UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

33979 - Visión por Computador

Computer Vision Lab

Author:
Eriksen, Knut Vågnes

29th May 2021

Contents

List of Figures

List of Tables

1 Gender Recognition

1.1 No limit on parameters

When trying to implement this model to achieve above 97 % accuracy I understood after some while that recompiling the model did not reset the weights of the model as I expected from my background in C, but rather allowed me to continue training from my previous checkpoint. As this was not what I had planned, I discarded the results up until this points. Luckily though, I experienced some strange results now and then, which gave me the opportunity to research a bit more and widen my knowledge.

From my previous trials and errors I made the model given in [2]. It consists of 5 layers of *Convolution + Batch Normalization + Convolution + Batch Normalization + Max Pooling* and a final *Flatten* and *Dense* to obtain the gender classification. I used *Adam* as the optimization function with a learning rate of 0.001, and *Categorical Cross Entropy* to calculate the loss. I used the data augmenter shown in Appendix A.

The learning development of the model can be seen in Figure 1, and the final results obtained is given in Table 1. Even though the model achieves a validation accuracy greater than 97 % I was not entirely happy about the deviation between loss and validation loss as shown in **??**. Furthermore, the model seems quite volatile, and while some of the spikes can be explained by using *Cross Entropy* as the loss function, I tried doubling the amounts of epochs and reduce the learning rate by a factor of 10 to hopefully obtain a more stable result. As you can see from Figure 2 it did not help much, as the validation loss was swinging around the same point as before, and validation accuracy did not increase. Since the CNN given in [2] achieved above 97 % I concluded my work.

Table 1: Gender Recognition CNN Validation Results

Validation accuracy	0.9792
Validation loss	0.0818

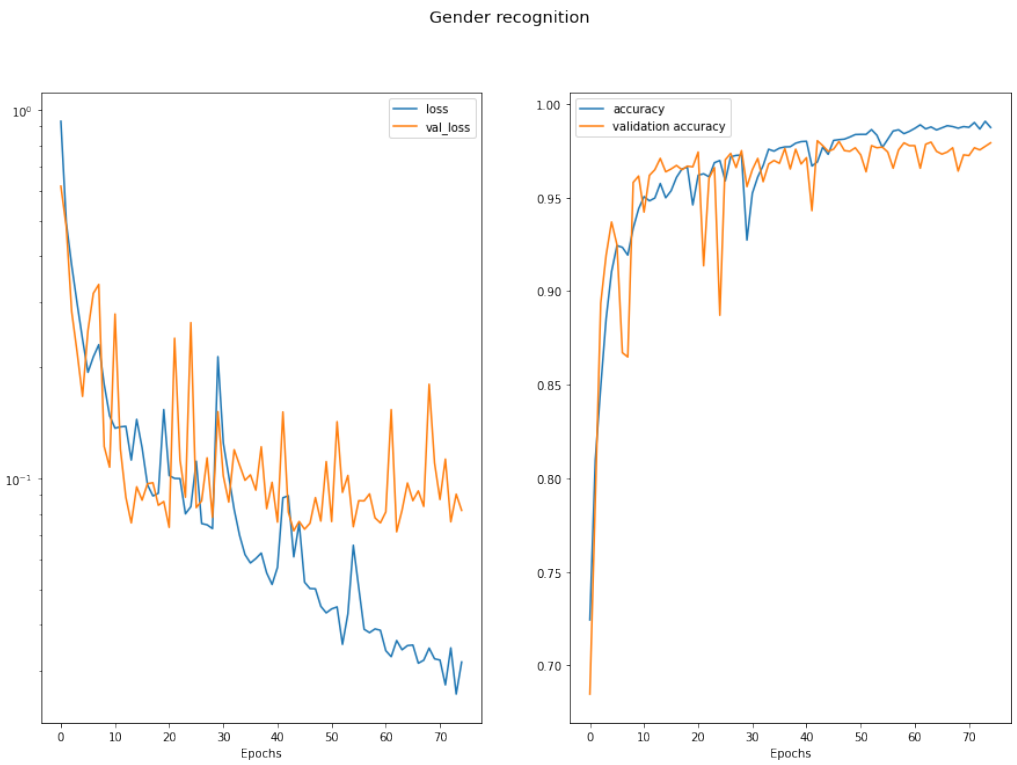

Figure 1: Gender Recognition CNN Training Development

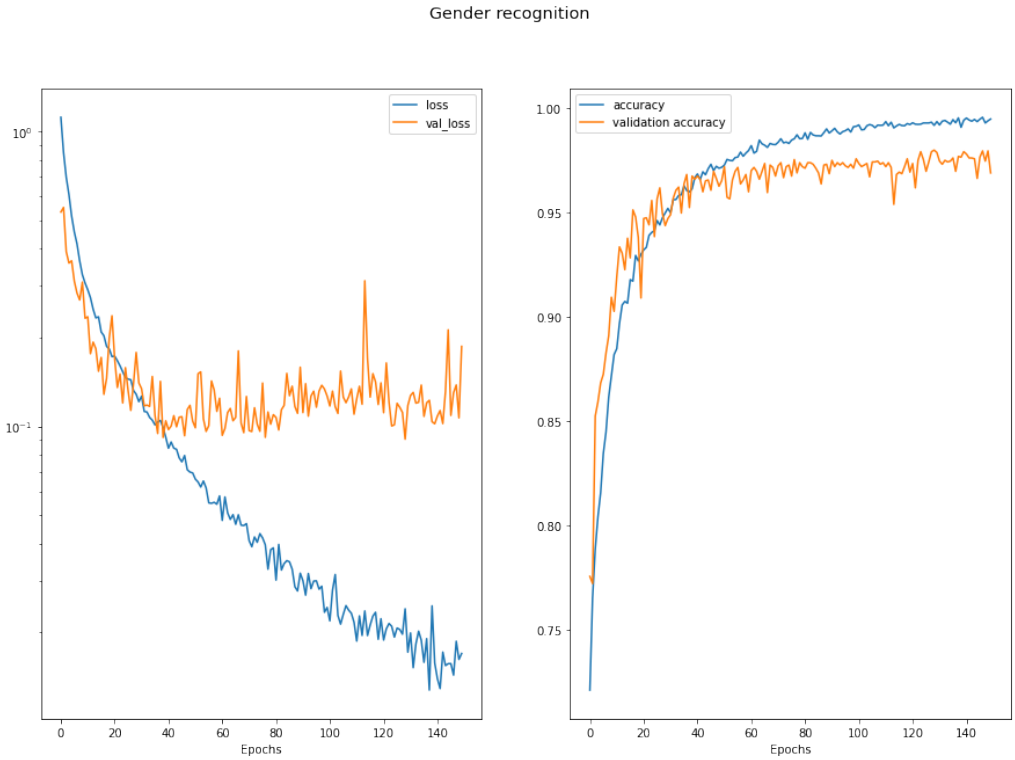

Figure 2: Gender Recognition CNN Training Development

1.2 Less than 100.000 parameters

Having first gotten the accuracy of the no-limited model to over 97% it was quite easy to follow the idea suggested in [1] and implement a model with less than 100.000 parameters. The model implemented in [2] yields the parameter distribution shown in Table 2. I used *Adam* as the optimization function with a learning rate of 0.001, and *Categorical Cross Entropy* to calculate the loss. I used the data augmenter shown in Appendix A.

The training development can be seen in Figure 3, and the final results obtained is given in Table 3. As you can see from Figure 3 the model seems to converge between 96-98 %, while it still seems that the losses could be further improved. Since the model achieved the goals from the given task I concluded my work.

Table 2: Gender Recognition CNN with <100.000 parameters

Total params	91.410
Trainable params	90.962
Non-trainable params	448

Table 3: Gender Recognition CNN Validation Results with <100.000 parameters

Validation accuracy	0.9780
Validation loss	0.0706

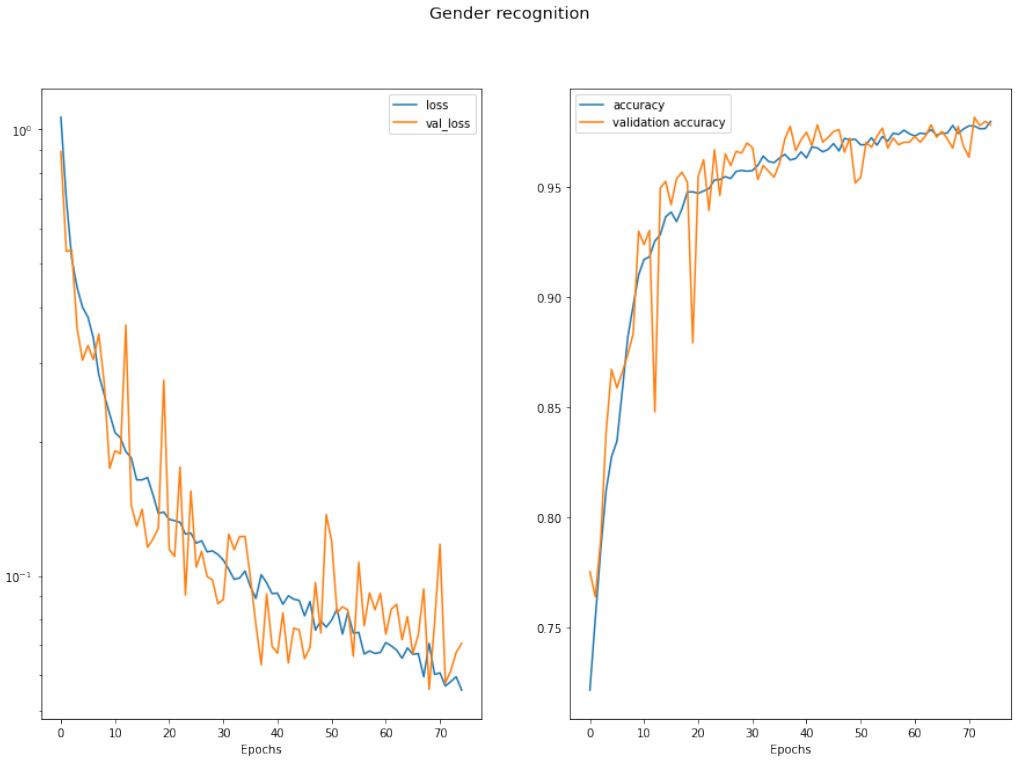

Figure 3: Gender Recognition CNN with <100.000 parameter Training Development

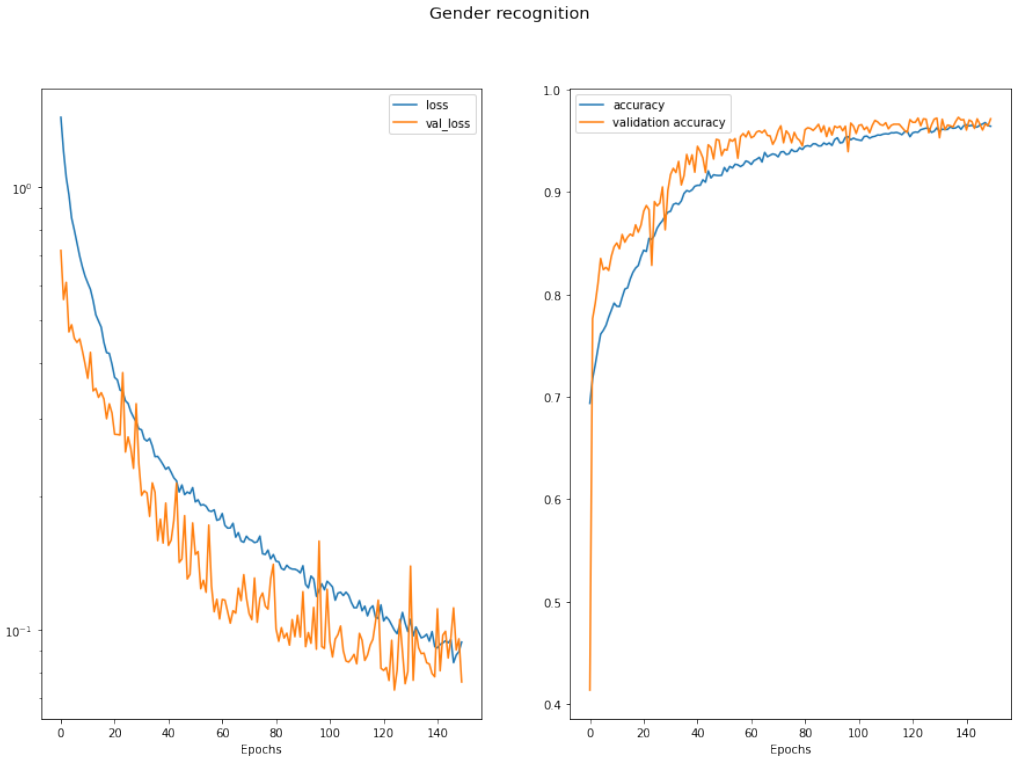

Figure 4: Gender Recognition CNN with <100.000 parameter Training Development

2 Car Model Identification with Bi-Linear model

Following the suggestions given in the task

- Load a pre-trained VGG16, Resnet... model
- Connect this pre-trained model and form a bi-linear
- Train freezing weights first, unfreeze after some epochs, very low learning rate

I wanted to try out two different strategies to form the feature-extracting parts of the bi-linear model.

1. 1 pre trained VGG16 model and 1 non-trained self made model
2. 2 pre trained VGG16 models

The first strategy turned out, as you confirmed in our Skype call, to be a rather bad idea, as it would be very difficult to get the two different CNN's to combine well. Training this network different ways gave an average validation accuracy of 30%, and over all performed very poorly. So after our call I gave up on this method, and fully focused on the second one.

Following the second method i first tried using *SGD* as the optimizer, freezing the VGG16 weights for 150 epochs, and unfreezing them for 50 epochs. I also tried with different momentum and learning rate pairs, in the range $[0.0001, 0.1]$, but always a slower learning rate with the unfrozen weights than with the frozen weights. In addition to this I also tried using different layers of the VGG16 model as the input to the outer-product function, while also testing the effect of different batch sizes during training. None of the combinations I tried performed well, with the highest accuracy obtained around 40%. Generally speaking, the combinations that performed the best had a rather quick learning rate, some momentum and used the final layer of the VGG16 model as input to the outer-product function.

Since I was quite far off the least expected validation accuracy of 65% I followed your suggestions of swapping out the *SGD* optimizer for the *Adam* optimizer, and using a frozen learning rate of 0.0001 and an unfrozen learning rate of 0.00001 with the final layer of the VGG16 model as input to the outer-product function.

This performed much better than the previous *SGD* attempts, and the training development can be seen in Figure 5 and Figure 6. From Figure 5 you can see that the model is slowly approaching convergence on validation accuracy, and if further training had been done we might have seem it converge around 50% accuracy. The validation loss though seems that could be further improved with more training. In Figure 6 you can see that the further training of the model with the VGG16 weights unfrozen lowers test loss a bit from Figure 5, and increases the validation accuracy with about 65% to around 75%. Here as well you can see that the validation accuracy is approaching convergence, and with further training I would believe it would converge around 80% accuracy. The validation loss also seems to approach convergence. The final results from the two training periods is given in Table 4 and Table 5. Since the model obtains around 77% validation accuracy after the final training, which is more than the expected 65% I concluded my work on this model. You can see the model in [2].

Table 4: Car Model Recognition Validation Results after first training with frozen weights

Validation accuracy	0.4694
Validation loss	2.3366

Table 5: Car Model Recognition Validation Results after second training with unfrozen weights

Validation accuracy	0.7768
Validation loss	1.1156

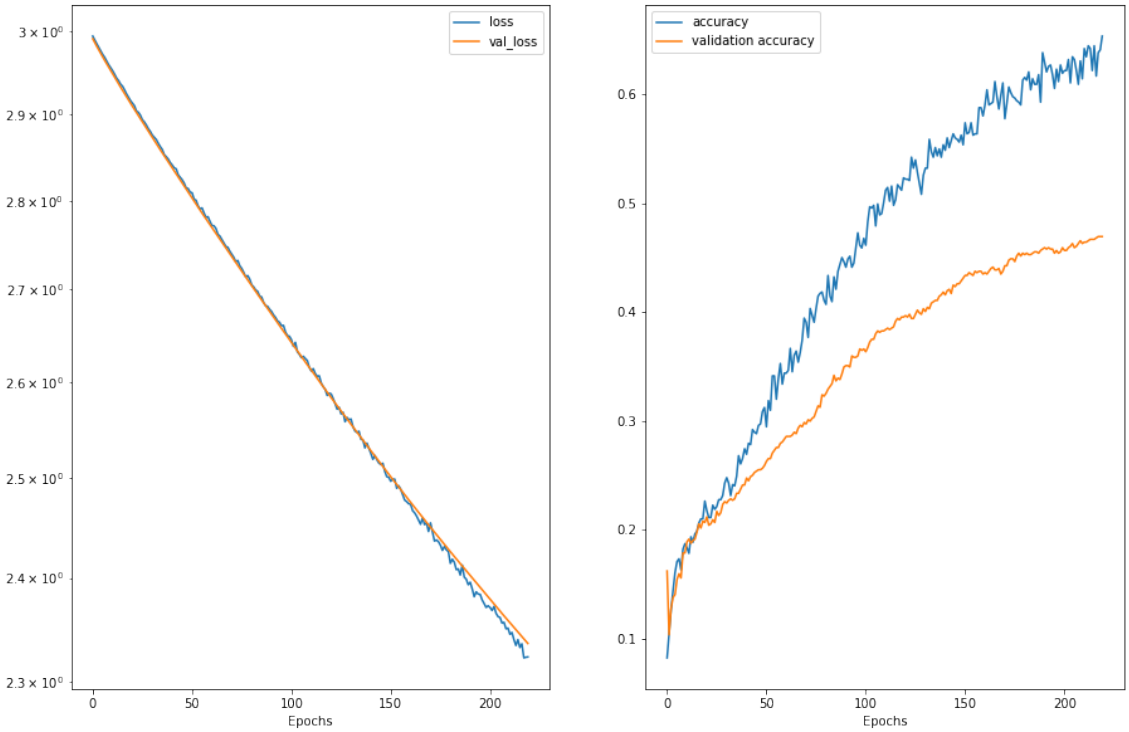

Figure 5: Car Model Recognition Training Development with Bi-Linear CNN with frozen weights

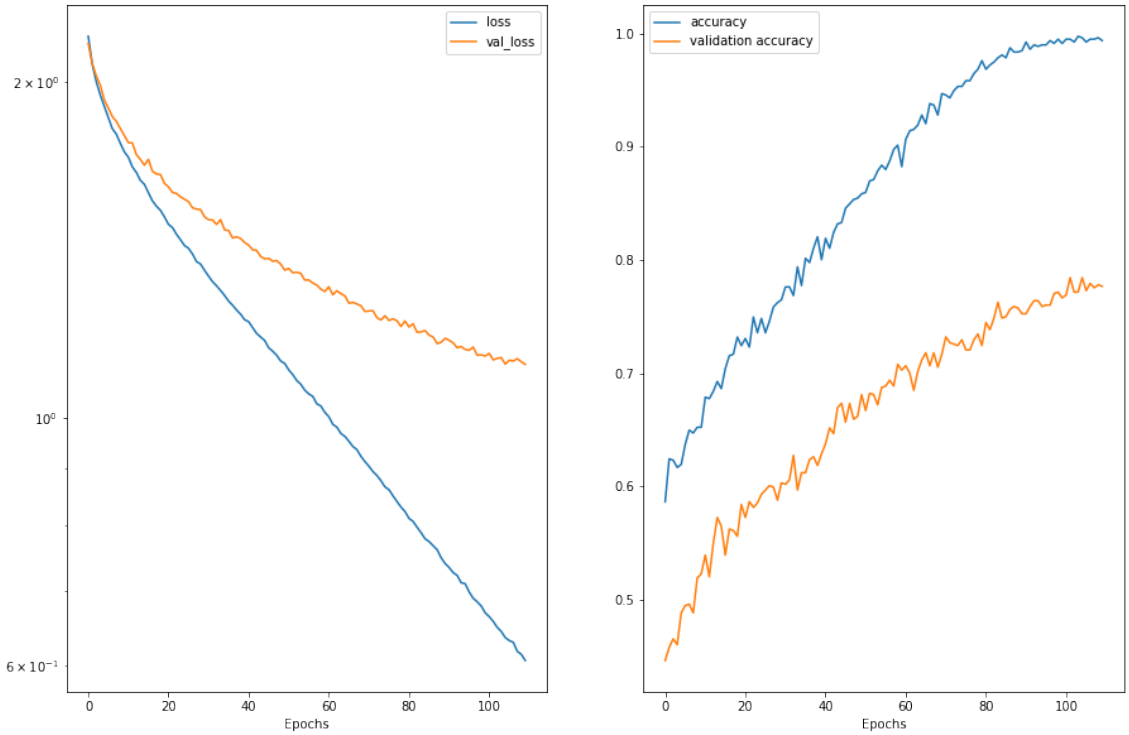

Figure 6: Car Model Recognition Training Development with Bi-Linear CNN with unfrozen weights

3 Image Colorization

See the imagecolor notebook in [2] for this task. First i did a quick training, with 10 epochs and 5 steps per epoch. After that I did a longer training session with 100 epochs and 10 steps per epoch. You can see the results from both sessions in [2].

Bibliography

[1] Grigory Antipov, Sid-Ahmed Berrani and J. Dugelay. 'Minimalistic CNN-based ensemble model for gender prediction from face images'. In: *Pattern Recognit. Lett.* 70 (2016), pp. 59–65.

[2] Knut Vågnes Eriksen. *VisionPorComputador*. https://github.com/knuteriksen/VisionPorComputador. 2021.

Appendix

A Image Data Generator

```
ImageDataGenerator(
  width_shift_range=0.2,
  height_shift_range=0.2,
  rotation_range=20,
  zoom_range=[1.0,1.2],
  horizontal_flip=True
 )
```