AIN311 : Introduction to Machine Learning Lab.
Fall 2024
Instructor: Prof. Dr. Erkut Erdem
TA: Sevginur İnce

PART I: Theory Questions

Due date: Wednesday, 04-12-2024, 11:59 PM.

Question 1

What are activation functions, and why are they important in neural networks?

Question 2

Consider the convolutional neural network defined by the layers below. Fill in the shape of the output volume and the number of parameters at each layer.

- **CONV5-8**: Convolutional layer with 8 filters of 5×5, padding is 0, stride is 1.
- **POOL-2**: 2×2 max-pooling layer, stride is 2.
- **CONV3-16**: Convolutional layer with 16 filters of 3×3, padding is 0, stride is 1.
- **POOL-3**: 3×3 max-pooling layer, stride is 2.
- **FC-30**: Fully connected layer with 30 neurons.
- **FC-5**: Fully connected layer with 5 neurons.
- **Input**: $64 \times 64 \times 3$.

Solution for the Question 2

Layer	**Output Volume Shape**	**Number of Parameters**
Input	$(64, 64, 3)$	0
CONV5-8		
POOL-2		
CONV3-16		
POOL-3		
FC-30		
FC-5		

PART II: Classification of Skin Lesion Images using Neural Network

For this assignment, you will implement a neural network with one hidden layer and Convolutional Neural Network (CNN) architecture to classify skin lesion images into two classes using the Skin Lesion Image Dataset mentioned below.

Skin Lesion Image Dataset

The dataset used for this assignment consists of skin lesion images that are classified into two categories: benign and malignant. This dataset is widely used for research in dermatology and medical imaging, particularly for the automated detection and classification of skin lesions. The dataset includes high-resolution images labeled based on their diagnostic classification.
The dataset you will use in this assignment is a curated collection of labeled skin lesion images. An example of these images can be seen in Figure 2 and Figure 3.

- You can download the dataset from the following link: **link**
- This dataset contains a total of 9605 images, divided as follows:
  - 4605 malignant lesion images.
  - 5000 benign lesion images.
- The class to which each image belongs is specified in the `train_data.csv` and `test_data.csv` files. The classes are provided in the *diagnosis* column.
- Classes in the dataset are:
  - **Class 1:** benign
  - **Class 2:** malignant
- You should use the given train-test split.

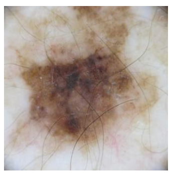
Figure 1: Malignant lesion

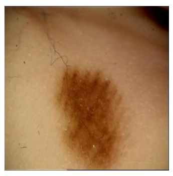
Figure 2: Benign lesion

Convolutional Neural Network

In this part of the assignment, you have to implement a Convolutional Neural Network (CNN) for classification (Figure 1). In other words, your network consists of n convolutional layers, m fully-connected layer(s), and one output layer. You will implement forward and backward propagations with the loss function and learning setting as explained in the previous section. Actually, you will implement a back-propagation algorithm to train a neural network.

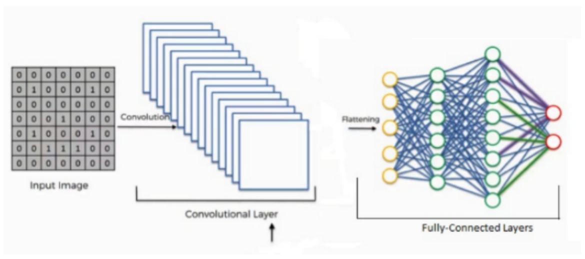

Figure 3: An example convolutional neural network

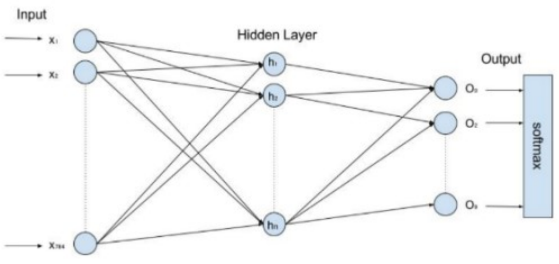

Figure 4: Multi Layer Neural Network

Multi Layer Neural Network

In this part of the assignment, you have to implement a multi-layer neural network for classification. In other words, your network consists of one input layer, n hidden layer(s), and one output layer. You will implement forward and backward propagation with the loss function and learning setting. Actually, you will implement a back-propagation algorithm to train a neural network. (You need to implement this part using the Numpy library.)
In this step, you will implement the network given in Figure 4 and train the network by feeding it with the given training set as gray-level image values. It is important to normalize image values (ranging from 0 to 255) to be between 0 and 1. We can express this network mathematically as:

$$O_i = w_{ij}x_j + b_i$$

As a loss function, you will use the sum of the negative log-likelihood of the correct labels. Write a Python function to compute the loss function. Then, you will update network parameters w and b to minimize the loss function using the gradient descent algorithm. You will implement a function that computes the derivative of the loss function with respect to the parameters. To ensure your function is correct, you must also implement a numerical approximation of the gradients.
Write a function to minimize your cost function using mini-batch gradient descent. You should try different learning rates (between 0.005 and 0.02) and batch sizes (between 16 and 128). Make a table to show the learning performance for each setting you tried.
Finally, you will visualize the learned parameters as if they were images. Visualize each set of parameters that connect to $O_0, O_1, \ldots, O_n$.

Training a Network

- You should determine the number of units in your convolutional layers and fully connected layers.
- You should determine the convolutional layer's width, height, and depth parameters.
- You should determine batch size as you learned in class.
- You should determine a learning rate for your gradient descent method.
- Remember, the learning rate parameter may be a problem (too big - may not converge, too small - very slow convergence). For this reason, you can define a learning rate decay parameter. You will start with a learning rate value and after each epoch, you will reduce the learning rate by multiplying it by a decay rate. This operation can deal with the mentioned problem.
- You can use different activation functions: Sigmoid (especially for the output layer), tanh, ReLU, etc.
- You can use binary cross-entropy error.
- You can control your implementation by plotting loss and tracking metrics such as accuracy, precision, recall, and F1-score. You can see if it converges or if it needs a different parameter setting.
- You should discuss about your each experiment in the report. Comment about their effects.
- Save your trained models to use later in test time.

Implementation Details for CNN

- For implementing CNN, you can utilize libraries from PyTorch. Ensure the final layer is adapted to have a single neuron with a sigmoid activation for binary classification.
- You can use different tactics for fixing overfitting and underfitting problems if necessary. For binary classification, consider addressing imbalanced datasets through techniques like oversampling or class-weight adjustments.

Important Notes About Assignment

- You should resize the image samples for classification.
- You can use a table for reporting your results. Experiment Input Size Model Activation Func. Hidden Layer Size Learning Rate Batch Size etc.

Experiment	Input Size	Model	Activation Func.	Hidden Layer Size	Learning Rate
Batch Size	Accuracy				
1	64x64x1	CNN	...	...	...
...					
2	64x64x1	MLP	...	...	...
...					
3	...	...	...	...	...
...					
4	...	...	...	...	...
...					

Table 1: Experiment Results

Obligatory Tasks

- You will implement a single-layer neural network (a neural network with only an output layer, which has a single neuron with a sigmoid activation function) and run experiments on the dataset. You will change parameters (activation func., objective func., etc.) and report results. Obligatory.
- You will implement a neural network that contains one hidden layer. You will change the mentioned parameters (unit number in the hidden layer, activation function, etc.) and report the results.
- Then you will change your architecture and use a network that contains two hidden layers. Repeat the same experiments and comment on the results.
- You will implement a convolutional neural network that contains one convolutional layer and one fully connected layer. You'll change the mentioned parameters (unit number in the hidden layer, activation function, etc.) and report the results.
- Then you'll change your architecture and use a network that contains two convolutional layers and two fully connected layers. Repeat the same experiments and comment on the results.
- You can also try different layer sizes (convolutional or fully connected) for your normal neural network or CNN model. But, you should not restrict yourself to the obligatory models above (especially for CNN), as you try different combinations with layer size and other parameters, your chance to get full points from the analysis report increases.
- You have to comment about results and parameters' effects on different values (learning rate, batch size, layer size, hidden neuron size in the layers, etc.).

Additional Implementation Guidelines

- Your implementation should be reproducible. In other words, do not write separate code for each architecture. If you use n layers, your method should create an n-layer network and learn the classifier with the final layer always containing a single neuron for binary classification.

- Comment your code with corresponding mathematical functions and explain what is going on in your code in the code scripts.
- You should also compare your classic neural network and convolutional neural network with respect to accuracy, F1-score, parameter (weight size) size, and training-test error curve plots. You must state every experiment you accomplish and related proper explanation/conclusion about why you obtain such a result in your analysis report to get full points on the analysis report.

Grading

- Code(60): 20 points for convolutional neural network implementations , 40 points for Multi-Layer Neural Network implementations
- Report (40) Theory part: 10 points, Analysis of the results for classification: 30 points.

Submit

- The filled-in Jupyter Notebook as both your source code and report.
- 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.

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 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.