PRACTICAL 2

Hacettepe University 01/11/2023

Unraveling CNNs

For the initial segment of this homework, we will be conducting a comparative analysis between fully connected neural networks (FCNNs) and convolutional neural networks (CNNs) to determine their respective efficiencies. This study will revolve around a visual classification problem encompassing 10 different categories, utilizing the Fashion MNIST dataset.

Begin by downloading the cnn.py starter Python script, and proceed to open it with your preferred code editor. Upon reviewing the script, you will observe that a FCNN model has been pre-implemented and is set up for training. Run the script, and make a note of the total number of trainable parameters, as well as the evaluation score achieved by the model.

Comparative Analysis of CNN vs FCNNs

In this task, we aim to build a Convolutional Neural Network (CNN) to solve a 10-class classification problem using the Fashion MNIST dataset. The network will consist of convolutional layers, max pooling layers, a flattening layer, and fully connected layers.

0.1 Experiment 1

Network Architecture

The CNN will be constructed as follows:

1. **Convolutional Layer**: Kernel size = 7×7, 16 feature maps, ReLU activation, padding = "same"
2. **Max Pooling**: Pool size = 2×2
3. **Convolutional Layer**: Kernel size = 3×3, 32 feature maps, ReLU activation, padding = "same"
4. **Max Pooling**: Pool size = 2×2
5. **Convolutional Layer**: Kernel size = 3×3, 64 feature maps, ReLU activation, padding = "same"
6. **Max Pooling**: Pool size = 2×2
7. **Flatten**: Flatten the output to feed into the fully connected layer
8. **Fully Connected Layer**: 64 neurons, ReLU activation

9. **Fully Connected Layer**: 10 neurons, Softmax activation

To implement this CNN, modify the cnn.py script according to the architecture specified above. Ensure to import the necessary libraries and define the model appropriately.

Make sure that you add this model as a second model in addition to the first fully connected model. The model selection mechanism is already implemented in the provided script via the “MODEL_NUMBER” global variable. That is, add this model for the case “MODEL_NUMBER = 2”. Do NOT change the training hyperparameters. Train this model and note the number of trainable parameters and the evaluation score.

0.2 Experiment 2

This task involves expanding our set of models for the Fashion MNIST 10-class classification problem by adding a third model, which is a fully connected neural network. This addition should be implemented in such a way that setting “MODEL_NUMBER” to 3 activates this newly added model.

Fully Connected Neural Network Architecture

The third model to be added is a fully connected neural network with the following architecture:

1. **Input Layer**: Flattened input of size 28×28
2. **Fully Connected Layer**: 74 neurons, ReLU activation
3. **Fully Connected Layer**: 50 neurons, ReLU activation
4. **Fully Connected Layer**: 10 neurons, Softmax activation

Modify the "cnn.py" script to include this third fully connected neural network model. Ensure that this model is added as an additional option, without altering the existing two models or the training hyperparameters. The mechanism for model selection based on the “MODEL_NUMBER” global variable should be utilized to activate this model when “MODEL_NUMBER” is set to 3.

After implementing the third model, proceed to train it using the same training hyperparameters as the previous models. Upon completion of the training, record the number of trainable parameters and the evaluation score achieved by the model.

0.3 Experiment 3

In this segment, we are tasked with extending our collection of models for the Fashion MNIST 10-class classification challenge by integrating a fourth model, which is a convolutional neural network (CNN). This model should be accessible and activated when “MODEL_NUMBER” is set to 4.

Convolutional Neural Network Architecture

The fourth model to be incorporated is a convolutional neural network with the following layers and configurations:

1. **Convolutional Layer**: 7×7 kernel, 64 feature maps, ReLU activation, “same” padding
2. **Batch Normalization**
3. **Max Pooling**: 2×2 pool size
4. **Batch Normalization**
5. **Convolutional Layer**: 5×5 kernel, 128 feature maps, ReLU activation, “same” padding
6. **Batch Normalization**
7. **Convolutional Layer**: 3×3 kernel, 128 feature maps, ReLU activation, “same” padding
8. **Batch Normalization**
9. **Max Pooling**: 2×2 pool size
10. **Batch Normalization**
11. **Convolutional Layer**: 3×3 kernel, 256 feature maps, ReLU activation, “same” padding
12. **Batch Normalization**
13. **Convolutional Layer**: 3×3 kernel, 256 feature maps, ReLU activation, “same” padding
14. **Batch Normalization**
15. **Max Pooling**: 2×2 pool size
16. **Flatten**
17. **Fully Connected Layer**: 128 neurons, ReLU activation
18. **Fully Connected Layer**: 64 neurons, ReLU activation

Adapt the "cnn.py" script to include this fourth convolutional neural network model. Ensure to add this model as a fourth option, maintaining the existing models and training hyperparameters. Utilize the “MODEL_NUMBER” global variable to facilitate the model selection, enabling this fourth model when “MODEL_NUMBER” is set to 4.

Following the implementation, train the fourth model with the identical training hyperparameters used for the previous models. After training, document the total number of trainable parameters and the evaluation score obtained by the model.

Model Comparisons and Analysis

Comparative Analysis of Trainable Parameters and Performance Scores

[13 points] Examine and contrast the total trainable parameters between MODEL_NUMBER = 1 (a dense neural network) and MODEL_NUMBER = 2 (a convolutional neural network). Also, evaluate and compare their performance scores. What insights or final thoughts can you derive from this comparison?

[12 points] Evaluate the difference in the quantity of trainable parameters between MODEL_NUMBER = 2 (a convolutional neural network) and MODEL_NUMBER = 3 (a dense neural network). In addition, contrast their performance outcomes. What deductions can you make based on these comparisons?

[10 points] Assess the performance scores across all the implemented models. Is there one model that stands out in terms of performance, disregarding the aspect of parameter quantity? If there is, identify which specific model excels under these particular circumstances.

Advanced Comparative Studies

[10 points] Considering the models with the highest and lowest number of parameters, analyze how the number of parameters correlates with the time taken for the models to train. Does having more parameters necessarily lead to longer training times?

[15 points] Examine the relationship between the depth of the convolutional neural network models and their performance in terms of image classification accuracy. In your analysis, consider both the increase in the number of layers and the associated memory complexity of the models. How does the depth of the network correlate with the memory usage, and what implications does this have on the model's ability to classify images accurately? Discuss whether adding more layers necessarily translates to better performance, taking into account the trade-off between computational efficiency and model complexity.

[10 points] The experiments utilize fixed training hyperparameters across different architectures. Discuss the potential challenges and limitations of this approach. How might the choice of hyperparameters influence the training dynamics and final performance of each model, and what strategies can be employed to find the optimal hyperparameters for each architecture?

0.4 Understanding CNN Through GRAD-CAM

[30 points] In prior studies, tools such as the deep visualization toolbox by Yosinski et al. have been used to shed light on the inner workings of convolutional neural networks. Yosinski et al. focuses on understanding what individual neurons "learn" and visualizing their activations across layers, Grad-CAM focuses on explaining the decisions of CNNs by highlighting

the most influential regions in the input image for a specific class. Implement the Grad-CAM technique as an alternative way to visualize the activations of hidden neurons in one of the convolutional neural network models from the experiments. Use Grad-CAM to highlight the regions in the input images that are pivotal for the model's predictions. Provide visualizations for at least three different classes from the Fashion MNIST dataset. Discuss the distinctions in insights gained from the Grad-CAM visualizations compared to tools like Yosinski's, and comment on the regions of interest identified by the model for each class.