Assignment 5: Action recognition – group 41

Emiel Eliens 7683634 Pim Veraar 6409458

1 Architecture Choices

In this project, four different Convolutional Neural Network (CNN) architectures were implemented, using the Keras TensorFlow interface for Python. Two different data sets were used in order to train these CNNs. The first data set was the Stanford 40 data set, which is an image-based data set for action recognition and consists of 40 classes, with 180-300 per class, resulting in a total of 9532 images for the data set in its entirety. Only 12 of those output classes were relevant for the current project, an overview of which is given in table 1. The second data set which was utilized, was the HMDB51 data set, which consists of short videos describing 51 action classes, with 100-150 videos per class, resulting in a total 6845 videos for the data set in its entirety. In a similar fashion as with the Stanford 40 Data set, only 12 of these output classes were deemed relevant, and these can also be observed in table 1. The differing nature of these two data sets (RGB images vs videos), have allowed us to train different architectures, which in turn were combined into our overarching two stream network. The need to train separate networks which are combined later on, arises from the fact that the HMDB51 data set is far smaller than the Stanford 40 data set, when we only take the relevant classes into account. In order to make the most of the available data, we first trained a CNN on the Stanford 40 data set, which was then fine-tuned on the middle frames of the videos of the HMDB51 data set. We also trained a CNN on the optical flow calculated from taking a subset of frames from the videos of the HMDB51 data set. Finally, the fine-tuned frame CNN and the Optical Flow CNN trained on the HMDB51 data set formed the foundation for our two-stream network, in which the two were combined in order to take both the RGB frames of the videos as well as the optical flow of the videos of the HMDB51 dataset into account for the task of action recognition.

Table 1: Data set descriptions

Stanford 40	# instances	HMDB51	# instances
applauding	180-300	clap hands	100-150
climbing	180-300	climb	100-150
drinking	180-300	drink	100-150
jumping	180-300	jump	100-150
pouring liquid	180-300	pour	100-150
riding a bike	180-300	ride bike	100-150
riding a horse	180-300	ride horse	100-150
running	180-300	run	100-150
shooting an arrow	180-300	shoot bow	100-150
smoking	180-300	smoke	100-150
throwing frisbee	180-300	throw	100-150
waving hands	180-300	wave	100-150
Total	3037		1200

1.1 Frame CNN Architecture: Stanford40

The architecture generally used for all data sets was inspired by Diba et al. [1] Five convolutional layers where used followed by 2 dense layers. Eight filters were used in the first convolutional layer, with double the amount of filters in every following convolutional layer. A stride of 1 was used with no padding. These filter sizes were chosen as we found they gave a good balance between the desired complexity and computation time. Max pooling was used after every convolutional layer to further reduce the complexity of the network. The first dense layer consists of 128 nodes with a L2 regularization penalty of 0.01 to reduce overfitting. In line with our network of the previous assignment, a dropout layer was added after the last convolution layer with a dropout rate of 0.2 and after the first dense layer with a dropout rate of 0.5. The model was then trained using the Adam optimizer with the standard learning rate of 0.01 for 15 epochs and a batch size of 32.

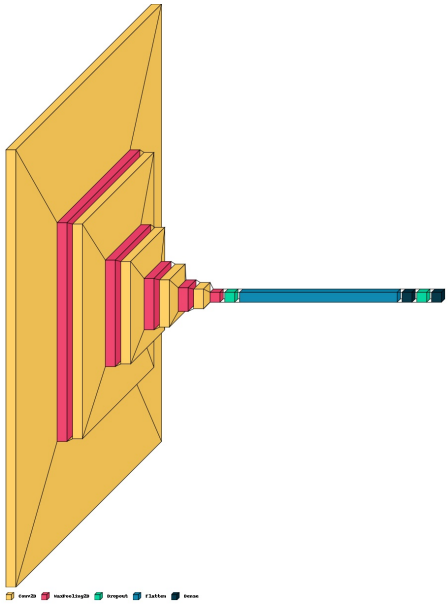


Figure 1: Baseline model visual

1.2 Frame CNN Architecture: HMDB51 Transfer learning

For this step, the Stanford40 model as seen in Figure 1 was trained on the HMDB51 dataset using transfer learning. First, all layers of the Stanford40 model were frozen, except for the output layer. This model was then trained on the middle frame selected for every video in the dataset for 15 epochs using a Stochastic Gradient Descent optimizer with a learning rate of 0.01 and a momentum of 0.9. Due to the fact that only a small amount of parameters had to be trained, it allowed for a greater batch size of 256.

After this, all weights were unfrozen and the model was fine-tuned on the same dataset using the same settings for only 8 epochs. As can be seen from the results in Figure 6, the fine-tuning improves the overall performance of the model greatly.

1.3 Optical Flow CNN architecture: HMDB51

For every video in the HMDB51 dataset, optical flow was performed using the middle sixteen frames. The corresponding CNN thus has to be of a 3D nature. While a similar architecture to the Stanford40 is used, the layers are switched with their 3D counterparts to accommodate the different data structure. The only structural difference to the Stanford40 model is that the last convolutional layer consists of only 64 layers, compared to 128 in the Stanford40 model. This was done to reduce the amount of parameters in the following flatten layer, as we found that the optical flow model suffered a lot more from overfitting compared to the previous models. We also train the model for only 10 epochs, as going beyond this resulted in the model overfitting too much. In addition, a batch size of 16 was used together with the standard learning rate of 0.01 of the Adam optimizer.

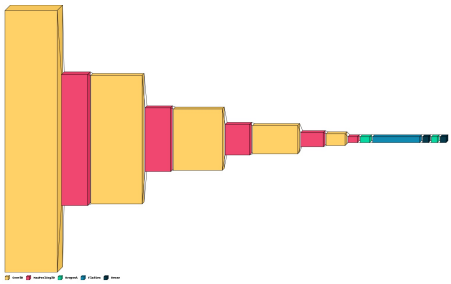


Figure 2: Optical Flow visual

1.4 Two-stream-network

To create the two-stream-network, the resulting models of step 2 and 3 (the HMDB51 Transfer learning model and HMDB51 Optical Flow) were fused into a single model. Only the convolutional layers of both models were kept, discarding the fully connected layers following them. But, as can be deduced from close inspection of Figure 1 and Figure 2, the dimensions of their output do not match. To solve this, a reshape layer was added after the optical flow model to reduce the amount of dimension of its output, and a 1x1 convolution layer of 64 filters was added to the transfer learning model to reduce the amount of feature maps from 128 to 64. These models were then fused using the keras Average layer. More types of fusion layers were considered and will be further discussed in the choice task section. A fully connected layer of 64 nodes and a L2 regularization penalty of 0.01 was added instead of the 128 nodes found in both seperate models to reduce over-fitting. The dropout layer and the output layer remain the same to the original models, however. This model was also trained for 10 epochs with a standard learning rate of 0.01 of the Adam optimizer.

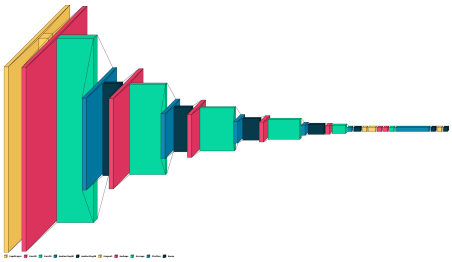
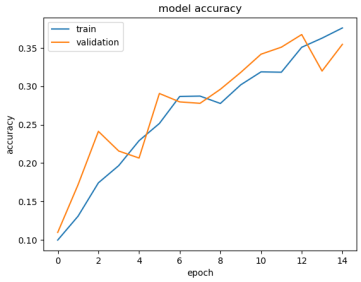


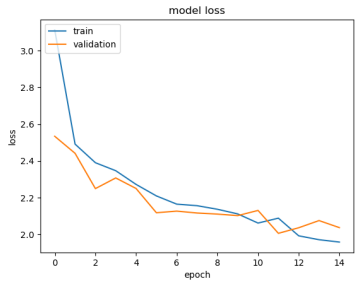
Figure 3: Two-Stream Network visual

2 Results

After each model was trained on its respective data set, the loss and accuracy curves as observed in figures 4 - 8 were



(a) Accuracy



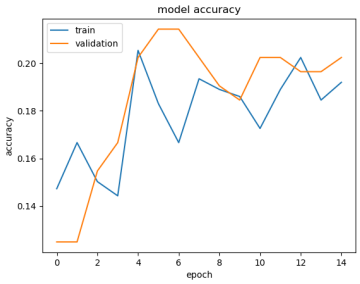
(b) Loss

Figure 4: Stanford 40 CNN accuracy and Loss plots

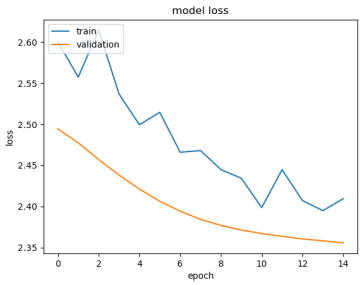
obtained, as well as the training, validation and test accuracies as depicted in table 2. Note that the number of epochs that was trained for differs among different architectures. As mentioned before, this is because certain implementations were more prone to over-fitting than others, such as the fine-tuning stage of the transfer learning network, and the optical flow networks. The risk for over-fitting is greater in these instances, because the data set on which they are trained is noticeably smaller than the Stanford 40 data set.

3 Discussion Of Results

Form the results in the previous section, one can immediately observe the impact of data availability on model performance. The CNN trained on the Stanford model performs almost as well as the Two-Stream HMDB51 model, which is far more complex in nature. This admirable level of performance is caused by the fact that the Stanford 40 model benefits from having far more data available, which also allowed for training longer on the data before over-fitting would occur. The transfer learning CNN still benefited from this increased data-availability, as it was pre-trained on the Stanford 40 data set. This model model could be trained for a similar number of epochs, during the training phase in which its earlier layers were frozen, because freezing the earlier layers in combination with a lower learning rate, allowed for this without introducing too much risk for over-fitting. During the fine-tuning stage, on the other hand, this model was trained for

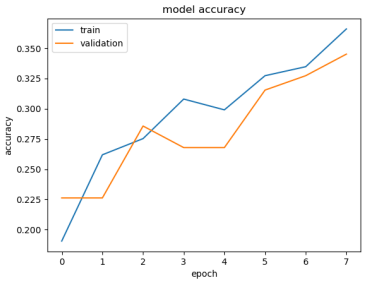


(a) Accuracy

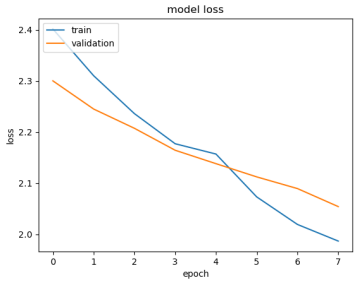


(b) Loss

Figure 5: Transfer Learning HMDB51 CNN accuracy and Loss plots

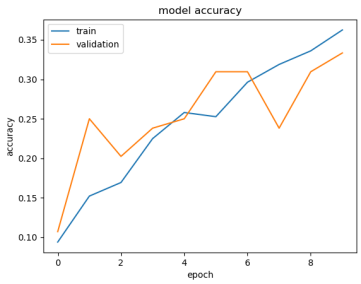


(a) Accuracy

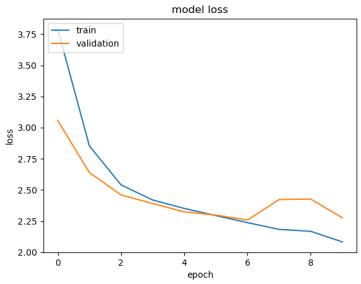


(b) Loss

Figure 6: Transfer Learning fine-tuning accuracy and Loss plots

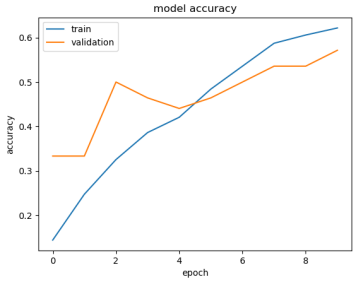


(a) Accuracy

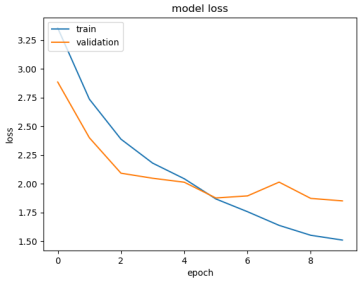


(b) Loss

Figure 7: Optical FLow CNN accuracy and Loss plots



(a) Accuracy



(b) Loss

Figure 8: Two-Stream CNN accuracy and Loss plots

far fewer epochs, because the entire model was now made trainable, which increases the risk for over-fitting with a data set as small as the HMDB51 data set. These risks were carried over into the training phase of the Optical Flow model, which was trained from scratch on the HMDB51 data set, and was therefore also trained for a fewer number of epochs than the Stanford 40 model. Finally, because the Two-Stream network was also mostly based on the HMDB51 data set, this model was trained for a shorter number of epochs as well in order to reduce over-fitting. The performance of these models as indicated in table 2, demonstrates the two-stream network's ability to overcome the limitations of having a small data set.

4 Link To Model Weights

Link to Github with model weights

5 Choice Tasks

During this project, the following choice tasks were implemented, with each of these choice tasks serving a different purpose.

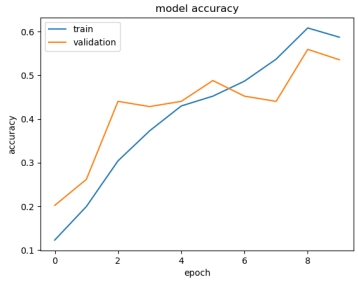
- Use 1x1 convolutions to connect activations between the two branches
- Experiment with at least three types of cooperation for the two networks (average, concatenate, maximum etc.)
- Present and discuss a confusion matrix for your (1) Stanford 50 Frames and (2) HMDB51 Optical flow models

5.1 1x1 convolutions

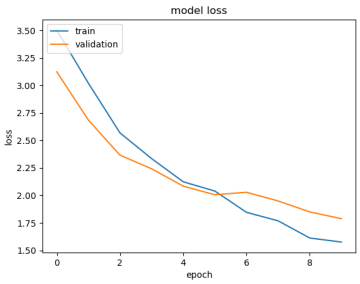
The introduction 1x1 convolutions serves both as a way to make our implementation less computationally expensive, as it represents a way to reduce the dimensionality of the activations of the frame branch of our two-stream network, and it also serves as a way to feasibly connect these two branches in the first place. Reducing the dimensionality of the frame stream allows us to merge the branches without needing the optical flow branch to also have 128 filters at the cutoff point. This is beneficial, as we found that the optical flow model was more prone to overfitting, and reducing the amount of filters to 64 in the last convolutional layer of the optical flow model seemed to have a positive effect on the results. The 1x1 convolution allowed us to reduce the dimensionality of the frame branch to match the dimensionality of the optical flow branch.

5.2 Different types of cooperation for the two networks

This choice task was implemented in order to investigate the best manner in which to combine the two networks, given that

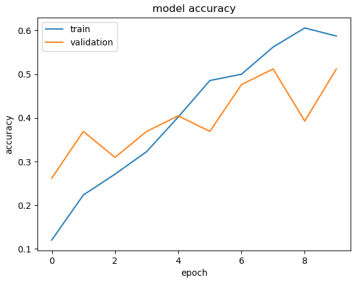


(a) Accuracy

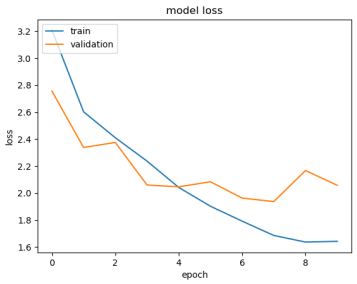


(b) Loss

Figure 9: Two-Stream Maximizing accuracy and Loss plots



(a) Accuracy



(b) Loss

Figure 10: Two-Stream Concatenating accuracy and Loss plots

the fusion happens at a predetermined point in the architecture. More specifically, our focus was on determining whether the loss of data accumulated from maximizing or the addition of data accumulated from concatenating, during the fusion process, would be accompanied by a decrease in performance, when compared to averaging the data during the fusion process. To that end the Two-Stream model was re-tested whilst utilizing these alternate cooperation strategies for the fusion layers, the results of which are depicted in figures 9-10 and table 3. As can be seen from these results, averaging seems to be the best of both worlds, with it yielding the overall highest results. This is probably due to the fact that maximizing cuts away too much detail, whereas concatenating leaves too much of it behind, resulting in slightly more noise, which the averaging method is able to suppress a bit better.

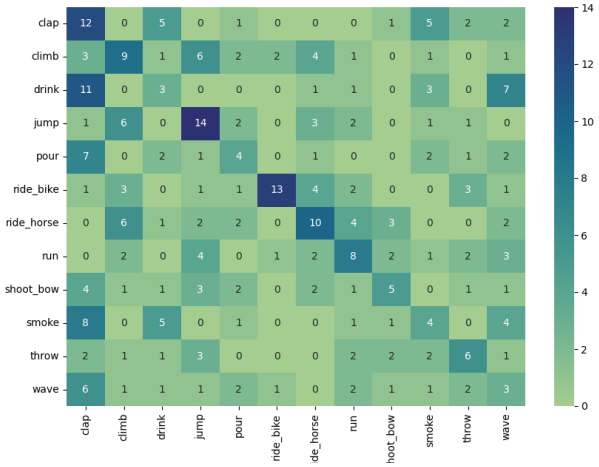


Figure 11: Confusion matrix of Stanford 40 CNN

5.3 Confusion Matrix

The final choice task that was implemented, was to utilize and discus a confusion matrix for the Stanford40 frames. The Benefit of utilizing a confusion matrix is that it allows us to see exactly which classes proved to be problematic for our architectures, which in turn can yield valuable insights, which can be used to make improvements to future versions of our architectures.

Table 2: Table with training, validation top-1 accuracy and cross validation accuracy

Dataset	Model	Top-1 acc. train/validation/test %	Top-1 loss train	Model Size (MB)	Model Parameters
Stanford40	Frames	33.7% / 31.1% / 33.9%	2.0072	6.044	509,660
HMDB51	Frames	36.6% / 34.5% / 30.6%	1.9869	6.044	509,660
HMDB51	Optical Flow	36.2% / 33.3% / 25.3%	2.0839	3.997	334,964
HMDB51	Two-Stream	62.2% / 57.1% / 36.1%	1.5112	4.105	338,372

Table 3: Table with training, validation top-1 accuracy and cross validation accuracy

Model	Top-1 acc train	validation	test %
Average	62.2%	57.1%	36.1%
Concat	58.73%	51.19%	34.44%
Maximize	60.85%	55.95%	30.55%

References

[1] Ali Diba, Ali Mohammad Pazandeh, and Luc Van Gool. “Efficient Two-Stream Motion and Appearance 3D CNNs for Video Classification”. In: Oct. 2016.