OBJECT RECOGNITION

FASHION SEMANTIC SEGMENTATION

ACADEMIC YEAR: 2021/2022

AUTHOR: JOAQUIM MARSET ALSINA

1 Introduction

The present report describes the results of the semantic segmentation task applied to the fashion dataset *Fashionpedia* [1]. In section 2 we briefly explain the dataset showing some statistics. In section 3 we explain the selected algorithm, the different experiments we have performed, and numeric results with the best model found. In Appendix A, we break down the numeric results of the best model for each class. In Appendix B, we present some qualitative results using our best model.

2 Dataset

Fashionpedia is a dataset containing clothing images, with different annotations that make it usable to other tasks besides semantic segmentation. It contains 45623 training images and 3200 test images (1158 with segmentation), with sizes ranging up to 1024x1024. For the semantic segmentation task we have forty-six classes, including different clothing pieces (e.g. a shirt or a short), accessories (e.g. a scarf), or even parts of another piece (e.g. the pocket of a trouser). In Figure 1, we present the histogram of the class frequency in the training dataset.

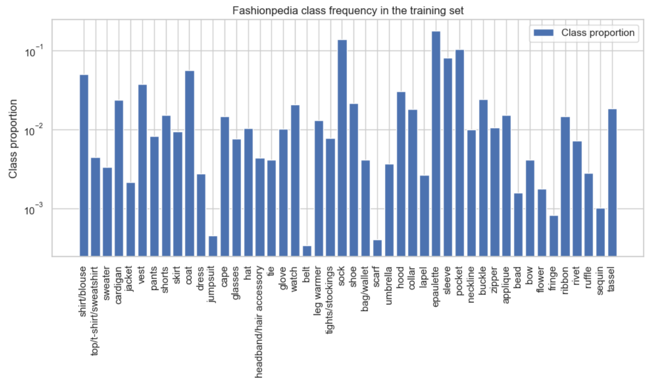

Figure 1: Training set classes proportion

3 Model and Training

To train our model we have used the semantic segmentation *MMSegmentation* [2] *Python* library, using *Pytorch* at its core. The training and evaluation are performed automatically, and the user only needs to facilitate a configuration file stating the model to use, the training schedule, the data to use and its preprocessing steps.

For this reason, we decided to use one of the already implemented models there. In particular, we decided to use the Swin Transformer as the backbone, UPerNet as the decoder head, and an auxiliar fully convolutional head. One of the reasons why we decided to use this combination is because *MMSegmentation* has some pre-trained models, and the only one combined with Swin was this one.

We decided to use Swin in the first place because it was listed in the ranking as achieving good results. The one listed was the v2 version, but the only configuration available in *MMSegmentation*, used in the official implementation [3], is the first version. Another reason for using Transformers is because they are causing quite the uproar, and having the chance to use them is engaging. Also, I am doing my Master's Thesis about vision Transformers, so I found it a good opportunity to learn and practise better with them.

3.1 Model Architecture

The Swin Transformer (Shifted Window Transformer) [4] is a vision Transformer that can be used as a general-purpose backbone. It is a hierarchical Transformer that uses patches of different resolutions at the different

[1]`https://fashionpedia.github.io/home/Fashionpedia_download.html`
[2]`https://github.com/open-mmlab/mmsegmentation`
[3]`https://github.com/SwinTransformer/Swin-Transformer-Semantic-Segmentation`
[4]`https://arxiv.org/pdf/2103.14030.pdf`

Transformer layers. It also computes the self-attention to only a limited window of neighbour patches, achieving a linear complexity. At each layer, windows of patches are fed through a self-attention block, outputting embeddings merged later. The next layer repeats the process but shifting the attention window w.r.t the previous layer (like stride convolutions), allowing communication between previously uncommunicated patches.

UPerNet (UnifiedPerceptual Parsing Network) [5] was developed to extract the objects of an image together with semantic information from those objects (e.g. their parts, textures, and materials). It presents a complex hierarchical architecture based on Feature Pyramid Network (FPN) and Residual Networks. In the paper, the authors argue that this new model can avoid the drawbacks of dilated convolution that make them unfeasible to segment those perceptual attributes. Also, the model is able to unify tasks that are usually performed independently as no dataset is annotated with all that information.

Combining both Swin and UPerNet is a good idea, as UPerNet needs to apply a Pyramid Pooling Module (PPM) [6] before feeding the input to the UPerNet, to account for the empirical small receptive fields of the deep CNN used as backbone. With the self-attention mechanism of the Transformers we could avoid this step, as they can capture longer-range interactions. We also think this architecture could be useful in extracting the object's attributes *Fashionpedia* has available in the other tasks.

3.2 Experiments

First, we have to mention that some experiments were impossible to run with our resources, and we ended up renting a high-spec remote machine to perform some of them. Also, those we could run in our machine last at least one day each, so we could not try as much as we wanted to. For these reasons, each experiment is performed during twenty epochs, evaluating the `mIoU` score (average Intersection over Union of all classes).

In particular, we have mainly changed the Swin configuration and the image preprocessing steps (i.e. image size, normalization and random cropping). We tried to see the effect of data augmentation (e.g. rotations and pixel distortions), but we did not observe significant changes. We decided to keep it in the experiments hoping to have a positive effect. We also tried to see the effect of the optimizer, but all the *MMSegmentation* Swin configurations [7], use `Adam` with weight decay and the same parameters (i.e. lr=6e-5, decay=0.01). We tried changing its parameters once without successful results, so in this case, we decided to keep it the same as the existing configurations.

The experiments are listed in Table 1. More experiments trying more combinations would be desirable, but it was not possible with our resources. In figure Figure 2 a plot comparing the validation performance is shown:

Experiment	Model	Optimizer Parameters	Image Size	Image Normalization	Augmentation	Random Cropping
1	Swin-T (smallest)	Unchanged	224x224 (pretrained model image size)	Scale by 255	No	No
2	Swin-T	Unchanged	224x224	*Imagenet* mean and std	Yes	No
3	Swin-T	Unchanged	512x512	*Imagenet* mean and std	Yes	256x256
4	Swin-B (biggest)	Unchanged	224x224	*Imagenet* mean and std	Yes	No
5	Swin-S (medium)	Unchanged	512x512	*Imagenet* mean and std	Yes	256x256
6	Swin-S	lr=1e-4, decay=0.05	512x512	*Imagenet* mean and std	Yes	256x256
7	Swin-S	Unchanged	1024x1024 (biggest size in the data)	*Imagenet* mean and std	Yes	512x512 (model expected crop size [8])

Table 1: Experiments

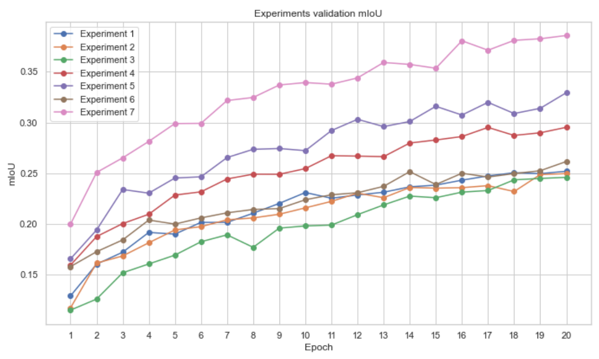

Figure 2: Experiments validation *mIoU* for 20 epochs

[5] `https://arxiv.org/pdf/1807.10221v1.pdf`
[6] `https://arxiv.org/pdf/1612.01105.pdf`
[7] `https://github.com/open-mmlab/mmsegmentation/tree/master/configs/swin`

As we can see, the last experiment is the one achieving best `mIoU` in all epochs. We extract from the plot that the most important factor is the image size previous to random cropping. Indeed, we can generate crops containing classes that take a small region in the original image, improving their prediction. As we will see in the next section, there are classes we cannot predict whose proportion is not that small (e.g. leg warmer), but we suspect their information is lost because other objects have more weight in the embedding. We can see in the first two experiments what we have said about the data augmentation not having a significant improvement. We can also see in the fifth and sixth experiments what we have said about worsening the results when changing the optimizer parameters. Finally, comparing the second and fourth experiment indicates the model size affects, but we could not explore that theory that much as we could not train a lot with the biggest model (because of the lack of resources).

3.3 Final Results

In this section, we show some numeric results obtained with the model of the best experiment, now trained during forty epochs. The validation performance is given in Figure 3 in terms of `mIoU` and `mDice` (average of the Dice score of all the classes), the most used metrics in semantic segmentation. In Appendix A, we show a table with the metrics for each class.

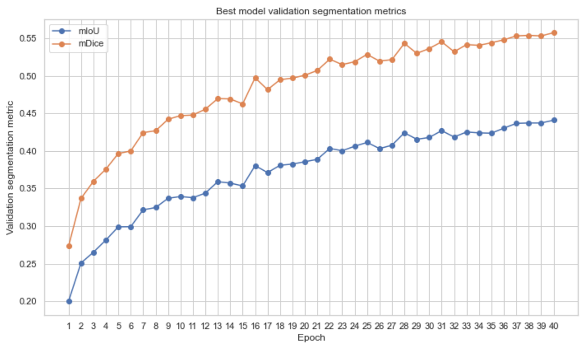

Figure 3: Final model validation `mIoU` and `mDice` for 40 epochs

As we can see, we obtain a final `mIoU` of **44.11** and `mDice` of **55.77**, which is quite acceptable. It is true that as the `mIoU` reflects, we are not able to predict half of the pixels of the different validation images. Nevertheless, if we look at the results of the pre-trained model for the ADE20k dataset, the `mIoU` for the Swin-S was around 49%, so we are not that far away from that value. However, as we can see in the plot, in the last epochs the improvement starts to saturate, so with more epochs, it could still improve or not.

If we look at Table 2, we can see what we have said before that there are some classes that we cannot predict very well. This can also be observed in Figure 4, as the model is not very good in predicting certain classes, mainly accessories and parts of other clothe pieces like bows, ribbons, pockets and collars, as they are not very well segmented. We also have some cases like the penultimate image, where we cannot predict the ruffle even though its IoU is not that low, probably because of the very similar color of both pieces. The same happens in the last image, as we predict the jumpsuit as separate parts. In some cases we even have classes with zero IoU, meaning that the model cannot predict a single pixel. Nonetheless, we can see cases where the model seems better than the ground truth, like the segmented shoes in the seventh image. Also, we can see that with the bigger and more common objects the model does not suffer that much.

4 Conclusions

In this assignment, we have trained and end-to-end semantic segmentation model with a relatively large dataset of big images. The results are decent, but we cannot predict at all three of the forty-six classes, and some others have an almost zero IoU. We believe we can obtain better performance with more epochs, maybe with the biggest model, and extracting crops of even smaller regions. Maybe other augmentation techniques can also help (e.g. over-sampling). In any case, we have to mention the problem of not having available resources, that has hindered a lot the task, and we have not been able to experiment to our heart content and extract more conclusions.

A Class Numeric Results

Class	IoU	Dice
background	98.85	99.42
shirt/blouse	64.51	78.42
top/t-shirt/sweatshirt	76.22	86.51
sweater	64.88	78.7
cardigan	38.82	55.93
jacket	61.37	76.06
vest	44.07	61.18
pants	79.44	88.54
shorts	64.35	78.31
skirt	60.34	75.27
coat	60.66	75.51
dress	79.94	88.85
jumpsuit	35.48	52.38
cape	60.53	75.41
glasses	77.62	87.4
hat	79.78	88.75
headband/head covering/hair accessory	45.66	62.69
tie	30.27	46.47
glove	53.33	69.56
watch	32.3	48.83
belt	42.11	59.26
leg warmer	0.0	0.0
tights/stockings	68.1	81.02
sock	19.36	32.44
shoe	72.29	83.92
bag/wallet	72.18	83.84
scarf	55.48	71.36
umbrella	88.12	93.68
hood	32.04	48.53
collar	38.48	55.58
lapel	54.02	70.15
epaulette	4.89	9.32
sleeve	79.9	88.83
pocket	31.3	47.68
neckline	39.79	56.93
buckle	16.24	27.95
zipper	21.42	35.28
applique	24.68	39.58
bead	12.04	21.5
bow	0.0	0.0
flower	0.71	1.42
fringe	32.95	49.57
ribbon	1.41	2.79
rivet	0.16	0.31
ruffle	40.32	57.47
sequin	16.64	28.54
tassel	0.0	0.0

Table 2: Class validation metrics

B Qualitative Results

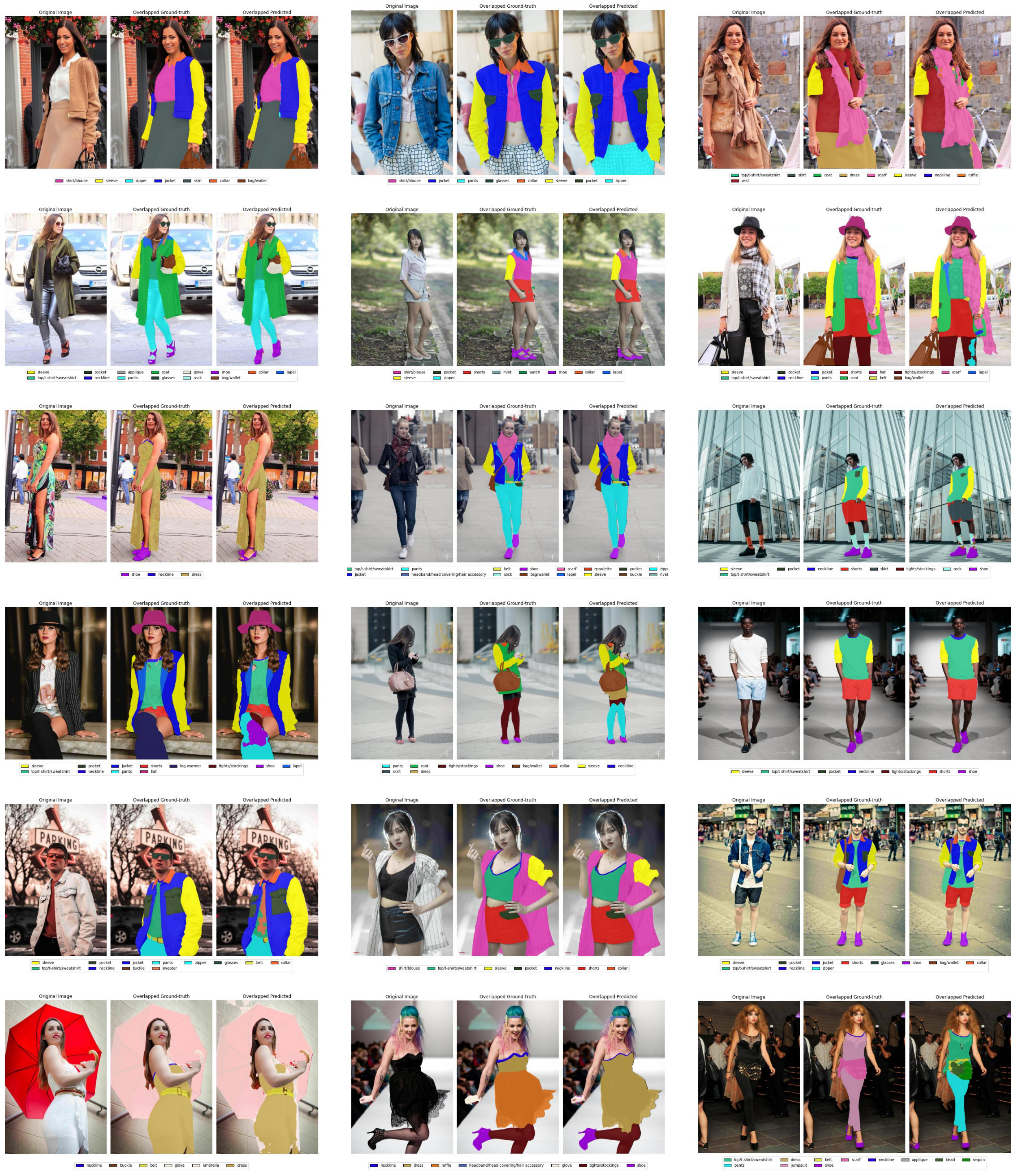

Figure 4: Comparison of ground-truth and predicted segmentations for some validation images