INFOMCV Assignment 5 Report

**Gideon Ogilvie (5936373) & Jeroen Schut (5640059)**

**Group 56**

# architecture choices

1. Frame CNN architecture:

Afbeelding met tekst

Automatisch gegenereerde beschrijving

In the above image the architecture of the Frame CNN is shown. We tried different amounts of convolutional layers, but we initially got the best results with two convolutional layers. From there we tried different values for kernel sizes and strides. For the kernel size we found 5x5 to work the best and for strides we have 2 for the first layer and 3 for the second. This gave us the best results and while there are no doubt better configurations, we felt that the results were as good as they were going to get without spending too much time.

1. Optical flow CNN architecture:

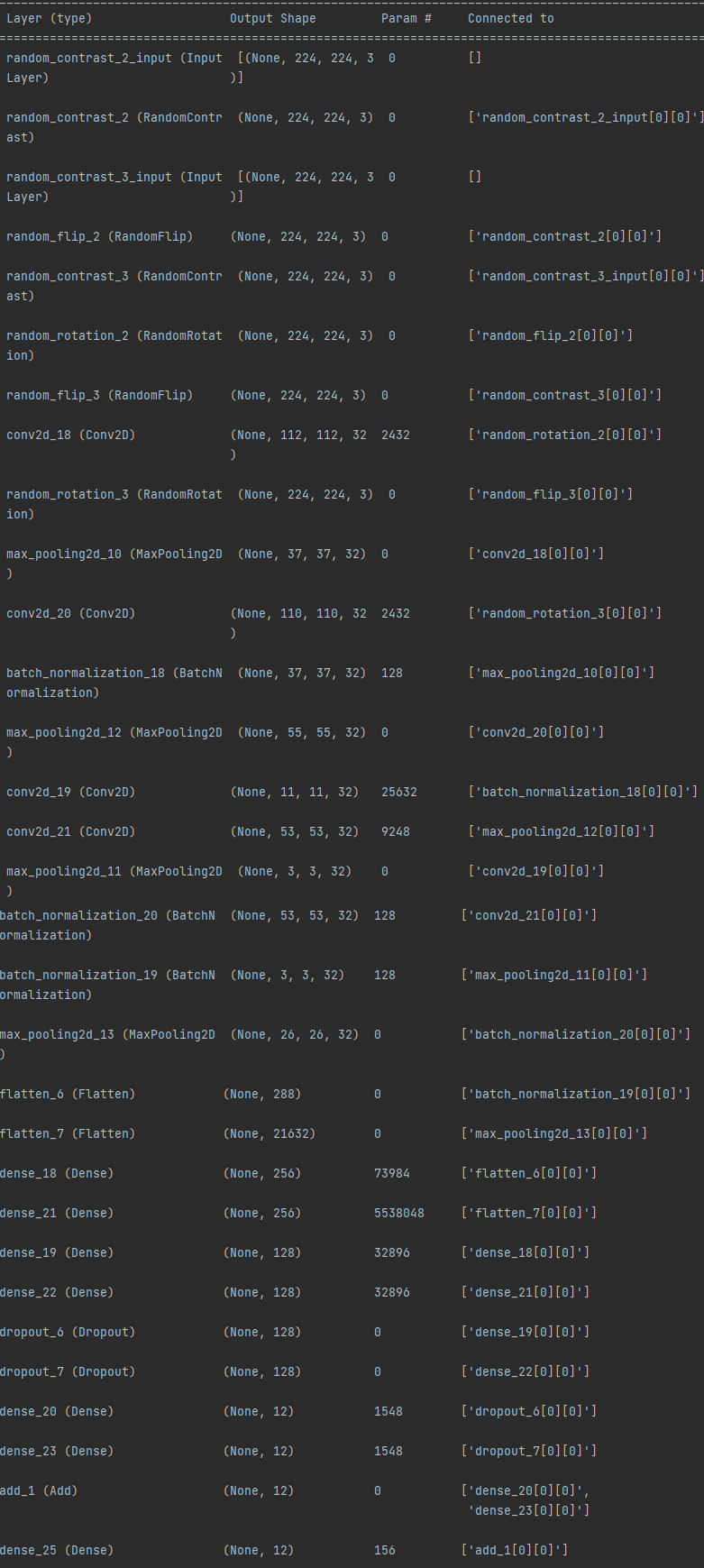
Afbeelding met tekst

Automatisch gegenereerde beschrijving

Our optical flow CNN is rather similar to our Frame CNN and this is because we started with that network as base. We tried fiddling around with the layers and parameters, but we did not get much better results, so we stayed close to our original network.

The biggest difference is the max-pooling layers. The max pooling size was 3x3 for the Frame CNN, but we ended up using 2x2 for the Flow CNN, because that gave us better results. Asa consequence, however, the first fully connected layer has a staggering 5538048 parameters. Perhaps this is a bit much, but considering the training did not take too long and the performance was okay, we felt it was fine to keep this many parameters.

1. Combination into two-stream network:



To fuse our two networks, we combined the output neurons of the Frame CNN and the Flow CNN in a dense layer. We chose this simple strategy because our Flow CNN did not have the greatest accuracy and therefore, we did not want it to have too much impact, but rather play a supporting role in the decision-making process. By doing this we hoped that the Frame CNN would use the Flow CNN mostly as a support or tiebreak in the case of two close classes and thus improve accuracy.

# Results

## Stanford 40 Frames

Afbeelding met grafiek

Automatisch gegenereerde beschrijvingAfbeelding met grafiek

Automatisch gegenereerde beschrijving

## HMDB51 Frames

## Afbeelding met grafiek Automatisch gegenereerde beschrijvingAfbeelding met grafiek Automatisch gegenereerde beschrijving

## HMDB51 Optical flow

Afbeelding met grafiek

Automatisch gegenereerde beschrijvingAfbeelding met grafiek

Automatisch gegenereerde beschrijving

## HMDB51 Two-stream

## Afbeelding met grafiek Automatisch gegenereerde beschrijvingAfbeelding met grafiek Automatisch gegenereerde beschrijving

## Result table

1. Summary of your results per model

| Dataset | Model specifications | | | |
| --- | --- | --- | --- | --- |
| Model | Top-1 accuracy (train / val / test) | Top-1 loss train | Model parameters |
| Stanford 40 | Frames | 0.4819 / 0.4161 / 0.3861 | 1.4890 | 136,748 |
| HMDB51 | Frames | 0.4048 / 0.4048 / 0.3972 | 1.7907 | 136,748 |
| HMDB51 | Optical flow | 0.3862 / 0.2024 / 0.20 | 1.8436 | 5,584,300 |
| HMDB51 | Two-stream | 0.502 / 0.5238 / 0.4055 | 2.1006 | 5,721,204 |

# Discussion of results

*Discuss the performance of your models, and compare the results between models. Explicitly discuss the training procedure, and risk of overfitting. (one column)*

To train our Stanford-40 network, we used optimizer ‘Adam’ with 25 learning epochs, with a focus metric of accuracy. We also used these numbers for training our HMDB51 net, except we only used 15 epochs, as the network did not seem to improve after 15 anyway and only overfitting seemed to increase.

The performance of the Stanford-40 and HMDB51 Frames network are very similar to one another. This makes sense, as they are essentially the same network, but with different data. Their validation accuracies are around 40%, which is quite alright considering the limited data and only have one frame for identifying an action. Overfitting seems to be minimal for both. Stanford-40 has some, with 48% training vs 41% validation accuracy. This is likely the result of the data augmentation layers discussed in choice task 1 and 2.

To train the optical flow network we also used optimizer ‘Adam’ with a learning rate of 0.001 and we again optimized for accuracy. We used 15 epochs to train this network.

The optical flow network has a way lower validation accuracy with way more overfitting. Why this exactly is, is hard to say, but it possibly because a lot of other information is lost outside of the movement information. For gesture recognition movement is not the only important element. Pose, for example is also very important and this information is partly or even completely lost when using optical flow frames.

For our fusion model, we again used ‘Adam’ with a 0.001 learning rate, optimized for accuracy with 20 epochs.

For the two-stream network, we get slightly higher values for train and validation accuracy, with roughly 50% for both. This seems to suggest that the inclusion of the optical flow image gives enough extra information to the neural network that it can use to determine a better result than either of its components separately. Sadly, this seems to break down in the test accuracy, which drops to 40%, which is comparable to the accuracy of just the Frame network. This could be the result of random chance, however, as both the training and validation accuracies match fairly closely.

# Link to Model Weights

<https://drive.google.com/file/d/16KysSGfZl6OJj3B1SB_M2-6dM4Gmi9SM/view?usp=sharing>

# Choice Tasks

*Choice 1:* We performed data augmentation on the Stanford-40 set. We tried several configurations, but in the end, we ended up using a random contrast layer with value 0.05, a random flip in the horizontal direction, as the vertical direction would likely alter the action too much, and a random rotation with 0.1 as value. We also attempted to use a random brightness layer, but whenever we added one, even with a small value as argument, our validation accuracy dropped to guessing level, so we opted not to use this.

As a result, we got comparable validation accuracies, perhaps slightly higher for the network with data augmentation, but the most significant result was in the reduction of overfitting. With roughly the same validation accuracies, the training accuracy for the network without the data augmentation reached roughly 95%, whereas with data augmentation the training and validation accuracies were very similar.

*Choice 2:* We also performed data augmentation on the HMDB51 set and we ended up with roughly the same results as the data augmentation of the Stanford-40 set. Again, random brightness seemed to kill the network, so we did not add a layer of it to the network.

*Choice 3:*

We used a cyclical learning rate scheduler in a separate network for the Stanford-40 set. It has a rather small cycle, because we did not use that many epochs and we also multiply by a variant of 1 / x to make the learning rate decrease overtime. In the graph below the learning rate is shown as a function of the epochs.

**Chart, histogram

Description automatically generated**For this network, we got a learning accuracy of 0.4368, a learning loss of 1.6208, a validation accuracy of 0.3723 and a validation loss of 1.8906. Because these numbers were not better than our values of the network without cyclical learning, and because we already completed everything else without cyclical learning rate, we did not do the other experiments with cyclical learning rate.

*Choice 5:*  To test the influence of which frame was chosen, we reran all our networks, except the Stanford-40 model, with different frames than the halfway frame. We tried at intervals of 25%, so we tried frames at 0%, 25%, 50%, 75% and 100%. We got the following results:

Pretrain:

|  |  |  |  |
| --- | --- | --- | --- |
| Frame % | Train Acc | Val Acc | Top 1 Loss Train |
| 0% | 0.3108 | 0.381 | 1.9876 |
| 25% | 0.3664 | 0.4048 | 1.8314 |
| 50% | 0.3981 | 0.4405 | 1.8018 |
| 75% | 0.4008 | 0.369 | 1.7629 |
| 100% | 0.3783 | 0.3929 | 1.798 |

Flow:

|  |  |  |  |
| --- | --- | --- | --- |
| Frame % | Train Acc | Val Acc | Top 1 Loss Train |
| 0% | 0.2593 | 0.2143 | 2.1343 |
| 25% | 0.3598 | 0.1905 | 1.9012 |
| 50% | 0.4233 | 0.2738 | 1.7644 |
| 75% | 0.3095 | 0.25 | 2.15 |
| 100% | 0.3558 | 0.25 | 1.8907 |

Twostream:

|  |  |  |  |
| --- | --- | --- | --- |
| Frame % | Train Acc | Val Acc | Top 1 Loss Train |
| 0% | 0.2738 | 0.2262 | 2.3584 |
| 25% | 0.3598 | 0.25 | 2.3163 |
| 50% | 0.2712 | 0.2024 | 2.3488 |
| 75% | 0.2659 | 0.2381 | 2.3499 |
| 100% | 0.2685 | 0.2262 | 2.3478 |

Going by validation accuracy, we can see that there are no big differences between the frames, but the middle frame performs the best for the pretrain and the flow network. This is not the case for the twostream network where the best performer was the frame at 75%.

Because the performance of all networks is rather similar no matter the frame, it seems that the frame selection does not have a major impact on the result. While this result seems counter intuitive, as you might expect there to be certain frames of actions to be more import, it is not entirely unexpected either. For every action the important frames can be elsewhere in the video, so this could middle out so that it does not really matter which frame you pick, if you must pick the same frame for all the videos. It could also be that our investigation was not granular enough, because we went in steps of 25%. Going in smaller steps, like 10% or even 5% could reveal that there are certain frames that are more optimal.