INFOMCV Assignment 5

Action Recognition with Automatic Model Search

Deadline: Sunday April 14, 2019, 23:00

Problem introduction

The recognition of human interactions from videos remain a challenging task despite the research progress that has been made over the last few years. The introduction of Convolutional Neural Networks for both image and video data is currently considered a de-facto method for many vision related tasks including that of action recognition. Most of the state-of-the-art projects use CNNs. Recently, because of the complexity of Neural Networks, methods have been developed to automatically search the optimal architecture based on a selected dataset. Neural Architecture Search [1] is currently the method used for producing the top-performing networks in baseline datasets such as MNIST [2] and CIFAR-10 [3]. The main problem however, is that in order to train the network, an excessive amount of computational power is required.

This assignment is split into three tasks. In Task A, you will create a Neural Network architecture capable of classifying human actions of the Stanford-40 dataset [4]. In Task B, you will work towards optimising your chosen architecture by using methods as presented in the lectures and practicals (e.g. Transfer Learning, weight decay, custom learning rates etc.). In Task C, you will create a simple algorithm for finding the optimal structure. There is a lot of freedom in how you approach each task. Therefore, there is no bonus for this assignment.

We will be using the Keras Python package with a Tensorflow backend. You will have to implement the requested functionality from scratch, meaning that you are not allowed to use third party code or rely on libraries with specific functionality. General libraries such as numpy are allowed. If you are in doubt if you are allowed to use a specific library, consult with Alex Stergiou.

The reflection on what you have done and how it explains the results you obtain is an important part of this assignment. So in addition to implementing the functionality and producing the results, we ask you to explain your results.

For this assignment, students or teams that do not have a GPU available can choose to implement task A and tasks B.1 and B.2 with a fully-connected neural network instead. If you choose to implement such a model instead, let the coordinator (Alex Stergiou) know.

I. TASK A: Classification on Stanford-40 (30 points)

The Stanford 40 Action Dataset contains images of humans performing 40 actions. There are 9532 images in total with 180-300 images per action class. Your task is to create a CNN or NN architecture limited to 8 layers and one additional fully-connected layer for your class predictions. The model chosen can include any type of layer (Convolutional, Pooling, Dropout, BatchNorm etc.) that were described in the course.

Essential experiments that should be recorded for obtaining full marks are:

1. Motivations for the choices in designing your network. Suboptimal choices can lead to points deducted. (**10 points**)
2. A visualisation of the chosen network structure chosen using the Keras plot_model function making sure that the shapes of the tensors at each layer are shown (with the 'show_shapes' argument in the function). (**5 points**)
3. A graph of the recorded training and testing accuracies at each epoch alongside a .csv file with the values produced by the Keras callback function. (**5 points**)
4. Conclusions that can be made based on the results obtained. E.g., which classes can be classified well and which ones are harder? (**10 points**)

II. TASK B: Using additional methods (40 points)

You will probably notice that there is a lot of room for improvement in the network architecture that you have created. As by now you should have an idea about the problems of overfitting and underfitting, the main focus of this task should be improving (as best as possible) the current model. Therefore, the three tests required here are:

1. Using a custom learning rate. Similarly to the practicals, but in this case you are asked to implement an exponential learning rate decay function. The formula of the function to be used is up to you. But do keep in mind that the quicker the learning rate decreases, the slower the weight updates will be. As in the previous task, you should record both the training and testing accuracies and losses, plot them and also save them in a .csv file. In addition, you should also include a graph showing the learning rate reduction formula that you have chosen. (**5 points**)
2. Weight decay. This can be simply done by including a Regulariser in the 'kernel_regularizer' argument in the Convolution/Dense layer function. Save your rates again in a .csv file and plot a graph. (**5 points**)

3. Transfer Learning. At the last step you will be using a pre-trained model from 'keras.applications'. You are free to use whichever architecture you would like, but do make sure that (a) all layers except your final fully connected layer have their trainable parameter set to False, and (b) there is no mismatch between the allowed sizes of the images. Make sure to save the epoch that the model performed the best, in a `.h5` file and write down (no figure required) the accuracy and losses. (**20 points**)
4. Conclusions that can be drawn based on the various improvements. Explain why you observe certain trends. Where do the improvements come from? Which classes are most affected? (**10 points**)

III. TASK C: Finding the best architecture (30 points)

As you have seen, when creating an architecture, there are many parameters to consider when trying to improve generalisation. Thus, instead of trying by hand all the possible combinations of layers and parameters, an automated algorithm can be created. This algorithm should use as feedback the produced loss/accuracy of the test set for the network at the end of each training. With that information (and some of your intuition) you should be able to add/delete layers of different types (e.g. Dropout, Dense, Batch Normalisation etc.) in the architecture, in order to progressively improve it. It is also important to also note that you will need to save the network weights at the end of each epoch and then re-initialise the layers of the new network with increase/decreased layers (that are the same as before) with previously learned weights. The way you implement this is up to you. This is not a necessity, but for better results, you might want to consider using an optimisation algorithm (for example simulated annealing).

As a baseline, you should work with the Transfer Learning network that you chose in Task B and only experiment with the neuron part of the network (i.e. do not extend your method for the feature extractor). You should discuss how your algorithm works in detail and what rates you were able to achieve in your report. Also include the changes that were made in your network at each step.

There is a lot of freedom in this assignment. If you are unsure whether your idea will be eligible (and how many points you might get for it), contact Alex Stergiou.
The grading concerns:

1. Implementation of the automatic model search. (**20 points**)
2. Motivation of your choices. (**5 points**)

3. Reporting of your results, conclusions about how your model search performs, and what you final model looks like. (**10 points**)

Results (or what to hand in)

- A folder containing all your code properly commented to explain custom classes and functions and variables.
- The mentioned .csv files for the various tasks. Make sure you use a proper name, e.g. task_a_2_training.csv
- A report of your findings, with proper explanations and observations. See the text of each task to understand what is expected for the report. The target length is 10-15 pages, including tables and figures.

You can use the submit system to upload a zip file.

Grading

The number of points per task and subtask are shown between parentheses in the text above. These are the maximum number of points. If mistakes are made, or the deliverable is unclear, points will be deducted.

Contact

If you have any questions about the assignment, contact Alex Stergiou, or use Slack.

References

[1] Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement learning." arXiv preprint arXiv:1611.01578 (2016).
[2] LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86, no. 11 (1998): 2278-2324. Link.
[3] Krizhevsky, Alex, and Geoffrey Hinton. Learning multiple layers of features from tiny images. vol. 1, no. 4. Technical report, University of Toronto, 2009. Link.
[4] Yao, Bangpeng, Xiaoye Jiang, Aditya Khosla, Andy Lai Lin, Leonidas Guibas, and Li Fei-Fei. "Human action recognition by learning bases of action attributes and parts." In 2011 International Conference on Computer Vision, pp. 1331-1338. IEEE, 2011. Link.