INFOMCV PR5

Alexander Alexander, 6272932, a.p.apers@uu.nl
Robert Oost, 6922856, r.w.oost@students.uu.nl

April 2022

1 Introduction

The goal of this assignment is to apply transfer learning and optical flow information to the problem of action classification from images and in videos. The main challenge with action recognition from images and videos is that the network is tasked to essentially perform multiple simpler tasks at the same time that make up action recognition. Further challenges concern properties such as the high number of possible different types of actions, high inter-class similarities between actions (such as waving and high-fives), presence of context clues (objects, environments), and unseen data.

Going from classifying action in images to videos makes it even more challenging since the action might not be present in all frames. There is the added challenge of picking out which frames to pay attention to. This is of course mitigated by using optical flow information from the frames. The movement of the people in a certain frame could give a clue about the action that will take place a few frames later, for example, the reaching out of a hand.

When combining two different networks into one it's important to feed the concatenated inputs to the model. This ensures that the model receives both types of input. Where the fusion occurs between the two networks is another design choice that can be played around with. Most common are mid-level and late fusion. In general, it is a good idea to use multiple models when two different subtasks can clearly be distinguished and the submodels can specialise in these individual subtasks. Simply adding the same model architecture twice might not necessarily improve performance on a single task.

2 Datasets

We have used the Stanford-40 Action Dataset [4] and the TV Human Interaction Dataset (TV-HI) [2]. The Stanford-40 Dataset contains forty classes with 180-300 images per class and a total of 9532 images. We used the suggested train-test split which left us with 5532 images for testing and 400 images for validation and 3600 images for training.

The TV Human Interaction Dataset only uses four classes with 300 videos in total. 100 of which are videos in which none of the four actions occur and these negative videos are therefore discarded. The 200 remaining videos were divided into 100 for testing, 10 for validation and 90 for training.

3 Data pre-processing

For the Stanford-40 dataset we performed the following pre-processing steps. First, we loaded the images from the directory *JPEGImages* and resized them to 224×224 pixels. The images were processed in the RGB color space. After this pre-processing step, we saved the jpg images into separate directories for Train, *Validation* and *Test* sets. Within these directories we made subdirectories for each of the 40 action classes. These directory splits into train and validation were made using stratification to ensure a balanced distribution of datasamples from each class and consistency between sets.

For the TV-HI dataset, we loaded each video separately and applied resizing in the same way as we processed the frames from the Stanford-40 dataset. After pre-processing the frames, we calculated the optical flow for each frame using *calcOpticalFlowFarneback()* from OpenCV. We ran this with the following settings:

- pyr_scale=0.5
- levels=3
- winsize=5
- iterations=3
- poly_n=5
- poly_sigma=1.2

We proceeded to take the middle frame from each video and used 16 optical flow frames equally spaced in time. This resulted in a (224, 224, 3) tensor for the middle frame and a (16, 224, 224, 2) tensor for the optical flow for each video. Again, stratification was used for the train and validation split.

After saving the results of these steps, all the separate sets are loaded again and made into a *tf.data.Dataset* object which ensures fast loading during training, validation and inference.

4 Main Tasks

The four main tasks we performed were as follows:

- Designing a CNN architecture to pre-train on the Stanford-40 dataset.
- Transfer learning with the pre-trained Stanford-40 model on the TV-HI dataset.
- Designing a CNN architecture to train on optical flow images generated from the TV-HI dataset.
- Fusing and training a two-stream network that uses frames and optical flow information from the TV-HI dataset.

5 Architecture choices

5.1 Base Network on Stanford-40

We started our search for a model architecture with the paper: "Two-Stream Convolutional Networks for Action Recognition in Videos" [3]. The paper uses a network with five convolutional layers and two very large fully connected layers. However, we quickly realised that we didn't have enough data to train the two large fully connected layers at the end of the network with 4096 and 2048 units respectively. So we removed one fully connected layer and only used 80 units in the remaining one. This improved the performance of the model. We proceeded to remove a convolutional layer and we added more MaxPooling layers (after every convolution) since we noticed we had a receptive field issue. This also made us increase the stride of each convolution operation. To keep the data from becoming to large we applied BatchNormalization after every convolution. The spatial integration in the model would not become large enough to pick up larger patterns in the images. We followed the general rule of using more filters at later convolutional layers and less filters at the earlier layers. The reasoning behind this is to pick up more complex patterns in the deeper part of the network. The fully connected layer is followed by a dropout layer to reduce overfitting as described in [1]. All layers use the ReLU activation function and use the padding option *same*. This is the final model we can up with:

Rescaling (input normalization) →
Conv2d_0 → BatchNormalization → MaxPool2d →
Conv2d_1 → BatchNormalization → MaxPool2d →
Conv2d_2 → BatchNormalization → MaxPool2d →
Conv2d_3 → BatchNormalization → MaxPool2d →
Conv2d_4 → MaxPool2d → Flatten →

Dense_0 $\rightarrow$ Dropout $\rightarrow$ Dense_1 (output)

with all MaxPool2d having pool size of 3×3 and stride 2 and

with Conv2d_0: 96 filters of size 7×7 and stride 2
Conv2d_1: 256 filters of size 5×3 and stride 3
Conv2d_2: 512 filters of size 3×3 and stride 3
Conv2d_3: 512 filters of size 3×3 and stride 3
Conv2d_4: 512 filters of size 3×3 and stride 3

with Dense_0: 80 units
Dropout: rate 0.5
Dense_1: 40 (output) units

This is the model summary from Tensorflow:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 rescaling (Rescaling)       (None, 224, 224, 3)       0         
                                                                 
 conv2d (Conv2D)             (None, 112, 112, 96)      14208     
                                                                 
 batch_normalization (BatchN  (None, 112, 112, 96)     384       
 ormalization)                                                   
                                                                 
 max_pooling2d (MaxPooling2D  (None, 56, 56, 96)       0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, 19, 19, 256)       614656    
                                                                 
 batch_normalization_1 (Batc  (None, 19, 19, 256)      1024      
 hNormalization)                                                 
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 10, 10, 256)      0         
 2D)                                                             
                                                                 
 conv2d_2 (Conv2D)           (None, 4, 4, 512)         1180160   
                                                                 
 batch_normalization_2 (Batc  (None, 4, 4, 512)        2048      
 hNormalization)                                                 
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 2, 2, 512)        0         
 2D)                                                             
                                                                 
 conv2d_3 (Conv2D)           (None, 1, 1, 512)         2359808   
                                                                 
 batch_normalization_3 (Batc  (None, 1, 1, 512)        2048      
 hNormalization)                                                 
                                                                 
 max_pooling2d_3 (MaxPooling  (None, 1, 1, 512)        0         
 2D)                                                             
                                                                 
 conv2d_4 (Conv2D)           (None, 1, 1, 512)         2359808   
                                                                 
 max_pooling2d_4 (MaxPooling  (None, 1, 1, 512)        0         
 2D)                                                             
                                                                 
 flatten (Flatten)           (None, 512)               0         
                                                                 
 dense (Dense)               (None, 80)                41040     
                                                                 
 dropout (Dropout)           (None, 80)                0         
                                                                 
 dense_1 (Dense)             (None, 40)                3240      
                                                                 
=================================================================
Total params: 6,578,424
Trainable params: 6,575,672
Non-trainable params: 2,752
_________________________________________________________________
```

We used the Adam optimizer with a learning rate of 0.001 and a batch size of 8. The model was trained for 15 epochs. We also used the learning rate scheduler from the previous assignment to halve the learning rate every 4 epochs. The beta parameters that control exponential decay of the momentum for the Adam optimizer were left at their default parameters of 0.9 and 0.999 respectively. The loss function we used was *SparseCategoricalCrossentropy()* which automatically applies a softmax when the argument *from_logits* is set to *True*.

5.2 Transfer Learning details

For the Transfer Learning task we froze the convolutional (feature extraction) section of the base model and removed the dense and dropout layers from the end of the network. The network was then appended with 3 new dense layers (30, 30 and 4 units) with dropout applied to the first two at rates of 0.5. We still used the same Adam optimizer but we set the learning rate to 1/10th of its original value which is 0.0001. When training we decreased the batch size to 2 and trained for 20 epochs. Here is the summary of the Transfer Learning network with the frozen layers removed from the summary.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 previous layers are the same
 as in the S-40 CNN model,
 including the following
 flatten layer

 flatten (Flatten)           (None, 512)               0

 dense_2 (Dense)             (None, 30)                15390

 dropout_1 (Dropout)         (None, 30)                0

 dense_3 (Dense)             (None, 30)                930

 dropout_2 (Dropout)         (None, 30)                0

 dense_4 (Dense)             (None, 4)                 124

=================================================================
Total params: 6,550,588
Trainable params: 16,444
Non-trainable params: 6,534,144
_________________________________________________________________
```

5.3 Network on Optical Flow

Since the optical flow network receives inputs of the form (16, 224, 224, 2) we decided to apply 3D convolutions and 3D pooling to also allow the network to pick up patterns along the time dimension. We again used the Adam optimizer with a learning rate of 0.001, batch size of 2 and it was trained for 20 epochs. We modified the network used for the base model since the two models for frames and optical flows in [3] are also almost identical. We did notice that we had very little data to work with for this dataset to train an entire model from scratch. Therefore we removed 2 convolutional layers.
Rescaling (input normalization) $\rightarrow$
Conv3d_0 $\rightarrow$ BatchNormalization $\rightarrow$ MaxPool3d $\rightarrow$
Conv3d_1 $\rightarrow$ BatchNormalization $\rightarrow$ MaxPool3d $\rightarrow$
Conv3d_2 $\rightarrow$ BatchNormalization $\rightarrow$ MaxPool3d $\rightarrow$
Flatten $\rightarrow$ Dense_0 $\rightarrow$ Dropout $\rightarrow$ Dense_1 (output)

with all MaxPool3d having pool size of $3 \times 3 \times 3$ and stride 2 and

with Conv3d_0: 128 filters of size $7 \times 7 \times 7$ and stride 2
Conv3d_1: 256 filters of size $5 \times 5 \times 5$ and stride 3
Conv3d_2: 512 filters of size $3 \times 3 \times 3$ and stride 3

with Dense_0: 100 units
Dropout: rate 0.5

Dense_1: 4 (output) units

Here is the summary of the full Optical Flow network:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling (Rescaling)       (None, 16, 224, 224, 2)   0

 conv3d (Conv3D)             (None, 8, 112, 112, 128)  87936

 batch_normalization (BatchN  (None, 8, 112, 112, 128)  512
 ormalization)

 max_pooling3d (MaxPooling3D  (None, 4, 56, 56, 128)   0
 )

 conv3d_1 (Conv3D)           (None, 2, 19, 19, 256)    4096256

 batch_normalization_1 (Batc  (None, 2, 19, 19, 256)   1024
 hNormalization)

 max_pooling3d_1 (MaxPooling  (None, 1, 10, 10, 256)   0
 3D)

 conv3d_2 (Conv3D)           (None, 1, 4, 4, 512)      3539456

 batch_normalization_2 (Batc  (None, 1, 4, 4, 512)     2048
 hNormalization)

 max_pooling3d_2 (MaxPooling  (None, 1, 2, 2, 512)     0
 3D)

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 100)               204900

 dropout (Dropout)           (None, 100)               0

 dense_1 (Dense)             (None, 4)                 404

=================================================================
Total params: 7,932,536
Trainable params: 7,930,744
Non-trainable params: 1,792
_________________________________________________________________
```

6 Two-Stream network outline

To fuse the two models trained on TV-HI we froze all of its layers. From the network trained on the frames we removed 2 dense layers and one dropout. We kept the first dense and dropout layer from the model since it was fine-tuned on the TV-HI dataset. From the optical flow network we removed all of its dense and dropout layers leaving only the convolutional section (feature extraction). We took the outputs of these models up until these points and concatenated and flattened them into a single 1-dimensional tensor which was passed on to the final section of the network. This section consists of 3 new Dense layers (128, 64, 4 units respectively) and 2 dropout layers (rate 0.5). For each sample we use a dictionary to map the inputs to their respective input layers. We kept the loss function and the optimiser the same with a learning rate of 0.0001. We used a batch size of 1 and trained for 20 epochs.
Here is the summary of the final two stream network with the convolutional sections of the network taken out of the summary for clarity since these weren't altered:

```
Model: "model_1"
__________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================
 frame_input (InputLayer)       [(None, 224, 224, 3  0           []
                                )]

 flow_input (InputLayer)        [(None, 16, 224, 22  0           []
                                4, 2)]

Both models are the same as
before. After the inputs
both models run the same
as before up to and
including the flatten layers

The flattened layer for the
frames network followed by
a pre-trained dense layer
with dropout (both frozen)


 flatten (Flatten)              (None, 512)          0            ['max_pooling2d_4
    [0][0]']

 dense_2 (Dense)                (None, 30)           15390        ['flatten[0][0]']

 dropout_1 (Dropout)            (None, 30)           0            ['dense_2[0][0]']

The flattened layer for the
flow network without any of
the previously trained dense
layers

 flatten_1 (Flatten)            (None, 2048)         0            ['max_pooling3d_2
    [0][0]']

Fusing the two networks
All of the following layers
are trainable

 concatenate (Concatenate)      (None, 2078)         0            ['dropout_1[0][0]',
                                                                   'flatten_1[0][0]']

 flatten_2 (Flatten)            (None, 2078)         0            ['concatenate
    [0][0]']

 dense_7 (Dense)                (None, 128)          266112       ['flatten_2[0][0]']

 dropout_4 (Dropout)            (None, 128)          0            ['dense_7[0][0]']

 dense_8 (Dense)                (None, 64)           8256         ['dropout_4[0][0]']

 dropout_5 (Dropout)            (None, 64)           0            ['dense_8[0][0]']

 dense_9 (Dense)                (None, 4)            260          ['dropout_5[0][0]']

==============================================================================
Total params: 14,551,394
Trainable params: 274,628
Non-trainable params: 14,276,766
______________________________________________________________________________
```

[3]

7 Results

7.1 Stanford-40

After training and validating the Stanford-40 CNN, we noticed that the model rapidly overfits on the training data after the validation accuracy converges around 17-21%. This can be seen in Figure 1. We decided to limit the total amount of epochs to 15, and determined the moment of convergence to be around epoch 11. The training/validation accuracy and loss recorded in Table 1 predict a reasonable improvement over Stanford 40's 2.5% random baseline accuracy. We then recorded the model's performance on the test set in Table 2 and found that the final accuracy and loss scores were consistent with the validation scores.

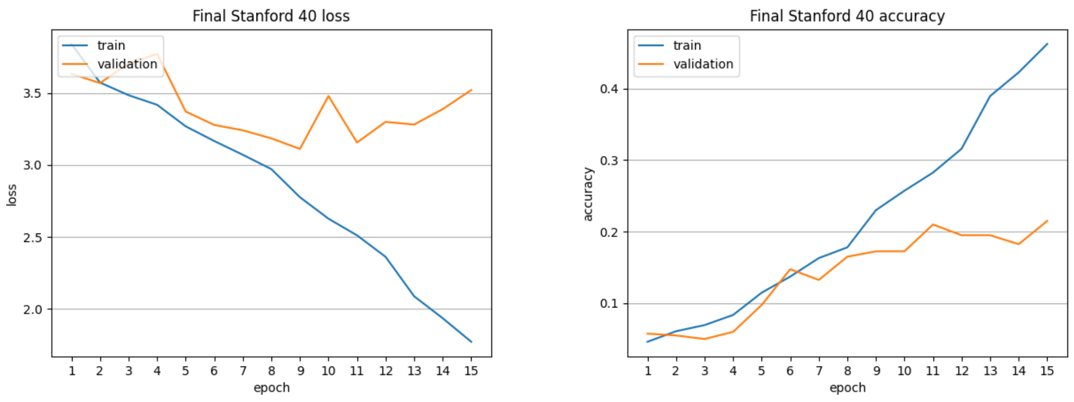

Figure 1: CNN training/validation performance over 15 epochs on Stanford 40 dataset.

7.2 TV Human Interactions

7.3 Transfer Learning Model

We used the stored weights from the Stanford-40 CNN and used them for fine-tuning a network to predict the 4 TV-HI action classes. The smaller amount of data resulted in the erratic training accuracy/loss scores seen in Figure 2. A further consequence of the small validation dataset is the fact that the 10 samples lead to increments of 0.10 in the validation accuracy score. These big steps made it hard to figure out where the model truly converged. In order to offset this, we decided to train and validate the model for 20 epochs instead of 15. Despite training for longer, the erratic accuracy graph was still not very useful. As such, we mainly used the training/validation loss to determine that convergence happens around epoch 8. Because the training/validation accuracy scores recorded in Table 1 happen to dip around this epoch, the final testing score of 34% in Table 2 is unsurprisingly a bit higher.

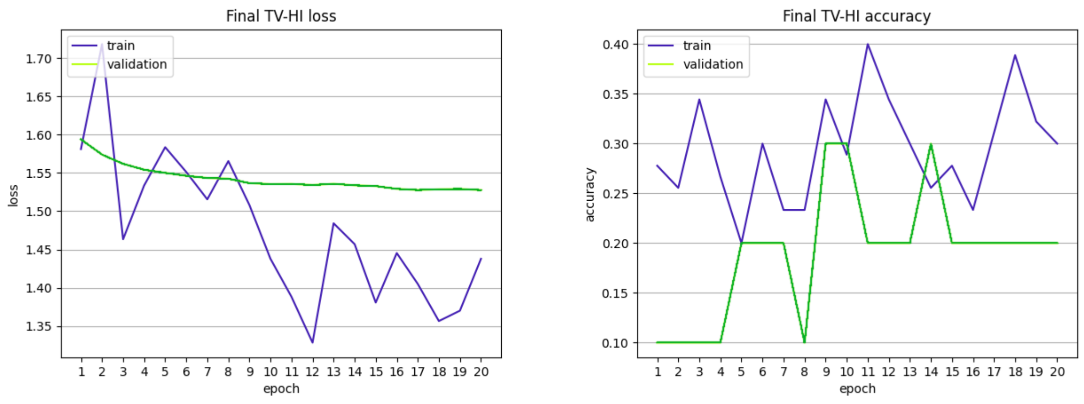

Figure 2: Finetuned CNN training/validation performance over 20 epochs on frames taken from the TV-HI dataset.

7.4 Optical Flow Model

As seen in Figure 3, the optical flows network has a training loss that goes steadily down while the validation accuracy suffers from the same problem as before. It appears that the model is overfitting since the validation loss is not going down at the same rate as the training loss. The training accuracy goes up to an almost perfect score while the validation accuracy fluctuates around 40%. We determined the 16th epoch to be the moment of convergence given that the loss curve somewhat slopes horizontally after this point. Though the training and validation accuracy scores are suspiciously high at this point, we found the testing accuracy score of 39% to be in line with the fluctuating validation.

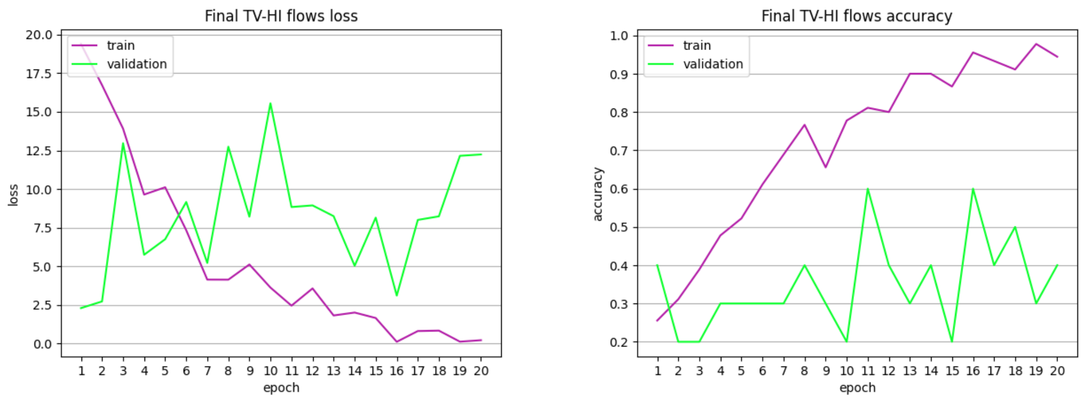

Figure 3: CNN training/validation performance over 20 epochs on stacked optical flow calculated for 16 frames per video from the TV-HI dataset.

7.5 Two-Stream Model

The final two stream network results in Figure 4 show a steady decline in training loss during training while the validation loss also appears to be trending downwards. The model seems to be overfitting a little bit as the train and validation loss crossover happens around epoch 8. However, it's not overfitting very much as there is no large gap between the training and validation loss.
The training accuracy appears to steadily climb on the training set while again trending upwards for the validation set. Of course, since the validation set only contains 10 samples it is a bit unreliable which explains the fluctuations. Overall it looks like this model has somewhat converged as the training and validation loss are close together and are not really going down much more. The final accuracy scores recorded in Table 1 don't seem in line with the final accuracy for the validation, which actually fluctuates more around 50%. This is in line with the testing accuracy of 40% recorded in Table2.

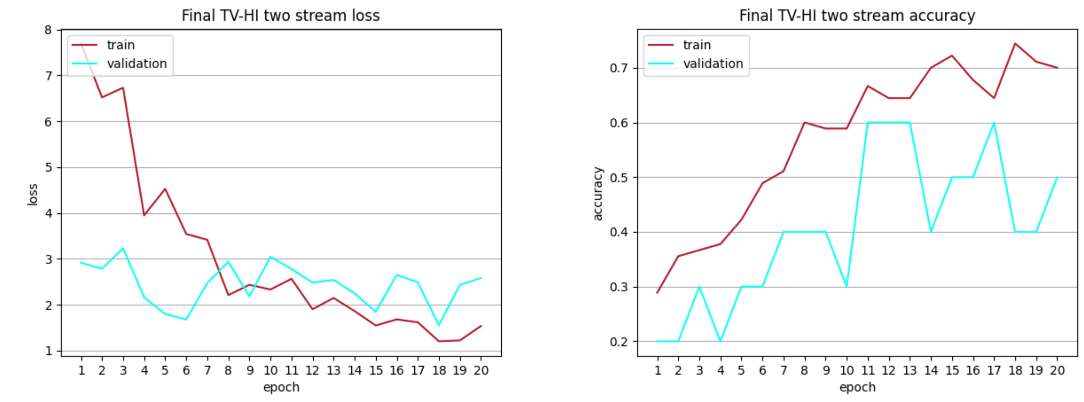

Figure 4: Two-stream CNN training/validation performance over 20 epochs on frames and stacked optical flow from the TV-HI dataset.

8 Conclusions

We observe that the addition of optical flow information to the model fine-tuned on TV-HI created a lot more stability in the network. The validation loss appeared to stay down a lot more consistently in the two stream network which combines frames and optical flow information compared to the two separate models on frame and optical flow information. Although this is not really apparent from looking at the final test accuracy which only improves 1% over the optical flow model, the test loss is significantly lower in the two stream network compared to the optical flow network. We think that the addition of the 16 frames of optical flow information does aid the network's performance at least to some extent.

	train acc	**val acc**	**train loss**	**val loss**	**Epoch**
Stanford 40	0.28	0.21	2.51	3.20	11
TV-HI frame	0.23	0.10	1.57	1.54	8
TV-HI flow	0.95	0.60	0.22	3.24	16
TV-HI two-stream	0.65	0.60	1.93	2.49	12

Table 1: Top 1 train/validation accuracy and loss of each model, and the epochs at which they were achieved.

	Test acc	**Test loss**	**Model size (GB)**
Stanford 40	0.17	3.11	0.075
TV-HI frame	0.34	1.39	0.025
TV-HI flow	0.39	14.20	0.091
TV-HI two-stream	0.40	5.77	0.058

Table 2: Final testing accuracy, loss, and sizes of the weights of each model.

Before considering future improvements in the model architectures, a good first step would be data augmentation, as the small overall size of the TV-HI dataset is definitely a bottleneck for each of the TV-HI models' ability to generalize correctly. This bottleneck made the evaluation of model performance very difficult.

9 Weights

All model weights are available here.

10 Choice Tasks

- For our choice tasks we chose to let go of perfectionism.

References

[1] Geoffrey E. Hinton et al. *Improving neural networks by preventing co-adaptation of feature detectors.* 2012. DOI: `10.48550/ARXIV.1207.0580`. URL: `https://arxiv.org/abs/1207.0580`.

[2] Alonso Patron et al. "High Five: Recognising human interactions in TV shows". In: *Proceedings of the British Machine Vision Conference.* doi:10.5244/C.24.50. BMVA Press, 2010, pp. 50.1–50.11. ISBN: 1-901725-40-5.

[3] Karen Simonyan and Andrew Zisserman. *Two-Stream Convolutional Networks for Action Recognition in Videos.* 2014. DOI: `10.48550/ARXIV.1406.2199`. URL: `https://arxiv.org/abs/1406.2199`.

[4] Bangpeng Yao et al. "Human action recognition by learning bases of action attributes and parts". In: *2011 International conference on computer vision.* IEEE. 2011, pp. 1331–1338.