Lab 2: Video classification

María Barroso Honrubia
Gloria del Valle Cano
May 15, 2022

I. INTRODUCTION

This report covers the work done for the video classification assignment. We have conducted a research on several experiments of video classification. Starting from a set of videos $\mathscr{X}$ labeled by a set of actions $\mathscr{Y}$, we aim to learn a function $f : \mathscr{X} \to \mathscr{Y}$.

Thus, we test three different methods, a fixed class one, a random class one and a CNN-based model, which is the one that outperfoms the other two baseline models.

Finally, we study the CNN-based model performance with several sets of classes, analyzing the plots of the learning curves and showing evidence that the accuracy score decreases as well as the number of classes increases.

II. METHOD

In this section the methods employed in the experiments are briefly described. They can be run using the Jupyter Notebook `Video_Classification.ipynb`

A. Random mode classifier

The first method implemented is a random classification algorithm, which assign a random class to each video frame. Therefore, the random classifier is a uniform random variable over all classes.

B. Fixed mode classifier

The second method always assigns to each video frame the same class so that the accuracy achieved matches the proportion of samples in each corresponding class.

C. Frame by Frame CNN classifier

The last method implemented is a frame-by-frame classifier using a pre-trained version of the InceptionV3 architecture and Imagenet weights.

A global average pooling is added to the output of the base model. Then, a fully connected layer with 1024 units is added on top of the pre-trained network, in addition to another one with as many units as the number of classes of the problem and a softmax activation function.

The model is built by using a simple fine-tune in two steps. First, we briefly fine-tune the top dense layers in an attempt to retain as much of the previous learning as possible. This is possible by training only the top layers for 10 epochs using the RMSProp optimizer. Next, we retrain properly the mid-top dense layers to fit the model to our particular problem. Specifically, the last 172 layers of the network are unfrozen and the model is trained using the SGD optimizer with a reduced learning rate.

III. IMPLEMENTATION

A. Random and fixed mode classifiers

We compare the performance of the random and fixed classifiers by evaluating them on all frames using only 5 classes. The evaluation metric considered is the *accuracy*. For the random classifier, 500 runs are performed in order to achieve more stable results.

The random and fixed algorithm are implemented in the script `random_vs_fixed.py` though a function called `random_vs_fixed()`, which returns the precision obtained for each method.

B. CNN Frame by Frame mode classifiers

We build the CNN model on four different data versions (according to the number of classes used) and we compare the results to those of the random and fixed classifiers.

The code to create the base pre-trained model and make the transfer-learning workflow is available in the file `train_cnn.py`. Once the model has been trained, the evaluation and the representation of the obtained accuracy are performed in the files `validate_cnn.py` and `plot_train_cnnlog.py`, respectively.

IV. DATA

Data used in this assignment consist of a subset of the UCF-101 action recognition dataset. The videos are preprocessed using the provided Python script `data.py` `processor.py`. For all the experiments performed, 5 frames per video are extracted. The image frames can be accessed via the `DataSet` class implemented in `data.py`.

In the experiments with CNN Frame by Frame mode, different number of video classes (5, 10, 15 and 20) are used to run the models. Moreover, a train/validation splot of roughly 70/30 is employed.

V. RESULTS AND ANALYSIS

A. Random vs. fixed mode

Results for the random and fixed mode classifiers are shown on Table I, which contains one column for class: Random (R), ApplyEyeMakeup (AEM), ApplyLipstick (AL), Archery (A), BabyCrawling (BC) and BalanceBeam (BB).

	R	AEM	AL	A	BC	BB
Acc (%)	20.03	22.52	17.70	22.52	20.50	16.77

TABLE I: Accuracy for random and fixed mode classifiers

Looking at the results, two considerations about each type of classifier can be made. On the one hand, we see that the random classifier achieves an accuracy close to 20% after 500 runs. This value coincides with the expected value of the uniform random variable when setting the number of classes at 5. On the other hand, observing the accuracy obtained by the fixed classifier, we can conclude that the dataset is balanced on these classes.

Finally, the accuracies obtained for both types of classifiers are summarized in Figure 1. Although such methods are very trivial, they serve as a baseline for comparing more complicated models, which will hopefully improve the random and fixed classifier.

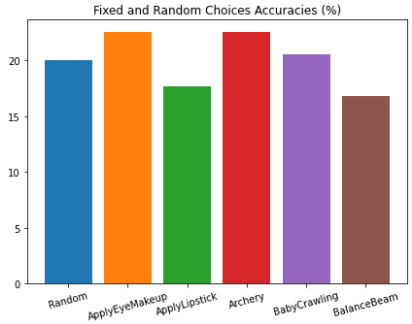


Fig. 1: Accuracy of random and fixed mode classifiers

B. Random vs. fixed vs. CNN (5 classes) mode

The training and validation accuracy curves of the frame-by-frame CNN model with data from 5 classes can be seen in Figure 3a.

In this experiment, the frame-by-frame CNN model was trained with the data from 5 classes following the procedure outlined in Section III. This model far outperforms the random and fixed mode classifiers, achieving an accuracy of better than 90% on both training and validation sets. We see that the early stopping point turns out to be 12 epochs. The corresponding accuracy of training and validation can be seen in Figure 3a, where it's clear to see that this model outperforms both random and fixed mode classifiers, obtaining a train and validation accuracy of 93.75%.

Although this model obtains valuable results, it takes about 30 minutes to execute on Google Colab Pro in overall, unlike the baseline models that its computational cost seems to be highly reduced.

C. CNN classifier with different number of classes

Another experiment is performed, increasing the number of classes little by little from 5 to 20 classes. In detail we select 5, 10, 15 and 20 classes. The complete accuracy results can be seen on Table II, which shows that the problem becomes even much more computationally expensive increasing the number of classes, besides the fact that the resulting accuracy is getting lower and lower. The complete results are illustrated on Figure 3 and Figure 4.

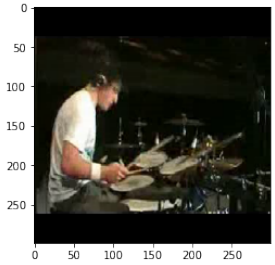
	Train Accuracy	Val. Accuracy	Train Loss	Val. Loss
UCF5	0.9375	0.9375	0.2366	0.1914
UCF10	0.8500	0.9375	0.5008	0.2735
UCF15	0.8875	0.8500	0.5732	0.3725
UCF20	0.7625	0.8375	0.9883	0.5649

TABLE II: Final accuracy and loss for every experiment

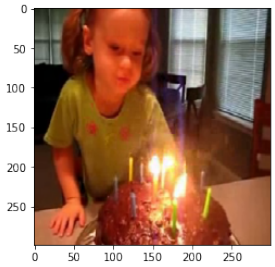
It is worth to point out that the validation accuracy is generally higher than the training one, so we can assure that the model has learned to generalize well new unseen actions.

We also include in Figure 4 the evolution of training loss for all classes proved in this experiment.

Finally, we can check examples well classified and misclassified. We see that generally actions are well classified with a very high score, and actions wrongly classified are very similar to its predicted result. We illustrate an example on Figure 2.



(a) Misclassified with a score of 95%



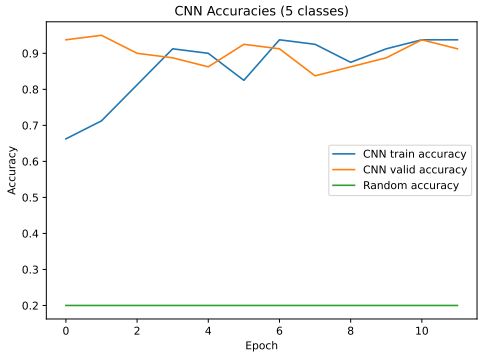
(b) Well classified with a score of 24%

Fig. 2: Well and misclassified examples for BlowingCandles class

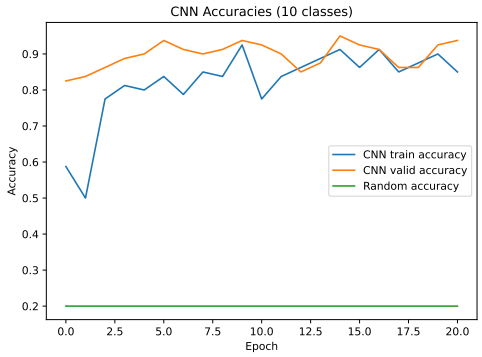
VI. CONCLUSIONS

In this practical assignment we have seen that a frame-by-frame CNN model vastly outperforms the baseline models, random and fixed mode, even more when the problem becomes more complex with a higher number of classes. Although, it is much more worthwhile at some point because of the much more accuracy is gained.

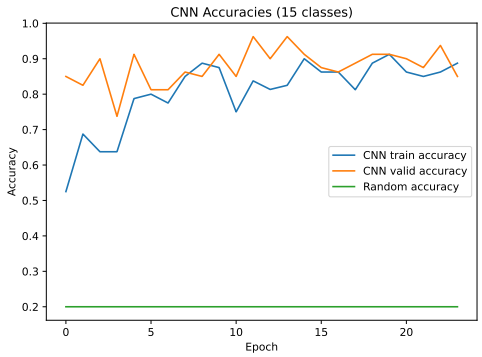
Conversely, it seems that this kind of models perform properly when we have a limited number of classes, so we have to take into account that when we have a much higher number of classes, we should consult other action recognition state-of-art models [1], [2].



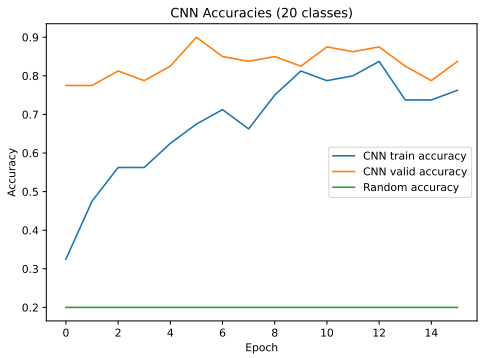
(a) Accuracy with $K = 5$ (CNN classifier)



(b) Accuracy with $K = 10$ (CNN classifier)

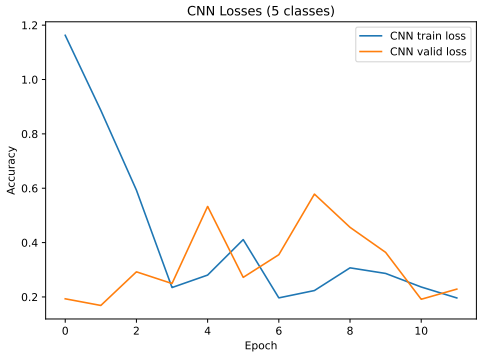


(c) Accuracy with $K = 15$ (CNN classifier)

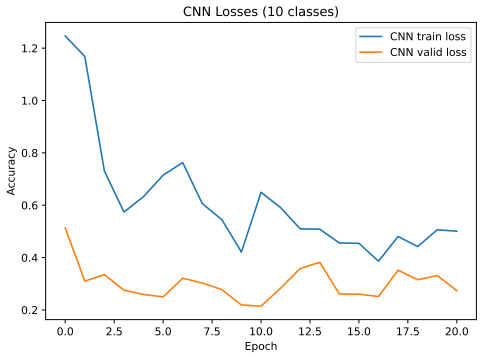


(d) Accuracy with $K = 20$ (CNN classifier)

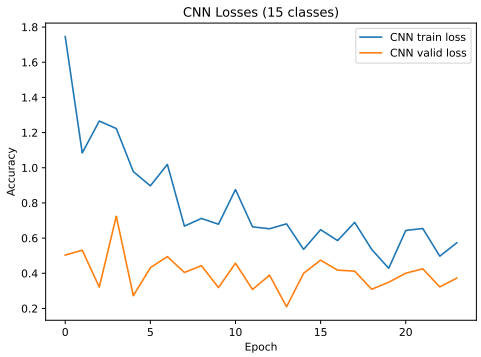
Fig. 3: CNN-based model accuracies per experiment with a set of classes



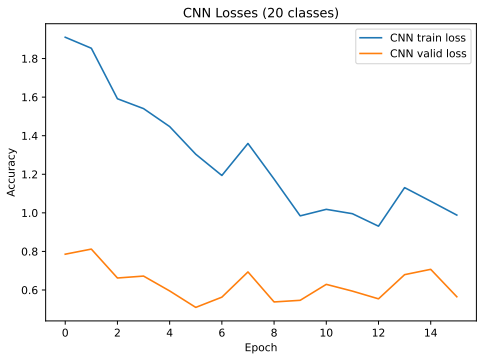
(a) Loss with $K = 5$ (CNN classifier)



(b) Loss with $K = 10$ (CNN classifier)



(c) Loss with $K = 15$ (CNN classifier)



(d) Loss with $K = 20$ (CNN classifier)

Fig. 4: CNN-based model loss functions per experiment with a set of classes

VII. TIME LOG

This assignment has been performed during the last week of this course, training and validating all experiments, and last but not least, plotting all results and making conclusions. We attach the breakdown of tasks with their respective duration:

- Theoretical study: 1.5 hours.
- Completing the models: 1 hour.
- Executing and working on Colab: 8 hours.
- Testing: 1 hour.
- Writing the report: 2 hours.

REFERENCES

[1] Seyed Mostafa Hejazi, Charith Abhayaratne, *Handcrafted localized phase features for human action recognition*, Image and Vision Computing, Volume 123, 2022, 104465, ISSN 0262-8856, https://doi.org/10.1016/j.imavis.2022.104465

[2] Gowda, Shreyank N and Rohrbach, Marcus and Sevilla-Lara, Laura, *SMART Frame Selection for Action Recognition*, https://arxiv.org/abs/2012.10671