AIN422 : Deep Learning Laboratory
Spring 2024
Instructor: Cemil Zalluhoğlu
TA: Sümeyye Meryem Taşyürek

Assignement 2: Vegetable Image Classification

Due date: Thursday, 25-4-2024, 11:59 PM.

The objective of this assignment is to cultivate your comprehension and familiarity with Convolutional Neural Network (CNN) concepts and transfer learning techniques in the context of CNN for image classification tasks.

1 Part 1: Implementation of a Simple CNN Model

In this section of the assignment, you will implement a basic Convolutional Neural Network (CNN) model from scratch for the task of image classification using a deep learning framework PyTorch.
Your objective is to construct a simple CNN architecture using fundamental components such as convolutional layers, activation functions, and pooling layers. Your model should have **at least 5 convolutional layers and a fully connected layer**. You will then train this model on the provided dataset containing images of various vegetables. To facilitate this task, a dataset containing images of various vegetables has been provided that you need to split into training, validation, and testing subsets.
You will use a subset Vegetable Image Dataset [1] contains about 4500 images of different vegetable species in different folders, corresponding 15 labels. The species type for each class is stated in Figure 2. To ensure balanced training, split the data into 3000 images for training (200 per class), with 750 images each for validation and testing (50 per class).

Convolutional Neural Networks (CNNs) have revolutionized the field of computer vision by enabling automated feature extraction and hierarchical learning from image data. Unlike traditional neural networks, CNNs preserve the spatial structure of images through convolutional layers, which apply filters to extract features such as edges, textures, and shapes.
Given an input image represented as a matrix of pixel values, CNNs perform a series of operations, including convolution, pooling, and non-linear activation, to progressively learn hierarchical representations of the input data. These learned representations enable CNNs to effectively classify images into different categories.

Steps to Follow

1. Load the vegetable image dataset using PyTorch's data loading utilities.
2. Preprocess the images by resizing them to a uniform size, normalizing pixel values, and applying any other necessary transformations using *torchvision.transforms*.
3. To ensure balanced training, split the data into 3000 images for training (200 per class), with 750 images each for validation and testing (50 per class).

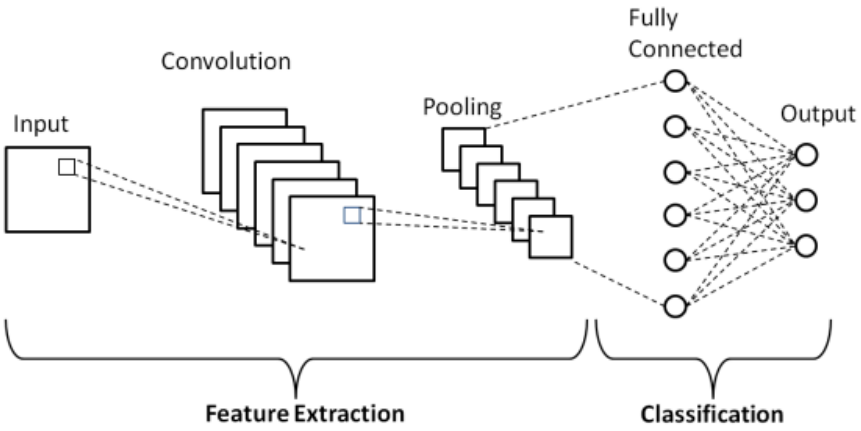

Figure 1: A simple CNN architecture

(a) Class 1: Bean

(b) Class 2: Bitter Gourd

(c) Class 3: Bottle Gourd

(d) Class 4: Brinjal

(e) Class 5: Broccoli

(f) Class 6: Cabbage

(g) Class 7: Capsicum

(h) Class 8: Carrot

(i) Class 9: Cauliflower

(j) Class 10: Cucumber

(k) Class 11: Papaya

(l) Class 12: Potato

(m) Class 13: Pumpkin

(n) Class 14: Radish

(o) Class 15: Tomato

Figure 2: Different vegetable species sample images from different classes.

4. Design a CNN architecture with at least 5 convolutional layers followed by activation functions (e.g., ReLU). Feel free to experiment with number of layers.
5. Intersperse pooling layers (e.g., max pooling) to reduce spatial dimensions and extract key features.
6. Flatten the output from convolutional layers to feed into a fully connected layer.
7. Add a fully connected layer at the end to map features to the output classes.
8. Define appropriate loss function for multi-class classification.
9. Choose an optimizer (e.g., SGD, Adam) and set its parameters (e.g., learning rate).
10. Explain your choice of activation functions, loss functions and optimization algorithms.
11. Iterate over the training dataset in mini-batches. Implemet forward pass, feed the inputs through the network to compute the predictions.
12. Compute the loss between predictions and actual labels. Implement backward pass, compute gradients of the loss with respect to model parameters.
13. Update model parameters using the optimizer based on computed gradients.
14. Validate the model on the validation set periodically to monitor performance and plot the validation loss.
15. Repeat the training process for a suitable number of epochs (at least 30 epochs).
16. Conduct experiments with different hyperparameters. Experiment with at least 3 different learning rates and 2 different batch sizes.
17. Keep track of performance metrics during these experiments to identify the optimal configuration. Visualize the accuracy and loss change of the experiments across training and validation datasets. Select your best model with respect to validation accuracy. Show this process in your notebook/report.
18. Test the best model on the test set to evaluate its performance. Compute metrics such as accuracy, precision, recall, and F1-score to assess classification performance.
19. Visualize confusion matrix for your best model to understand the model's behavior across different classes.
20. Explain and analyze your findings and results.
21. Summarize the performance of the model on the test set. Comment on the results.
22. Discuss any challenges encountered during training and potential areas for improvement.

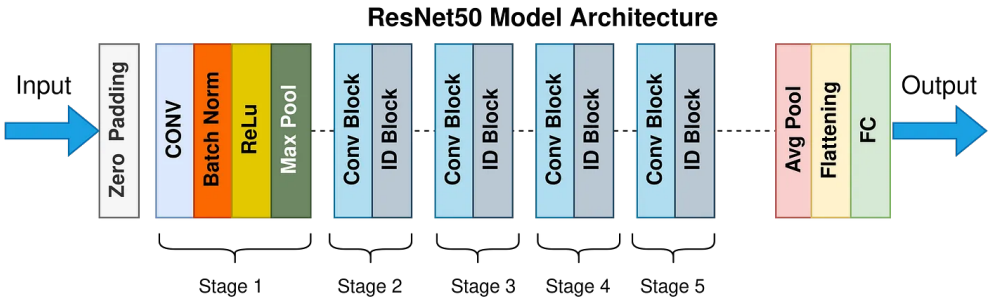

Figure 3: ResNet-50 architecture [2]

2 Part 2: Transfer Learning and Comparative Analysis

In this section of the assignment, you will explore the concept of transfer learning by utilizing a pre-trained CNN model for image classification. Specifically, you will employ transfer learning techniques to fine-tune pre-trained ResNet50 model on the vegetable image dataset.
The objective is to compare a fine-tuned pre-trained model with the CNN model implemented from scratch. Through this comparative analysis, you will provide insights and comments on the advantages and disadvantages of both approaches, shedding light on the practical considerations when choosing between implementing a CNN from scratch versus leveraging pre-trained models.

Transfer learning is a powerful technique in deep learning, especially when working with limited labeled data. Instead of training a CNN from scratch, transfer learning leverages pre-trained models that have been trained on large-scale datasets like ImageNet, which contain millions of labeled images across thousands of categories. These pre-trained models have already learned generic features that are useful for a wide range of visual recognition tasks.
In the context of this assignment, you will employ transfer learning with the ResNet-50 architecture, a deep residual network that has been pre-trained on ImageNet. By utilizing ResNet-50 as a feature extractor, you can leverage the learned representations from ImageNet to classify images of vegetables in the Vegetable Image Dataset. You will finetune the weights of the final fully connected (FC) layer in the ResNet50 network.

Steps to Follow

1. Utilize PyTorch's torchvision library to load the pre-trained ResNet50 model that has been trained on ImageNet.

2. Ensure that the model's architecture matches the ResNet50 architecture.

3. Freeze all layers of the ResNet50 model to prevent them from being updated during training. This ensures that the pre-trained weights remain intact.

4. Replace the final fully connected (FC) layer of the ResNet50 model with a new FC layer that matches the number of classes in the vegetable image dataset.

5. Define appropriate loss function and optimizer for training.

6. Train only the final fully connected layer using vegetable dataset while keeping the ResNet-50 weights frozen.

7. Monitor the training process and evaluate the model's performance on the validation set periodically. Visualize the accuracy and loss changes of the model across training and validation datasets.

8. Test the model on the test set to evaluate its performance. Compute metrics such as accuracy, precision, recall, and F1-score to assess classification performance.

9. Compare the performance of the fine-tuned model with the CNN model implemented from scratch in terms of accuracy and training efficiency.

10. Provide insights and comments on the advantages and disadvantages of transfer learning with pre-trained models versus implementing a CNN from scratch.

11. Discuss practical considerations when choosing between these two approaches, considering factors such as dataset size, computational resources, and task complexity.

12. Summarize the findings of the comparative analysis between fine-tuned ResNet50 models and the CNN model implemented from scratch.

3 What to Hand In

Your submitted solution should include the following:

- The filled-in Jupyter Notebook as both your source code and report. The notebook should include (1) markdown cells reporting your written answers alongside any relevant figures and images and (2) well-commented code cells with reproducible results.

- Additionally, your report should be self-contained, encompassing a concise summary of the problem statement and a detailed exposition of your implemented solution.

- Begin by providing a concise overview of the problem statement, detailing the objectives and scope of the assignment. Define the tasks involved in Part 1 (CNN from scratch) and Part 2 (transfer learning).

- Describe the dataset. Include information about the features, target variables, and any preprocessing steps applied to the data.

- Explain the approach adopted for each part of the assignment. Describe the algorithms implemented.

- Explain the evaluation metrics utilized to assess the performance of your models and justify their relevance to the problem at hand.

- Reflect on the outcomes of your experiments.

- Feel free to include tables or figures to emphasize specific aspects of your solution and enhance clarity for the reader.

You should prepare a ZIP file named <student id>.zip containing the Jupyter notebook in ipynb format as well as its .py (Python file) version containing all your codes. Submit it to submit.cs.hacettepe.edu.tr, where you will be assigned a submission. The file hierarchy is up to you as long as your Jupyter notebook and Python file works fine.

Attention! Please note that training the models could potentially require a considerable amount of time, particularly depending on the performance capabilities of your computer. To ensure that you obtain the required results in a timely manner, it is advisable to start the assignment early. If you find yourself starting the assignment later than anticipated and your computer's GPU capacity is limited, it might be beneficial to implement the assignment using Google Colab, leveraging its GPU resources for faster computation.

4 Grading

- Part1: Implementation of Simple CNN Model (60)
- Part2: Transfer Learning and Comparative Analysis (40)

Note: Preparing a good report is important as well as the correctness of your solutions! You should explain your choices and their effects to the results.

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 online/AI sources. Make use of them responsibly, refrain from generating the code you are asked to implement. Remember that we also have have access to such tools, making it easier to detect such cases.

References

[1] Vegetable Image Dataset
[2] The Annotated ResNet-50