AIN 433 - Introduction to Computer Vision Laboratory

HACETTEPE UNIVERSITY

Department of Artificial Intelligence Engineering

Programming Assignment 4
Fall 2023
Assoc. Prof. Dr. Ali Seydi KEÇELİ
R.A. Görkem AKYILDIZ

Due Date: 03.01.2024 - Wednesday (23:00:00) **Programming Language: Python 3**

Image Classification with Convolutional Neural Networks

In the first two parts of this project, you will get familiar with image classification by training Convolutional Neural Networks (CNN).The goals of these parts are as follows;

1. Understand CNN architectures and build a model from scratch and train on data.
2. Understand and implement transfer learning from a pre-trained CNN.
3. Analyze your results.
4. Experience with a deep learning frame, TensorFlow.

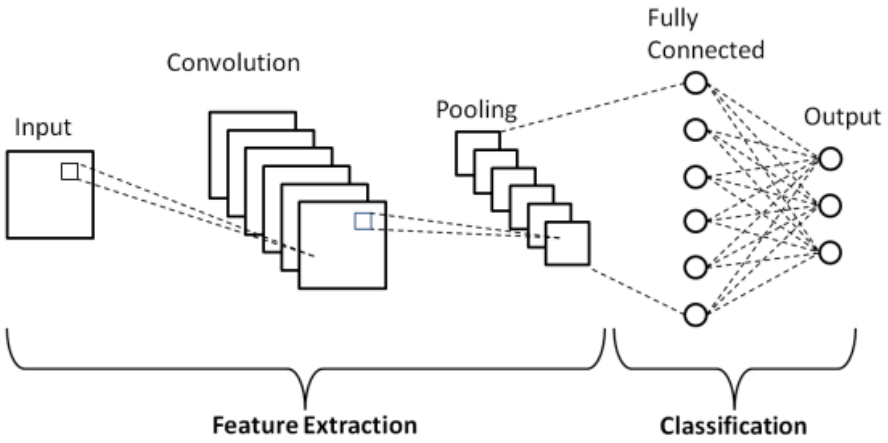

Figure 1: A simple CNN architecture [1].

Dataset for First Two Parts

You will use MIT Indoor Scenes dataset [2] for the first two parts. Dataset consists of sample of scene images, see Figure 2. There are 67 indoor categories. For these two parts, you will train classifiers to classify images to one of the 67 categories. If the computation time is a problem, you can use a subset of this dataset. However, make sure that your training set is at least 3000 images (200 images per class). The validation and test sets should include at least 50 images per class, a total of 750 images each. Note that the more data you use, the better the learning will be.

PART 1 - Modeling and Training a CNN classifier from Scratch

In this part, you are expected to model CNN classifier and train it. You should first define the components of your model.

- Give parametric details with relevant TensorFlow code snippets; number of in channels, out channels, stride, etc. Specify your architecture in detail. Write your choice of activation functions, loss functions and optimization algorithms with relevant code snippets.

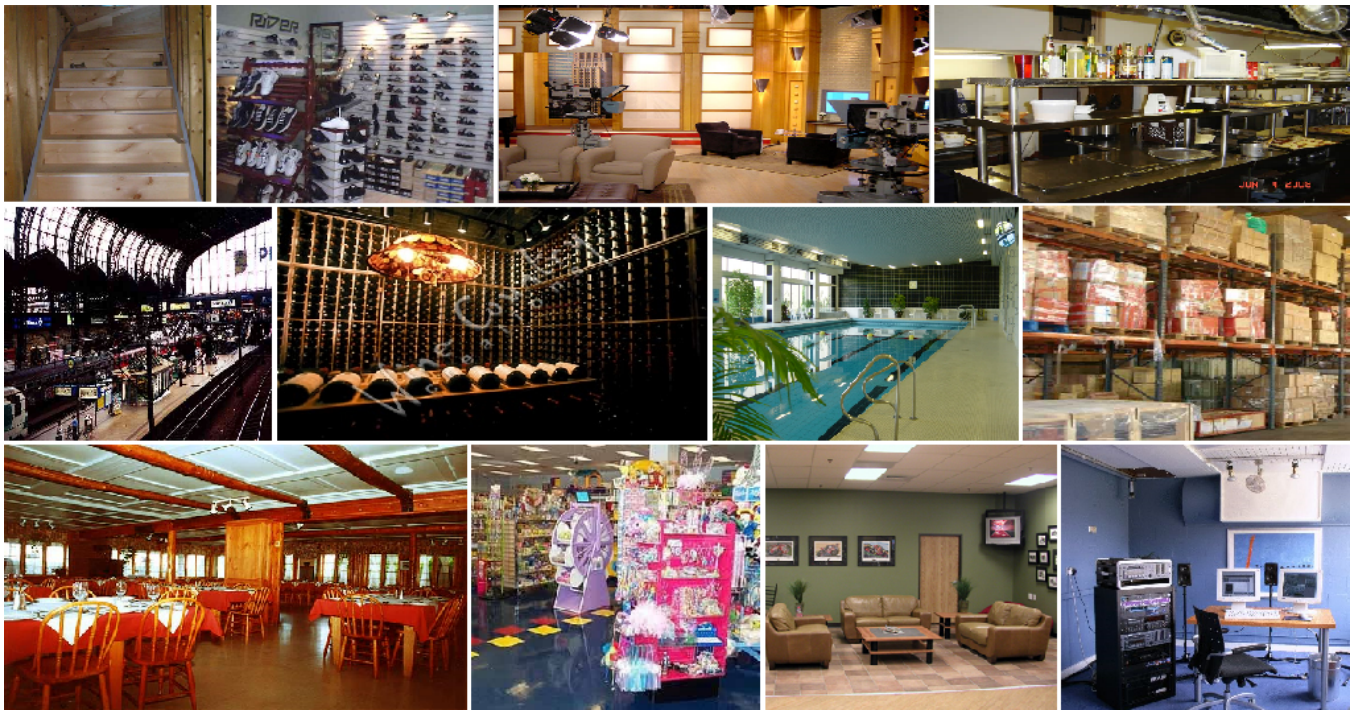

Figure 2: Dataset class samples.

Training and Evaluating your model

For both of your models you will set epoch size to 100 and evaluate your two model with three different learning rates and two different batch sizes. Explain how you calculate accuracy with relevant code snippet. Moreover for each of the points below add relevant code snippet to your document.

1. Draw graphs of loss and accuracy change for three different learning rates and two batch sizes.
2. Integrate dropout to your your model. In which part of the network you add dropout and why? Explore four different dropout values and give new validation and test accuracies.
3. Plot a confusion matrix for your model's predictions.
4. Explain and analyze your findings and results.

PART 2 - Transfer Learning with CNNs

Now, you will fine-tune the pre-trained VGG-16 network which is available at TensorFlow. This network is trained on ImageNet dataset so you are not initializing the weights randomly, instead, you are using the pre-trained weights from this network. Freeze all the layers before training the network, except the FC layers. Consequently, the gradients will not be calculated for the layers except the ones that you are going to update (FC layers) over the pre-trained weights. However, since the number of classes is different for our dataset, you should modify the last layer of the network, which the probabilities will be calculated on. Therefore, the weights will be randomly initialized for this layer.

1. What is fine-tuning? Why should we do this? Why do we freeze the rest and train only FC layers? Give your explanation in detail with relevant code snippet.
2. Explore training with two different cases; train only FC layers and freeze rest, train last two convolutional layers and FC layers and freeze rest. Tune your parameters accordingly and give accuracy on validation set and test set. Compare and analyze your results. Give relevent code snippet.
3. Plot confusion matrix for fine-tuned model and analyze results.
4. Compare and analyze your results in Part-1 and Part-2. Please state your observations clearly and precisely.

PART 3 - Object Classification and Localization

This part is separated from the first two parts and there is no direct relation between first two parts and the last part. You can assume this part as a completely different task. In this part, you will implement the basic object recognition and localization pipeline. You will use Raccoon dataset for localizing raccoons by using classification and localization framework discussed in class. The overview of the framework is given in Figure 3. You will use given train, validation and test splits.

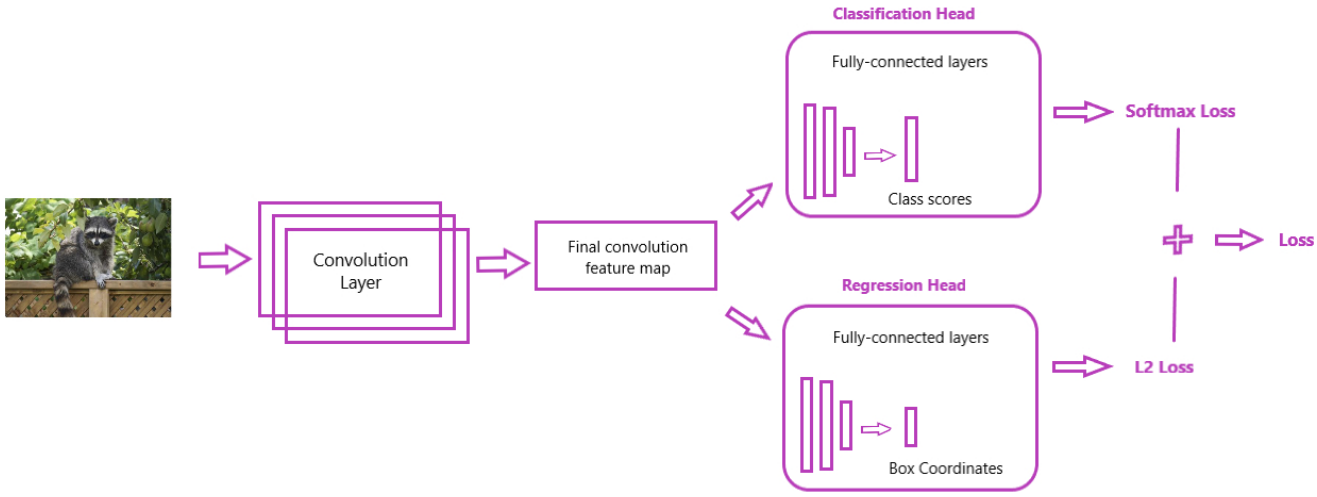

Figure 3: Overview of classification and localization.

Background

Classification and localization is a popular task that is used for classifying the object and finding the object's location in the image. Object location is represented with bounding box, which is defined by four numbers. In this project, you can use the pretrained ResNet18 [3] on ImageNet and use it in the classification head. Then, extend the network by adding a regression head after the last convolutional layer of ResNet. The classification head will be used for classification and the regression head will be used for finding the bounding box. The classification head will be fine-tuned using softmax loss and the regression head will use L2 loss. The full network will be trained using the combined loss. The whole process is depicted in Figure 3.

Implementation Details

1. Test the performance of the fine-tuned ResNet18 model for classification.
2. Extend the ResNet18 model with regression head to predict the location of the raccoons in the image.
3. Train both the classification and the regression heads using Raccoon dataset train set. Try different hyperparameters (learning rate, batch size) and report their performance over the validation set. Also report how the softmax, L2 and combined loss changes over time.
4. Implement non-maxima suppresion. Investigate how it affects the performance.
5. Finally report the final performance (MIoU) and classification accuracy over the test set, for the best performing model on the validation set.

Dataset for Part 3

In the third part of the project, you will use the Raccoon dataset [4] (You need to register the Roboflow and download the raw version). The raccoon dataset has one class with ground truth bounding box. Use existing train, validation, and test splits.

What should you write in the report?

- Give explanations for each step.
- Give experimental/visual results, used parameters and comments on the results in detail.
- Give your model's loss plot both for training and validation set during training.
- Put the results of different hyper-parameters (learning rate, batch size), the effect of them, with the loss plots and visualized bounding box predictions.
- A basic structure might be:
  1. Introduction (what is the problem, how do you approach to this problem, what is the content of your report)
  2. Implementation Details (the method you followed and details of your solution)
  3. Experimental Results (all results for separate parts with different parameters and your comments on the results)
  4. Conclusion (what are the results and what are the weaknesses of your implementation, in which parts you have failed and why, possible future solutions)
- You are encouraged to write your report in LaTeX
- You should give visual results by using a table structure.
- Do not forget to put the details that are requested under each part.
- It would be beneficial for reader to separate first two parts from the last part at your report as they are almost individual tasks.
- **Each part of your implementation must be reported at your report in order to get credits from the relevant code part, otherwise, as it is not that easy to evaluate your work just from your codes, you may face with point deductions just due to not being sure about did that part really works as implemented.**

What to Hand In

Your submission must contain following:

- README.md *(Text file containing the details about your implementation, how to run your code, which libraries have to installed, the organization of your code, functions etc. The template must be followed will be shared.)*
- src/ *(directory containing all your code)*
- Report.pdf

File hierarchy is as follows and must be zipped before submitted (Not .rar, only not compressed .zip files because the submit system just supports .zip files).

```
- b<StudentID>.zip
     - README.md
     - Report.pdf
     - <src>
        - *.*
```

Note that you MUST exactly score ONE from the submit system, otherwise you will have 20% of point deduction even if your hierarchy is correct. (For MacOS users, you can check the Piazza post for the script that zips the folder without any extra content which causes getting zero from submit)

Grading

The project will be graded out of 130:

- **45 % (part 1):** CODE: 35 REPORT: 10
- **40 % (part 2):** CODE: 30 REPORT: 10
- **45 % (part 3):** CODE: 35 REPORT: 10

The score for the report will be multiplied by your code score for that part which is divided by maximum score that can be taken from code part. Each part will be evaluated individually. For example, for second part say that some scored 25 for code part and 6 for report part, his/her final score for that part will be calculated as follows: 25+(25/30)*6=30

You MUST use justify -iki yana yasla in Turkish- page alignment and passive voice at your report, otherwise 20% of your score will be deducted FOR EACH VIOLATION for the relevant part of your report. Note that also your report's alignment must be well designed, otherwise you may also face with some point deductions for bad alignments/designs

You MUST write comments to your code as necessary and also your code MUST be readable.

Academic Integrity

All work on assignments must be done individually unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) will not be tolerated. In short, turning in someone else's work, in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.

References

[1] Binary image classifier cnn using tensorflow | by sai balaji | techiepedia | medium. `https://medium.com/techiepedia/binary-image-classifier-cnn-using-tensorflow-a3f5d6746697`˙(Last access: 19.12.2023).

[2] Mit indoor scenes. `https://www.kaggle.com/datasets/itsahmad/indoor-scenes-cvpr-2019`˙ (Last access: 19.12.2023).

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[4] Mit indoor scenes. `https://public.roboflow.com/object-detection/raccoon/2/download/yolov4pytorch`˙ (Last access: 19.12.2023).