Computer Version Assignment 5

Yunming Hui(5334977)
Graduate School of Natural Sciences
Utrecht University

Yuqi Liu(5383986)
Graduate School of Natural Sciences
Utrecht University

This assignment, is an attempt to recognise the actions of people in images, videos through CNN networks. Specifically, with images, video as data. Action labels are specified in advance. Predicting the label of the action shown by an image. The assignment is based on the Windows operating system, Keras 2.7.0 and python 3.9.

I. Basic tasks

A. Description of the model framework and selection of parameters

In this assignment we trained a total of four models, and below we describe in detail the data, model architecture, and parameter selection for each model.

1) the model architecture of Stanford 40 – Frames: First of all the first model is called “Stanford 40 – Frames”. It is based on a set of colourful action pictures from Stanford University. There are 9,532 images in total, in 40 classes. We chose 4000 of these as the test set and 3532 as the training and validation set (15% of the 3532 for the validation set). Additional, we change the size of all images to (112,112,3).
Obviously three thousand images is not enough data for a total of 40 labels. We may consider data augmentation of the dataset(This is described in section 2). In addition to this, we also try to use deep neural networks to learn as many features as possible. But gradient disappearance affects the performance of deep neural networks. This problem may be improved by the residual network proposed according to He et al [1]. So we use a residual network with a total of 50 convolutional layers, called "ResNet50".

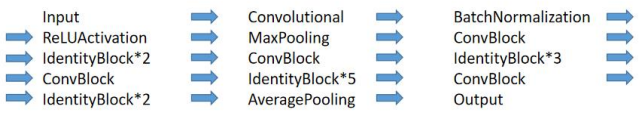

Fig. 1: the architecture of ResNet50

The architecture of "ResNet50" as shown in the figure1. There are two kind of layer blocks, the convolution block and identity block. A convolution block is a block consisting of three layers of convolution and one layer of convolution followed by an "add" operation at the end. Where the kernel is all 3*3 in size and the number is the same. A similar one, identity block, consists of three convolutional layers(also 3*3 kernel with same number of filters) and the input at the beginning being "added" at the end. In identity block, because we end up adding the convolution result to the input, when gradient disappearance occurs, the output of the block is actually the input, so it is equivalent to skipping these convolution layers. The residual network uses this approach to improve the gradient disappearance problem. At design time, a convolutional block is formed into a stage with several identity blocks, and as the network layers go deeper, each stage has twice as many filters in the convolutional layers as the previous stage. To avoid too many parameters, the number of filters in the first stage is 32 and the last one is 256. Apart from this, other hyper parameters are selected as shown in the tableI.

TABLE I: ResNet50 Hyperparameter values

Hyperparameter	value
Batch size	128
epoch	25
Activation function	relu
Output Activation function	softmax
Learning rate	0.0001
Optimizer	Adam
Loss function	categorical crossentropy

The code is in the “Standford40.py” file. The convolution block is implemented in the code with function "convolutional_block", the identity block with function "identity_block" and finally ResNet50 with "ResNet50". The model achieved an accuracy rate of 10.81%, with other specific performances shown in the second subsection. Links to the models are in the "Supplementary material" section.

2) the model architecture of TV-HI – Frames: The next model is used to make predictions on the TV-HI data. TV-HI has a total of 200 videos (no negatives), with a total of four labels. Half of these are used as the test set and the other half as the training, validation set. We take the middle frame of each video to form the image data with the same size as previous model(The analysis of the effect of taking frames at different times is described in section 2). So we have to map 200 images into 4 labels. As with the previous dataset, there is not enough data compared to the number of labels, so the framework of the previous model can be used for this dataset. And we experimented with data augmentation to expand the data, the effects of which are described in Section 2. However, as there were only 200 data, we considered transforming learning directly, transferring the weights from the previous model to this one. And set the learning rate to the original 0.1 for fine-tuning. Other than that, the rest of the

hyper parameters, we only changed the batchsize to 4. The code is in the "TVHIFrames.py" file. The final accuracy rate on the test set was 31%. More details analysis follows.

3) the model architecture of TV-HI – Optical Flow: For this model we used the same original dataset as for the second model, a total of 200 videos. The difference is that instead of extracting the image from the video, we generate a optical flow of the video. Since the sparse optical flow has to determine the key point first, and in practice the key point does not locate the main action position very well (the key point is always located to the background and not to the limb, e.g. the hand), a dense optical flow is considered, where the set analyses its movement for each pixel.

Specifically, we take a total of 16 frames around the middle of the video. Every 2 frames are transformed into a 112*112 grayscale image by obtaining the optical flow map using OpenCV's "calcOpticalFlowFarneback" method. So a total of 15 optical flow grayscale images were obtained. We combine them together to form one (112*112*15). So in the end, we have an input size of (112*112*15). The CNN framework we still use the previous ResNet50, the hyper parameters, and the model framework remain the same as the second model. But we retrain to find weights from scratch without using the previous weights.

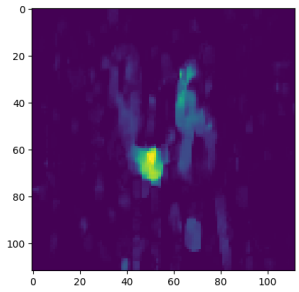

Fig. 2: handshake optical flow grayscale images

One of the handshake optical flow grayscale figure2 is shown, and you can see that the part of the handshake between the two people is well captured. The results are also better as the dense optical flow map reflects the action profile well, with a final accuracy of 37.03%. A more detailed analysis will follow. Links to their model weights can be found in the "Supplementary material" section.

4) the model architecture of TV-HI – Two-Stream: For the last model, we tried to combine model 2 and model 3 to generate a two-stream CNN model. Two of the inputs are the image extracted from the video (112*112*3) and the collection of optical streams extracted from the video (112*112*15). The specific fusion method is shown in the figure3. Where the fusion is done we use keras' "concatenate" method, which means that the two inputs are concatenate in terms of thickness. And it is worth noting that we fused the optical flow information twice. Before each fusion, we first compressed the two input data to only one thickness using 1X1 convolution. Except for the last time for the optical flow information. Since optical flow data captures movement well, we take this approach so that optical flow data makes up a large proportion of the fused data. For the two ResNet50 networks, we loaded the weight information of model 1,3 and still set the learning rate to an initial 0.1 for fine-tuning. The other hyperparameters are consistent with models 2,3. The final accuracy of this model was 41%. Links to their model weights can be found in the "Supplementary material" section.

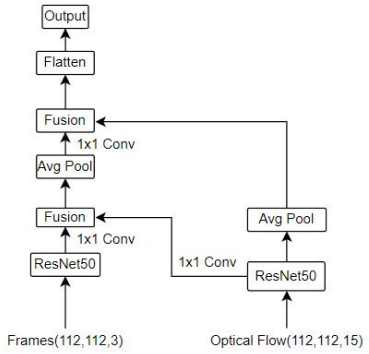

Fig. 3: the framework of two-stream CNN

B. Experimental results

This section shows the detailed results of the four models on the test set and their training process.

The performance of the four models on the test set is shown in the tableII. One of them, the two-stream CNN model, performed the best. Of course in terms of model size, the two-stream CNN model is significantly larger. We will analyse this later in conjunction with the training process.

TABLE II: Test top-1 accuracy/Loss value for all four models

Dataset	Model	Top-1 acc.	Top-1 loss	Model size (MB)
Stanford40	Frames	10.81%	6.8428	14.8
TV-HI	Frames	31.00%	7.8169	14.8
TV-HI	Optical flow	37.03%	3.7222	14.3
TV-HI	Two-stream	41.00%	2.2226	28.4

The images of the accuracy and loss functions of the four models on the training set are shown in figure4 and figure5 below respectively. Can reflect the training process. Where the blue line in each image represents the training set and the yellow line represents the validation set. From the figure we can see that the optical flow based model and the two-stream CNN model perform relatively well on the training and validation sets, and the first two models do not perform well. Some degree of overfitting occurred in all four models, suggesting that there was too little data and too many different types of labels. And the first two models are even more severe. We analyse this specifically in the next subsection.

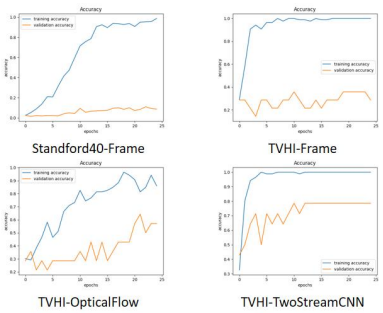

Fig. 4: the accuracy on training/validation set for four models

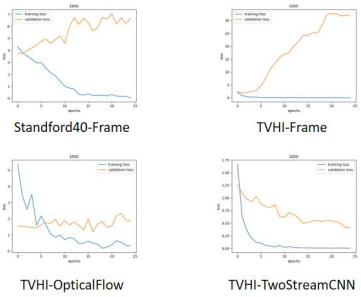

Fig. 5: the loss value on training/validation set for four models

C. Experimental analysis

We begin by analysing the performance of the ResNet50 network. The results from the test set show that the accuracy on the Standford40 dataset is around 10%, which is normal performance. On the TV-HI dataset, the accuracy rate was 31% with frames as input and 37% with optical flow as input, with the results performing better overall than the data provider's 32.7%. So ResNet50 performed well. And from the training process, the results of all four models on the training set eventually converge, so the complexity of this neural network is able to match this problem. However, the results are not good because of the data problem.

For the advantages of the optical flow, as mentioned earlier, it captures the position of the action very well. Its advantages are also reflected in the results. The optical flow achieves an accuracy of 37%, which is significantly higher than the first two models. And, from the training process, the accuracy of the validation set of the first two models barely increased, but the accuracy of the validation set of the optical flow model increased significantly, indicating that useful features were indeed learned. And also in terms of loss, the loss of the first two models increased with epoch, but not the optical flow, indicating that no spurious features were learned.

Similarly, we can find that the two-stream CNN model performs best. Its accuracy is higher than that of the model based only on optical flow, and the difference in performance between the validation and training sets during its training is also better than that of the previous three models. We believe that this is due to the fact that the two-stream CNN model fuses the optical flow twice to make the best possible use of it. And frames as input bring in more feature information in some extent.

Finally we come to the issue of low accuracy. As mentioned earlier, both datasets have the problem of having too little data relative to the number of labels. So often the features to solve this problem cannot be learned from the limited training data. In addition to the above responses, we should think about using data augmentation methods to solve the problem. This will be covered in the next section.

II. Choice Tasks

A. Data augmentation for Standford-40 and TV-HI

To address the problem of insufficient data, we attempted data augmentation on both datasets. We increase the data mainly based on colour changes. This means adding or subtracting a small random value from the RGB data for all three channels. In addition to this, we also tried to generate images using horizontal flips (i.e. to obtain mirror images). This is achieved by using the "ImageDataGenerator" method of keras. The results of the data augmentation applied to models 1 and 2 are shown in figure6. The two figures show the training process for each.

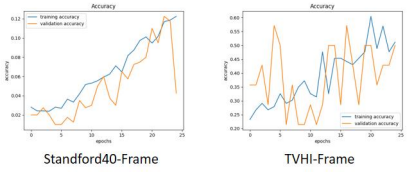

Fig. 6: training process with data augmentation

From the figure we can see that there is no longer a huge difference between the performance of the training and validation sets for the two frames-based models after using data augmentation. Of course the validation set fluctuates greatly because the amount of TV-HI data is so small that the key frames of the images are not always captured. However, there was no significant improvement in the accuracy of the test set. For the Standford40 dataset, the reasons for this were insufficient training and too many labels. For the TV-HI dataset, the problem should be that only the intermediate frames of the video are acquired, not the key frames that show the action, and also that the amount of data is too small.

B. Create a Cyclical Learning rate schedule

We design a cyclic learning rate schedule. Specifically, every five epochs, the first epoch has the same learning rate, the second and third epochs multiply the learning rate by 2, and the fourth and fifth epochs multiply the learning rate by 0.5. The results of the loss function for the fixed and cyclic learning rates in the TVHI-Frames model using data augmentation (model 2) are shown in the figure7. We can find that the cyclic learning rate, because it keeps changing the learning

rate, makes the accuracy of the validation set more similar to the change in accuracy of the training set. On the other hand, with a constant learning rate, the results of both do not increase or decrease together. This is because changing the learning rate is in some ways a way to jump out of the local optimum. The results are closer to the true value. However, in terms of the results from the test set, there was no significant improvement by applying the cyclic learning rate. This is because there are serious problems with the dataset itself, as analysed earlier.

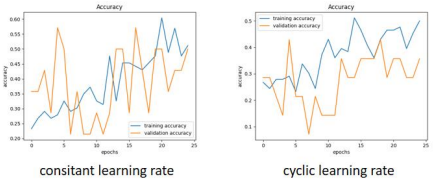

Fig. 7: the accuracy of training/validation set for constant and cyclic learning rate

C. Use a kernel regularizer for your convolutions

We know that the purpose of applying penalty coefficients to the convolution layer is to regularise and prevent over-fitting from occurring. The figure**??** shows the training process with and without L2 penalty in terms of accuracy after applying data augmentation. We continued our analysis using the TVHI-Frames model for the application of the regularization penalty function. Similar to the results of applying the cyclic learning rate, the trend of the results of the validation and training sets are similar after applying the L2 penalty. This means that the overfitting problem is improved to some extent. And, there continues to be no significant improvement in accuracy on the test set.

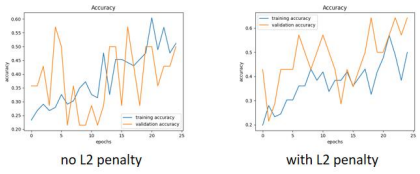

Fig. 8: applied L2 penalty on convolution layers

D. Use 1x1 Convolutions to connect activations between the two branches

In our model of two-stream CNNs, we use 1x1 kernel to compress the data at each fusion in order to reduce the number of parameters. And, as mentioned above, we did not compress the optical flow data during the second fusion, thus making it the majority of the fused data. Specifically, the third dimension of the data input at the flatten layer of the two-stream model is 16, since optical streams account for 15, and we compress the non-optical stream data to 1 by 1x1 kernel. In addition to this, we also apply a 1x1 convolutional kernel at the end of each block of ResNet50 to compress the number of parameters.
As a result, the final flatten layer has 12,288 parameters if we do not use a 1x1 convolutional kernel for each fusion, and 4,128 parameters if a convolutional kernel is used as shown previously. Similarly, if the convolution kernel is not used at the end of each block, the parameters increase even more.
In terms of test set accuracy, the use of 1x1 convolutional kernels to compress the data increases the weight of optical flow data and the accuracy of model4 reaches 41%, as shown previously. Without compressing the data, the accuracy would only be around 31%, which is around the accuracy of model2. So in other words, we are also increasing the influence capability of well performing features with the 1x1 convolution kernel.

E. Analysis of the effect of extracting different frames of video on the results of TV-HI-Frames

We explore the changes in the accuracy of the TVHI-Frames model on the test set by taking different times of the video frame to form the input data. We go to 0%, 10%, and up to 90% of the time points of the video to form ten different sets of inputs and use model2 to make predictions. The results are shown in the figure9.

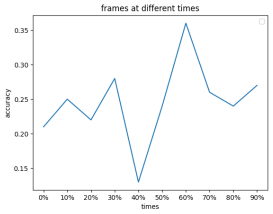

Fig. 9: the accuracy for TVHI-Frames medel with frames in different times

Firstly, as the accuracy of the model is subject to fluctuations, we conducted several experiments and the results were not the same for each trial. The graph above shows the results of one of the experiments. However, we can still find that, overall, the accuracy of the images at different times hovers around 25%. But the accuracy of taking images from the 50-60% time node is relatively high. In this trial the 60% time node achieved an accuracy of 36%. This is due to the fact that generally the action occurs fully in the 50%-60% interval of the video. Of course we need to stress again that the results of this data are very volatile, so only a rough guess can be made.

III. Supplement material

There is the link to models: https://github.com/andyzxq/ComputerVisionAssignment5.git

References

[1] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). https://doi.org/10.1109/cvpr.2016.90