U-NET model for Carvana dataset image segmentation

Design, training and experimentation

Vicent Cano Caravaca

Tuesday 4th January, 2022

1 Introduction

This memory will document the creation of a model to segment cars from their background, which will always be the same. This projects is inspired in the kaggle competition "Carvana Image Masking Challenge".

One of the most difficult challenges that have had been faced is the technical limitations derived from experimenting with neural networks with a laptop. This has made necessary to use the cloud service "Google Collab", however, due to the limitations of the free version, the time dedicated to training has had to be reduced. This impacted directly the number of experimental configurations, and the number of epochs each one was given to train.

2 Research and selection of approach

The first step in this project was to choose the model to use. At the start, the hypothesis was that picking a pretrained model and retraining it for this dataset through transfer learning would be the easier, quicker and most sensible way. To start with this approach, 3 pretrained models were selected and studied:

- YOLO
- SOLO
- DeepLab

Finally DeepLab was chosen and the project started by creating scripts adapting both the dataset to be procesed with this model and adapting the model to the project needs.
After some weeks of working with this approach many complications appeared, which slowed dramatically the progression of the project due to the lack of documentation in the internet about how to adapt certain datasets to DeepLab. Then it was decided to change the approach and start from scratch.
The next and final approach was to build **U-NET** an unet model and train it from scratch using pyhton, with the help of the U-NET paper and some usefull guides found on the internet.

3 Python Scripts

This project is based on 4 Python scripts:

- *unet.py*
- *data.py*
- *train.py*
- *extra.py*

3.1 Script *unet.py*

This script implements a modification of the model described in the paper "U-Net: Convolutional Networks for Biomedical Image Segmentation".
The U-Net model as described in the paper is based on a contracting or "down" pathway, a bottleneck, and an expanding or "up" pathway.
Both the contracting and the expanding pathways are formed by a series of two convolutional layers, followed by a rectified linear unit and finally a pooling layer in the case of the downsampling path, and an up-conv in the case of the upsampling part.
Specifically, this structure is repeated 4 times for compression and other 4 times for expansion.

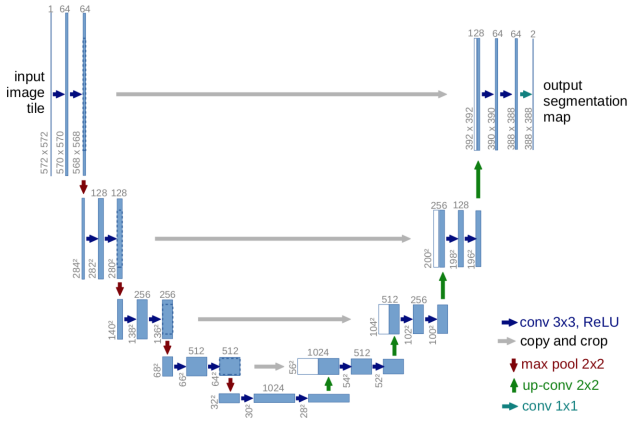


Figure 1: U-NET

Two relevant modifications have been applied to the original U-NET model.
The first one is that in the model used for this project the convolutional layers are padded.
The second one is that batch norm is applied after each convolution.

3.2 Script *data.py*

This is the simplest script of the whole project. In this script the dataset will be prepared to be used to train our model.

3.3 Script *train.py*

This is the main script of the project. As it's name indicates it's used to train the model and will call every other script. This script has many parameters that can be modified by the user to find the best configuration for the dataset and the system in which the model is trained. Some parameters that should be remarked are:

Batch size
This parameter the number of examples utilized in each iteration of the training process. A change in this parameter can change both the accuracy and the speed of the trained model, so it will be one of the parameters modified in the experimental part.

Image width and height
Due to technical limitations of the systems training the model, in this project the model will always be trained with a fixed image size of 240x160 pixels.

Number of epochs
An epoch is a pass of the entire dataset through the model. In the experimentation part this model has been trained with 10 epochs for each different configuration.

Train images, train masks, test images and test masks
This parameters are used to indicate to the script the directories in which it has to look for each resource.

Learning rate
This is another parameter which could highly impact the training of the model. For this reason, this will be the most changed parameter in the experimental part. It determines how much will change the model in response to the estimate error.

Load project
This boolean parameter will determine if the training will be done on the top of an already trained "checkpoint" or configuration of the model.

3.4 Script *extra.py*

This script will implement some utility functions:

- *save*() (Save checkpoint)
- *load*() (Load checkpoint)
- *get_loaders*()
- *check_accuracy*() (Calculates both accuracy and Dice score)
- *save_predictions_as_images*()

4 Experiments

4.1 Introduction

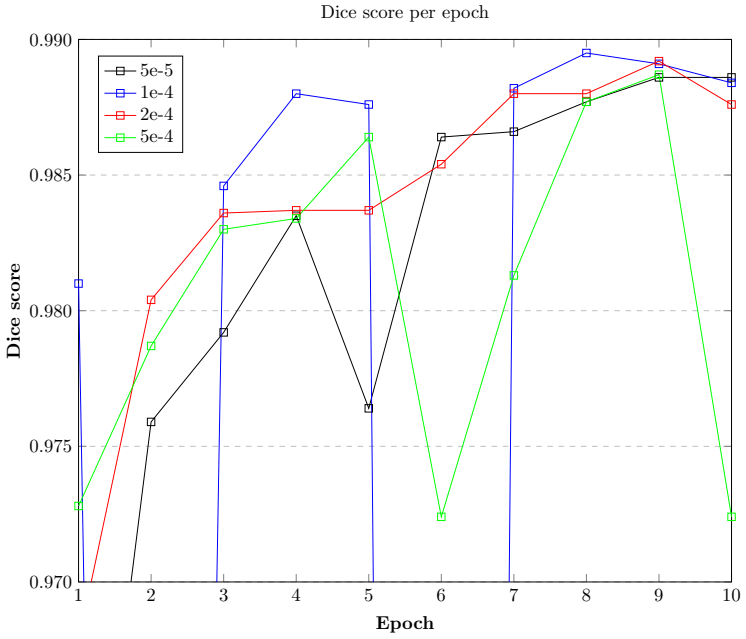
In this section 2 parameters and one feature of the model will be changed and combined in 6 experimental trainings of 10 epochs. Specifically the model will be trained with:

- 4 different learning rates
- 2 different batch sizes
- 2 different convolution kernel sizes

4.2 Learning rate

To calculate the effect of changing the learning rate (lr) the batch size will be fixed to 16 and the kernel size to 3.

	Dice Score			
Epoch	**lr = 5e-5**	**lr = 1e-4**	**lr = 2e-4**	**lr = 5e-4**
1	0.9546	0.9810	0.9678	0.9728
2	0.9759	0.8296	0.9804	0.9787
3	0.9792	0.9846	0.9836	0.9830
4	0.9835	0.9880	0.9837	0.9834
5	0.9764	0.9876	0.9837	0.9864
6	0.9864	0.7070	0.9854	0.9724
7	0.9866	0.9882	0.9880	0.9813
8	0.9877	0.9895	0.9880	0.9877
9	0.9886	0.9891	0.9892	0.9887
10	0.9886	0.9884	0.9876	0.9724

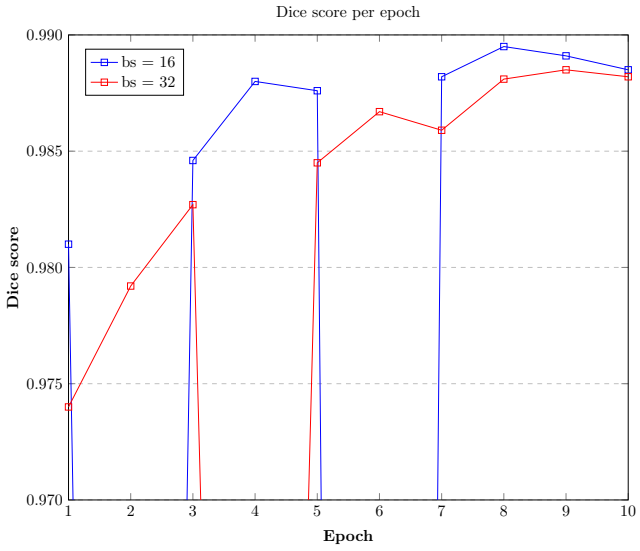


If the attention is focused in the higher part of the dice score (From 0.97 to 0.99), where the majority of the values are, and also take a look at the table of values we can obtain the following observations:

- Every learning rate presents a high instability except 2e-4. This is specially big in 1e-4.
- The last few epochs of the training usually have a very similar dice score.

4.3 Batch size

In this case batch sizes (bs) of 16 and 32 will be compared by fixing the learning rate to 1e-4 and the kernel size of convolutional layers to 3.



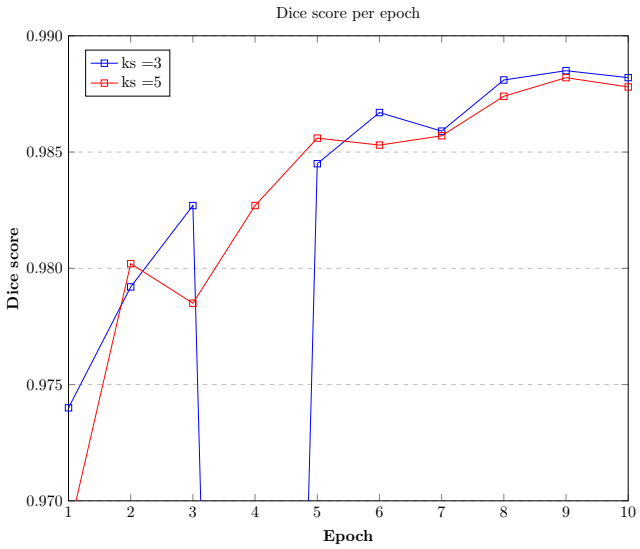
	Dice Score	
Epoch	**bs = 16**	**bs = 32**
1	0.9810	0.9740
2	0.8296	0.9792
3	0.9846	0.9827
4	0.9880	0.8818
5	0.9876	0.9845
6	0.7070	0.9867
7	0.9882	0.9859
8	0.9895	0.9881
9	0.9891	0.9885
1	0.9885	0.9882

In this case, again we can see instability in both experiments. However, it must be noted that except for those moments, the experiment with batch size of 16 is always more accurate than the one with batch size of 32.

4.4 Convolutional kernel size

The convolutional kernel size is one of the features of the model that could affect accuracy. To check it 2 experiments will be run, one with kernel size 3 and padding 1, and another one with kernel size

5 and padding 2. The padding must also be changed together with the kernel size because we want "same convolution" layers, that is, a convolutional layer which output has the same size as its input. In these experiments the learning rate will be of 1e-4 and the batch size of 32.



	Dice Score	
Epoch	**ks = 3**	**ks = 5**
1	0.9740	0.9686
2	0.9792	0.9802
3	0.9827	0.9785
4	0.8818	0.9827
5	0.9845	0.9856
6	0.9867	0.9853
7	0.9859	0.9857
8	0.9881	0.9874
9	0.9885	0.9882
1	0.9882	0.9878

Here kernel size of three seems to be slightly better than kernel size of 5, but more unstable.

5 Conclusion

After all these experiments we can conclude that the U-NET implementation for the Carvana dataset is a pretty robust model, and, after 10 epochs performs well for every reasonable parameter. Changes in parameters only lead to very small modifications.
In this case, from the observations we can deduce that the smaller the kernel size and batch size, the better accuracy of the prediction.
However, since the sample is pretty small, we can't be sure that these aren't coincidences.

References

[1] Ronneberger, O., Fischer, P. & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation.

[2] Ioffe, S. & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.