and the *Bus* have increased their IoU.

Table 5.5: Comparison between results on Pascal-VOC 2010 and results on Pascal-Part after we have merged the results from the Softmax model. The per-class IoU reported below is computed starting from the predictions of the sub-parts.

Class	IoU	Softmax IoU	Class	IoU	Softmax IoU
background	90.71 %	90.62 %	-	-	-
airplane	80.88 %	75.66 %	**bus**	88.42 %	**89.72** %
bicycle	71.81 %	63.26 %	car	85.34 %	81.44 %
bird	85.05 %	78.57 %	cat	89.10 %	85.45 %
boat	46.55 %	**61.77** %	**chair**	41.58 %	**43.36** %
bottle	74.18 %	**74.38** %	person	77.04 %	76.55 %
table	54.49 %	48.64 %	potted plant	70.25 %	61.30 %
dog	83.20 %	83.02 %	sheep	81.06 %	75.58 %
horse	75.28 %	75.00 %	**sofa**	39.43 %	**50.50** %
motorbike	66.68 %	61.61 %	train	79.03 %	77.05 %
cow	66.57 %	**80.45** %	tv	71.27 %	52.42 %

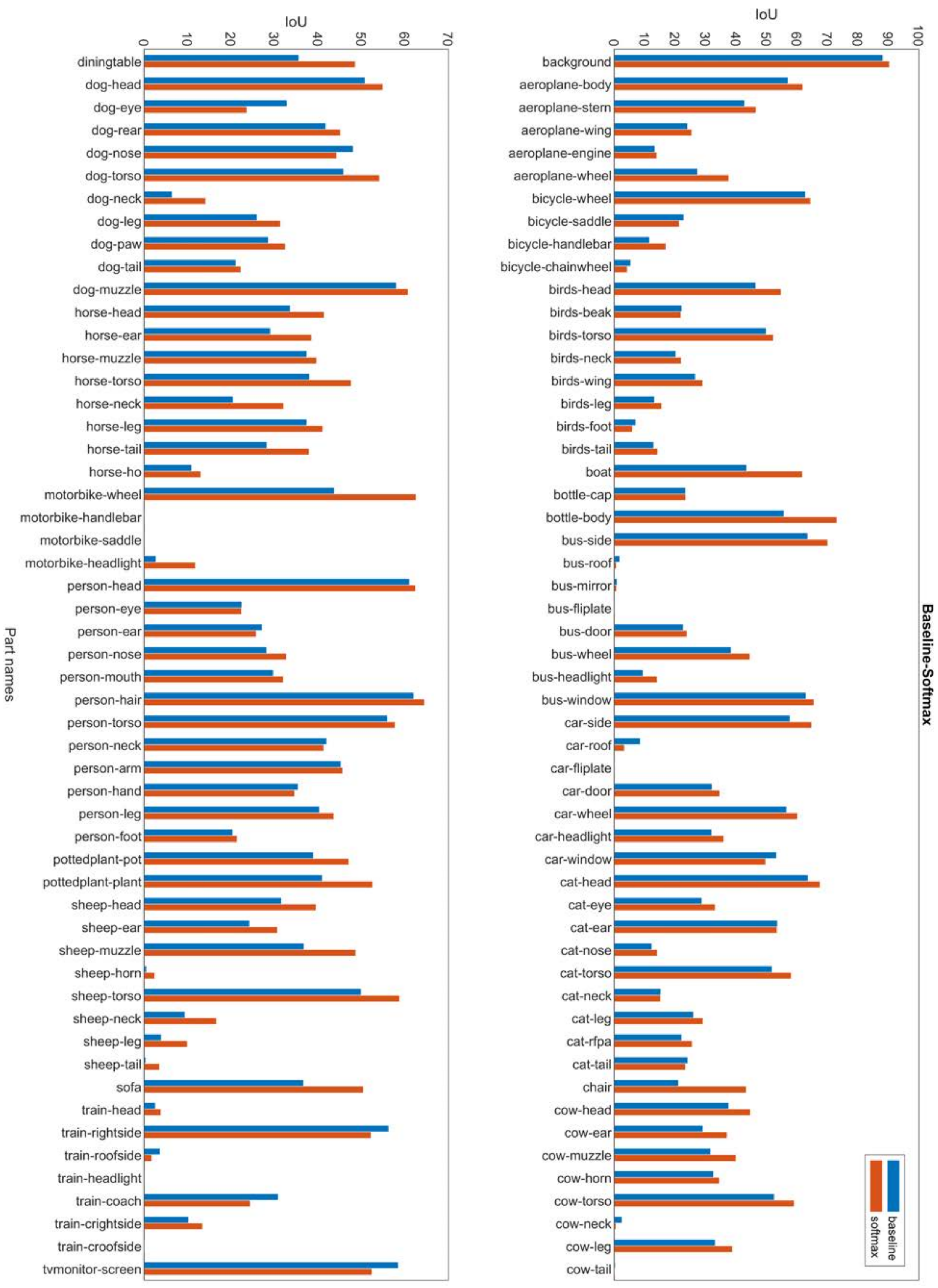


Figure 5.5: Histogram with Baseline and Softmax value for each parts.

5.5 Adjacency Loss

On Paragraph 3.3 we have discussed about a new loss to discriminate parts that should not be close, for example a person-head and a cat-neck. We propose three different variations, a simple one with an unweighted adjacency matrix (i.e., with binary values), a weighted adjacency matrix where each link between two parts is weighted on the basis of how much two parts are close and a third one where we create a different weighted adjacency matrix for each batch image. Finally we trained our model end-to-end with a new loss given by Eq. [5.6], which is the linear combination of an L2 adjacency loss as described on Eq. [3.2] and the *categorical crossentropy* loss.

$$NewLoss = CategoricalCrossentropy + \lambda * AdjacencyLoss \tag{5.6}$$

The term *AdjacencyLoss* can be one of the three cases describe above and the λ term is an hyper-parameter that must be set. Empirically,, we set $\lambda = 10^{-2}$. Another parameter that must be set is related to the weighted Adjacency matrix. Studying the Pascal-Part dataset we noticed how different parts of the same class that should be close are instead divided by background pixels, as it is shown in Figure 5.6. For this reason we modify the algorithm described on Paragraph 3.3.1 and set the distance between contour points as a variable, so two parts will be consider close if their contour distance is $\leq pix$, where pix is the distance variable. Even for this case we made some empirical trials with different distance value and in the end we set $pix = 3$ pixels.

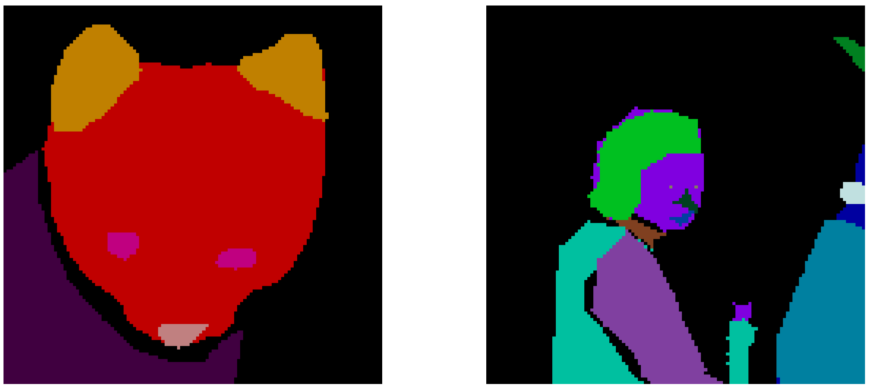


Figure 5.6: Two examples where related parts that should be attached are instead divided by background pixels.

All these losses are applied on the model that takes as input RGB + Softmax, since we have proved that the Argmax does not provide enough additional information to our purpose. Table 5.6 reports the results obtained from the losses proposed. The first line of the table shows that the unweighted Adj-matrix proposed performs better of the softmax model, it gains $\sim$ 1%, from 33.36% to 34.26%. The second and third line with the weighted Adj-matrix with 1 pixel distance and the one that calculates a different matrix for each batch image do not show considerable improvements. However the weighted Adj-matrix with pixel distance set to 3 performs better compared to the softmax model of $\sim$ 2%.

Table 5.6: From Top to Bottom: the unweighted adjacency-matrix loss, the weighted adjacency-metrix loss with 1 and 3 pixel distance, the weighted adjacency-matrix loss which calculates a different adjacency matrix for each batch image($*$), in this case 2 matrix for a batch size 2.

Baseline		**29.9** %
Loss	Pixel Distance	mIoU
Unweighted Adj-matrix	-	34.26 %
Weighted Adj-matrix	1	34.36 %
Weighted Adj-matrix	3	**35.1** %
2 Weighted Adj-matrix($*$)	1	34.76 %

The Weighted Adjacency matrix loss, with pixel distance set to 3, obtains the best results. From Figure 5.7 it is possible to notice the relevant improvements compared to the previous models. The dog on image 5.7u does not have pixels with labels from other classes and moreover it has a more defined shape, in fact its muzzle is correctly found even if its ground-truth is partially wrong. On Figure 5.7e the dog holds a toy with its mouth and it is labelled as part of the muzzle.
The train, 5.7v, and the cow, 5.7x, do not show relevant improvements compared to the Argmax instead the cat, 5.7w, is the one that has gained more benefits from this loss compared to other model results.
Figure 5.8 reports the Baseline and Adjacency model results for the 108 parts, we can notice that parts that were not found or have a small IoU on Figure 5.5 are not improved from this approach. This supports the previous hypothesis according to which the network could not recognize them because are too tiny and with few occurrences.

Additionally, on Figure 5.8 we can confirm the wide and general trend of improvement brought by our final framework with the novel loss function with respect to the baseline model.
Furthermore this model after merging the resulting parts with its 72.27 of % mIoU is able to match the 72.28 % obtained with the DeeplabV2 on Pascal-VOC and Table 5.7 reports the IoU for each class. On Figure 5.9 we compare all the parts remapped from the previous model with the DeeplabV2 results from the Pascal-VOC.

It is clear how our approach gradually enhances the results and obtains an equivalent and in some case better results compared to the DeeplabV2. For example on the third column of Figure 5.9, the DeeplabV2 finds some wrong pixels labelled as a person (pink). These pixels are there also in the other models results, together with other wrong pixels tagged as bird (light brown), however with our Adjacency loss wrong labels are drastically reduced and the image is clearly better. Also the dog in the last row does not show some pink pixels near the right paw.

Table 5.7: Comparison between results on Pascal-VOC 2010 and results on Pascal-Part after we have merged the results from the Adjacency loss model. The per-class IoU reported below is computed starting from the predictions of the sub-parts.

Class	IoU	Adjacency IoU	Class	IoU	Softmax IoU
background	90.71 %	**91.02** %	-	-	-
airplane	80.88 %	75.28 %	**bus**	88.42 %	**90.07** %
bicycle	71.81 %	63.32 %	car	85.34 %	82.03 %
bird	85.05 %	80.56 %	cat	89.10 %	87.16 %
boat	46.55 %	**61.08** %	**chair**	41.58 %	**44.63**%
bottle	74.18 %	**74.96** %	**person**	77.04 %	**77.84** %
table	54.49 %	49.76 %	potted plant	70.25 %	58.58 %
dog	83.20 %	**83.90** %	sheep	81.06 %	78.67 %
horse	75.28 %	**76.24** %	**sofa**	39.43 %	**51.91** %
motorbike	66.68 %	61.51 %	train	79.03 %	78.13 %
cow	66.57 %	**82.31** %	tv	71.27 %	68.68 %

Here we show some Tables that recap the results. Tables 5.8 reports the model results organized in ascending order from the Baseline to the Adjacency loss model. In this case in addition to the mIoU we compare the mPA, Eq. [5.4], and the mCA, Eq. [5.5]. Table 5.9 reports the same metrics but for the remapped results.
These metrics show how our proposed models gradually enhance the mIoU as well as the mPA and the mCA, showing coherence between the results of the Part-Based Semantic Segmentation Task and the Standard Semantic Segmentation after remapping.
Moreover, looking to the last line of Table 5.9, it possible to notice how the mIoU of our approach matches the results obtained with the DeeplabV2 but exceeds it with the mPA of ∼0.5% and with the mCA of ∼4%, showing the robustness of our approach.

Table 5.8: Results in ascending order. From Top to Bottom: mean Intersection over Union, mean Pixel Accuracy and mean per Class Accuracy for each model.

Model	mIoU	mPA	mCA
Baseline	29.9%	84.92%	39.5%
Best Argmax	30.4%	86.06%	41.25%
Best Softmax	33.3%	87%	44.13%
Best Adj-matrix	35.1%	87.45%	45.92%

Table 5.9: Results after remapping the parts in ascending order. From Top to Bottom: mean Intersection over Union, mean Pixel Accuracy and mean Class Accuracy for each model.

DeeplabV2	72.28%	91.95%	81.25%

Model-ReMapped	mIoU	mPA	mCA
Baseline	60.2%	89.5%	73.48%
Best Argmax	65.8%	91.09%	81%
Best Softmax	70.9%	92%	83.25%
Best Adj-matrix	72.27%	92.45%	85.26%

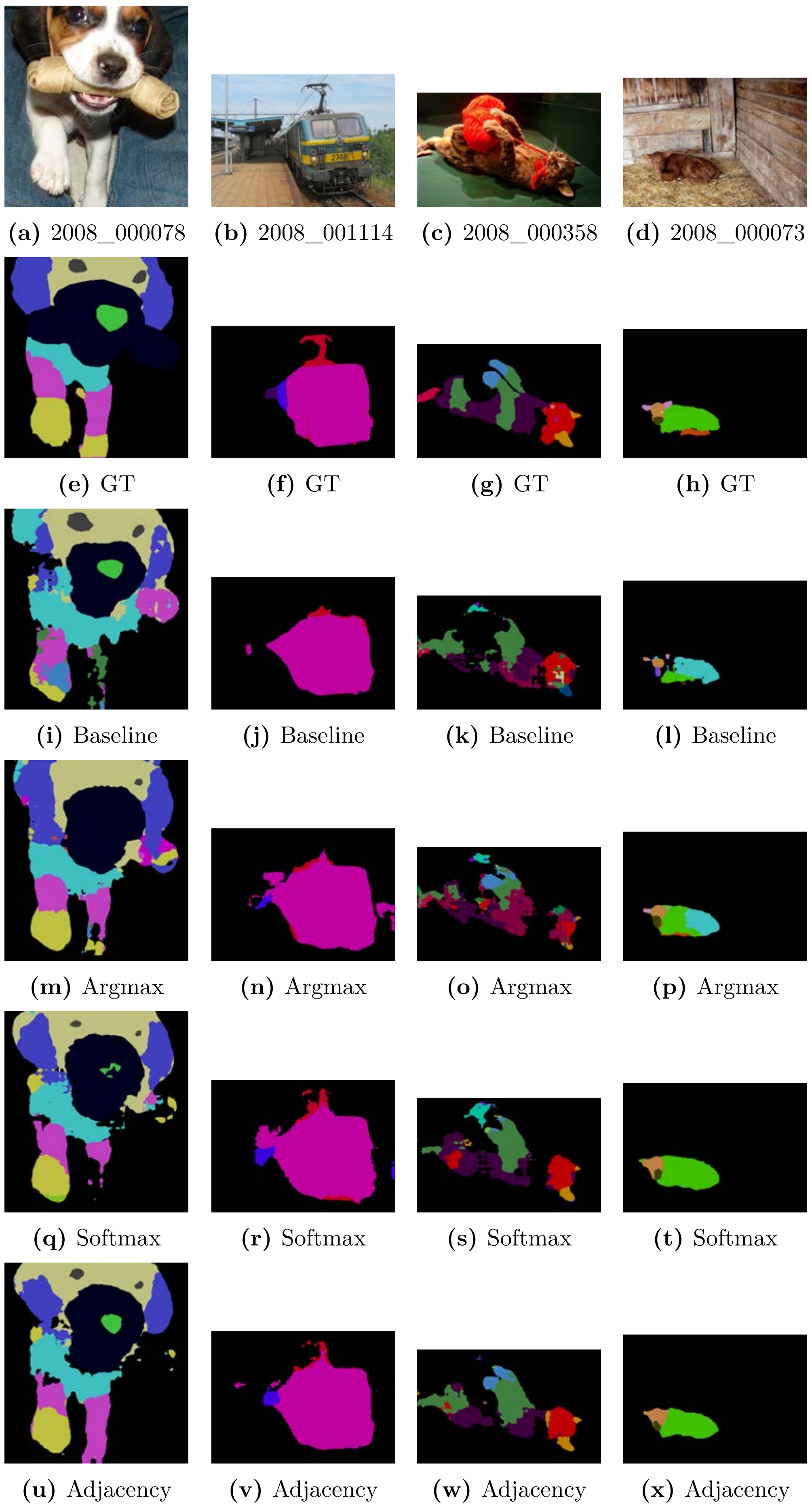


Figure 5.7: Results comparison between all models. From Top to Bottom: RGB, Ground Truth, Baseline, Argmax, Softmax and Adjacency Model.

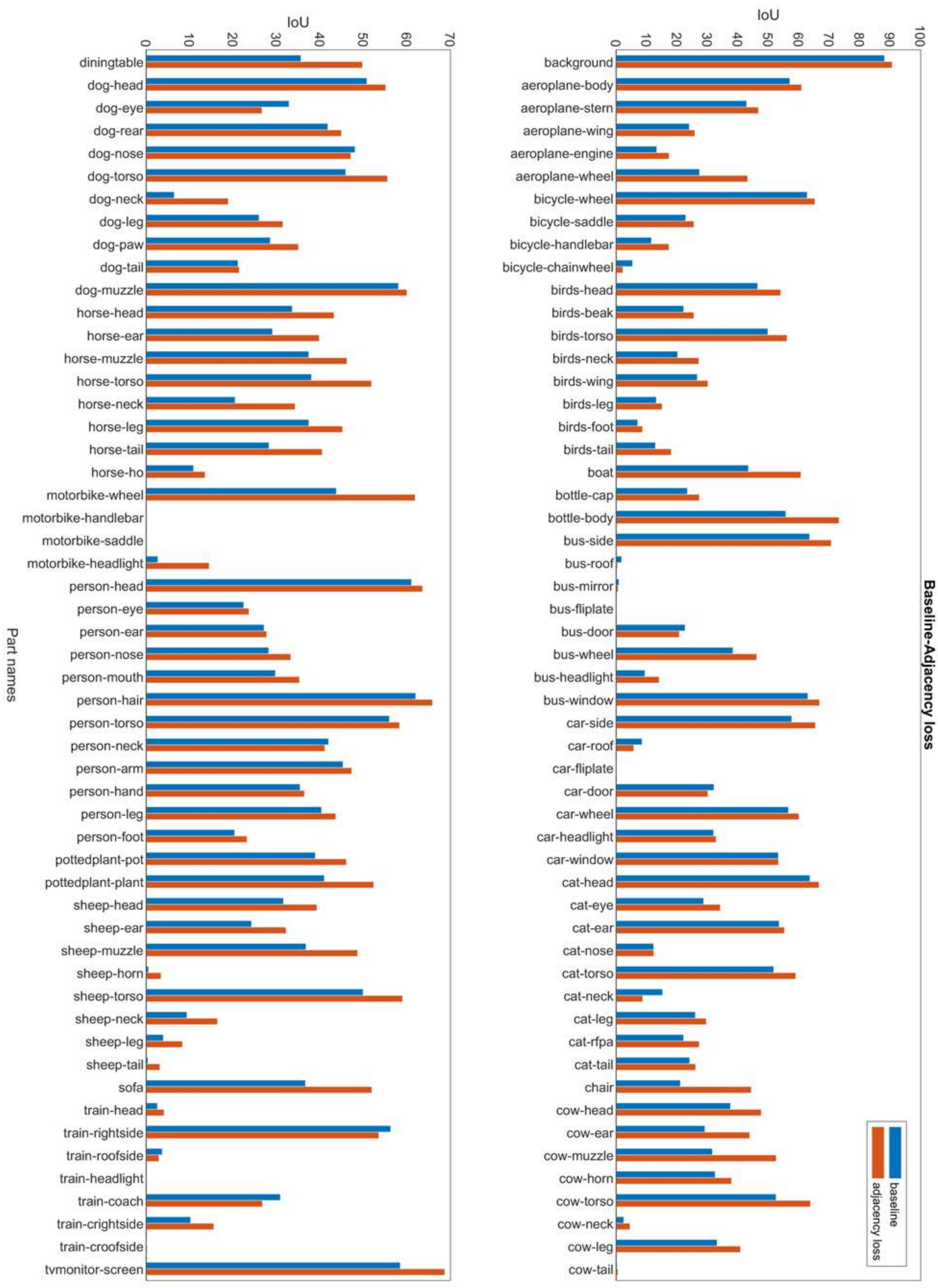
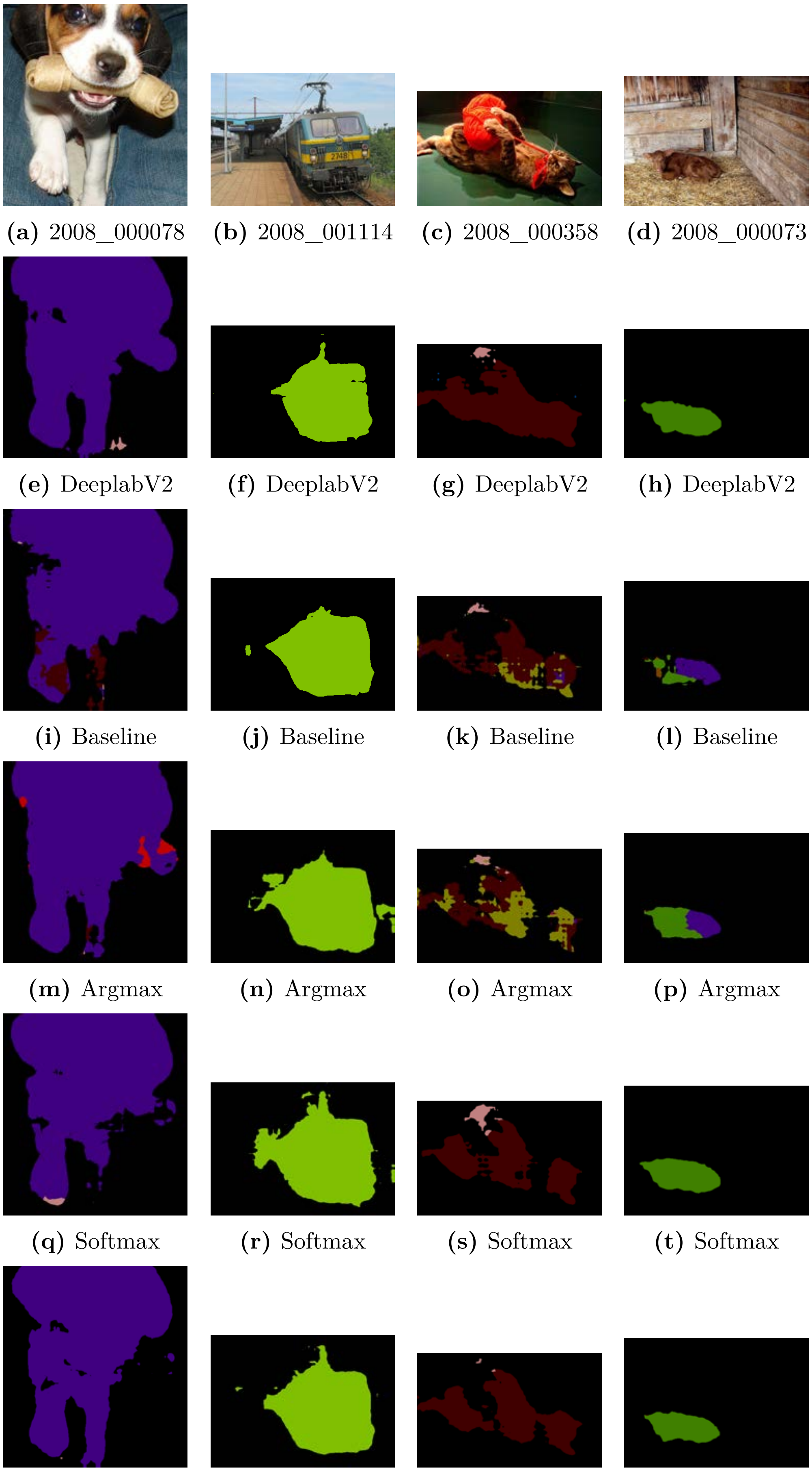


Figure 5.8: Histogram with Baseline and Adjacency model values for each parts.



(a) 2008_000078 (b) 2008_001114 (c) 2008_000358 (d) 2008_000073

(e) DeeplabV2 (f) DeeplabV2 (g) DeeplabV2 (h) DeeplabV2

(i) Baseline (j) Baseline (k) Baseline (l) Baseline

(m) Argmax (n) Argmax (o) Argmax (p) Argmax

(q) Softmax (r) Softmax (s) Softmax (t) Softmax

(u) Adjacency (v) Adjacency (w) Adjacency (x) Adjacency

Figure 5.9: Results remapped compared with all models. From Top to Bottom: RGB, DeeplabV2, Baseline, Argmax, Softmax and Adjacency Model.

6 Conclusions and future work

In this work we have investigated the Part-Based Semantic Segmentation task with the aim of improving the performance when the set of parts to be learnt is not small. As a matter of fact, many works have studied this topic but most of these involve a restricted set of parts belonging to just few classes such as human, animals or cars.

This was made possible by using a framework with two networks: the first was trained in the Standard Semantic Segmentation task with 21 classes while the second one had the same architecture of the first but it was adapted to predict a finer set of 108 parts derived from the initial segmentation.
These networks can be considered as a cascade, the second network is fed with the output of the first plus the RGB images. This architecture allows to the second one to extract features from the softmax (or argmax) outputs of the first network, in addition to the features from the RGB images. Features from the previous output are useful to constrain and condition the prediction of the parts.

Furthermore, we have studied a new loss called Adjacency loss that is inspired by the Adjacency-matrix from the Graph theory. We built an adjacency matrix for each predicted image and its ground-truth and we applied a distance function, in such a way we minimize the discrepancies between them. This should teach to the network how to penalize parts that should not be close.

Experimental results have demonstrated the effectiveness of this approach. A State-of-the-Art network for Semantic Segmentation trained with 108 parts obtains 29.9% of mIoU, while our model is able to reach 30.5% and 33.3% of mIoU respectively with the Argmax and the Softmax output. Moreover applying our new loss we have reached 35.1% of mIoU with a visible improvement on predicted images. Additionally, remapping the parts to their respective parent, we were able to reach the mIoU obtained by the first network with the 21 classes. This proves that not only we improved the Part-Base Semantic Segmentation task but we also do not lose information for the Standard Semantic Segmentation task and moreover we could improve it.

Finally, further research could be devoted to improve the Adjacency loss: some classes are too small and have few occurrences in the dataset and it could be useful to find a way to weight each part differently. For example, it could be possible to weight each part by considering its frequency in the entire dataset or for single image during training. Additionally, it could be possible to study a new loss which not only consider the adjacent parts but also their spacial location; it will consider if a head and a torso person are close as well as if they are correctly located. This new loss combined with our Adjacency loss and model could further improve the performances.

Bibliography

[1] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner. “Gradient-based learning applied to document recognition” . In: *Proceedings of the IEEE*, Volume 86, Issue 11, Nov. (1998), pp. 2278-2324.

[2] A. Krizhevsky, I. Sutskever, and Georey E. Hinton. “Imagenet classication with deep convolutional neural networks”. In: *Advances in neural information processing systems.* (2012), pp. 1097-1105.

[3] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez and Jose Garcia-Rodriguez. “A review on deep learning techniques applied to semantic segmentation”. In: *arXiv preprint arXiv:1704.06857*, (2017).

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun. “Deep Residual Learning for Image Recognition”. In:*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* (2016), pp. 770-778.

[5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich. “Going deeper with convolutions”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* (2015), pp. 1-9.

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

[7] M. Everingham, Luc Van Gool, C. K. I. Williams, J. Winn and A. Zisserman. “The Pascal Visual Object Classes (VOC) challenge”. In: *International journal of Computer Vision.* Volume 88, Numbers 2, Jun. (2010), pp. 303-338.

[8] X. Chen, R. Mottaghi, X. Liu, S. Fidler, R. Urtasun and A. Yuille. “ Detect what you can: Detecting and representing objects using holistic models and body parts”. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* (2014).

[9] A. Ahmed, K. Yu, W. Xu, Y. Gong, and E. Xing, “Training hierarchical feed-forward visual recognition models using transfer learning from pseudo-tasks”. In *European Conference on Computer Vision (ECCV)*. Springer, (2008), pp. 69–82.

[10] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, “Learning and transferring mid-level image representations using convolutional neural networks.” In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*. (2014), pp. 1717–1724.

[11] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?”. In *Advances in neural information processing systems*. (2014), pp. 3320–3328.

[12] J. Long, E. Shelhamer and T. Darrell. “Fully convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*. (2015), pp. 3431-3440.

[13] V. Badrinarayanan, A. Kendall and R. Cipolla. “Segnet: A deep convolutional encoder-decoder architecture for image segmentation”. In: *IEEE transactions on pattern analysis and machine intelligence (TPAMI)* . Volume 39, Issue: 12 , Dec. (2017), pp. 2481-2495.

[14] O. Ronneberger, P. Fischer and T. Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *International Conference on Medical image computing and computer-assisted intervention (MICCAI)*. Springer, (2015), pp. 234-241.

[15] Tsung-Yi Lin et al. “Feature pyramid networks for object detection”. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 1. 2. 2017, p. 4.

[16] Hengshuang Zhao et al. “Pyramid scene parsing network”. In: *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. (2017), pp. 2881-2890.

[17] Liang-Chieh Chen et al. “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs”. In: *IEEE transactions on pattern analysis and machine intelligence (TPAMI)* 40.4 (2018), pp. 834-848.

[18] Philipp Krähenbühl and Vladlen Koltun. "Efficient inference in fully connected CRFs with gaussian edge potentials". In: *Advances in neural information processing systems.* (2011), pp. 109-117.

[19] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. "Semantic image segmentation with deep convolutional nets and fully connected crfs". *arXiv:1412.7062*, 2014.

[20] F. Yu and V. Koltun. "Multi-scale context aggregation by dilated convolutions". *arXiv:1511.07122*, 2015.

[21] Y. Yang and D. Ramanan. "Articulated pose estimation with flexible mixtures of parts". In: *Proceedings of the IEEE International Conference on Computer Vision (CVPR).* (2011), pp. 1385–1392.

[22] J. Dong, Q. Chen, X. Shen, J. Yang, and S. Yan. "Towards unified human parsing and pose estimation". In: *Proceedings of the IEEE International Conference on Computer Vision (CVPR).* (2014), pp. 843–850.

[23] H. Azizpour and I. Laptev. "Object detection using strongly-supervised deformable part models". In: *European conference on computer vision (ECCV).* (2012), pp. 836–849.

[24] X. Chen, R. Mottaghi, X. Liu, S. Fidler, R. Urtasun, and A. L. Yuille. "Detect what you can: Detecting and representing objects using holistic models and body parts". In: *Proceedings of the IEEE International Conference on Computer Vision (CVPR).* (2014), pp. 1979–1986.

[25] N. Zhang, J. Donahue, R. B. Girshick, and T. Darrell. "Part-based r-cnns for fine-grained category detection". In: *European conference on computer vision (ECCV).* (2014), pp. 834–849.

[26] L. Zhu, Y. Chen, C. Lin, and A. L. Yuille. "Max margin learning of hierarchical configural deformable templates (hcdts) for efficient object parsing and pose estimation". In: *International Journal of Computer Vision (IJCV).* May 2011, Volume 93, Issue 1, pp 1–21.

[27] S. M. A. Eslami and C. K. I. Williams. "A generative model for parts-based object segmentation". In: *Neural Information Processing Systems (NIPS).* (2012), pp. 100–107.

[28] J. Dong, Q. Chen, W. Xia, Z. Huang, and S. Yan. “A deformable mixture parsing model with parselets”. In: *Proceedings of the IEEE International Conference on Computer Vision (CVPR).* (2013), pp. 3408–3415..

[29] P. Wang, X. Shen, Z. Lin, S. Cohen, B. Price, and A. L. Yuille. “ Joint object and part segmentation using deep learned potentials”. In: *Proceedings of the IEEE International Conference on Computer Vision (CVPR).* (2015), pp. 1573–1581.

[30] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille. “Attention to scale: Scale-aware semantic image segmentation”. *arXiv:1511.03339.* (2015).

[31] F. Xia, P. Wang, L.-C. Chen, and A. L. Yuille. “ Zoom better to see clearer: Human and object parsing with hierarchical auto-zoom net”. In: *European conference on computer vision (ECCV).* (2016).

[32] K. Yamaguchi, M. H. Kiapour, L. E. Ortiz, and T. L. Berg. “ Parsing clothing in fashion photographs”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* (2012).

[33] L. Ladicky, P. H. Torr, and A. Zisserman. “Human pose estimation using a joint pixel-wise and part-wise formulation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* (2013), pp. 3578–3585.

[34] F. Xia, P. Wang, X. Chen and A. L. Yuille. “Joint Multi-Person Pose Estimation and Semantic Part Segmentation”. *arXiv:1708.03383.*

[35] V. Dumoulin and F. Visin. “A guide to convolution arithmetic for deep learning”. *arXiv:1603.07285.*

[36] T. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. Lawrence Zitnick and P. Dollár. “ Microsoft coco: Common objects in context” . In: *European conference on computer vision (ECCV).* Springer, (2014), pp. 740-755.

[37] Abel Gonzalez-Garcia1, D. Modolo1, V. Ferrari.“ Received: Do Semantic Parts Emerge in Convolutional Neural Networks?”. In: *International Journal of Computer Vision.* Volume 126, Issue 5, May (2018), pp 476–494.

[38] T. Xiao, J. Zhang, K. Yang, Y. Peng, and Z. Zhang. “ Error-driven incremental learning in deep convolutional neural network for large-scale image classification. ” In: *Proceedings of the 22nd ACM International Conference on Multimedia (ACMMM).* ACM, 2014, pp. 177–186.

[39] D. Roy, P. Panda, and K. Roy. “Tree-CNN: a hierarchical deep convolutional neural network for incremental learning.” *arXiv preprint arXiv:1802.05800*, 2018

[40] R. Istrate, A. C. I. Malossi, C. Bekas, and D. Nikolopoulos. “Incremental training of deep convolutional neural networks.” *arXiv preprint arXiv:1803.10232*, 2018.

[41] S. S. Sarwar, A. Ankit, and K. Roy. “Incremental learning in deep convolutional neural networks using partial network sharing.” *arXiv preprint arXiv:1712.02719*, 2017.

[42] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. “Overcoming catastrophic forgetting in neural networks.” In: *Proc. of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[43] U. Michieli and P. Zanuttigh. “Incremental Learning Techniques for Semantic Segmentation.” In: *Proc. of IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[44] U. Michieli and P. Zanuttigh. “ Distillation Techniques for Incremental Learning on Semantic Segmentation.” 2019.

[45] Michieli U., Camporese M., Agiollo A., Pagnutti G., Zanuttigh P., “ Region Merging Driven by Deep Learning for RGB-D Segmentation and Labeling ”. In: *Proceedings of the International Conference on Distributed Smart Cameras (ICDSC).* Trento (Italy), 9-11 September 2019.

[46] G. Hinton, O. Vinyals, and J. Dean. “ Distilling the knowledge in a neural network.” In: *Neural Information Processing Systems (NIPS) Deep Learning and Representation Learning Workshop*, 2015.

[47] C. Bucilua, R. Caruana, and A. Niculescu-Mizil. “ Model compression.” In: *Proceedings of the 12th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD).* ACM, 2006, pp. 535–541.

[48] M. Abadi et al. “TensorFlow: A system for large-scale machine learning.” In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16).* pp. 265–283, 2016.

[49] G. Pagnutti, L. Minto and P. Zanuttigh. “Segmentation and Semantic Labeling of RGBD Data with Convolutional Neural Networks and Surface Fitting.” In: *IET Computer Vision*, Volume: 11 , Issue: 8 , 2017.

[50] G. Csurka, D. Larlus, F. Perronnin. “What is a good evaluation measure for semantic segmentation?”. In: *British Machine Vision Conference (BMVC)* . January 2013.