Assignment 3

Due on December 13, 2023 (23:59:59)

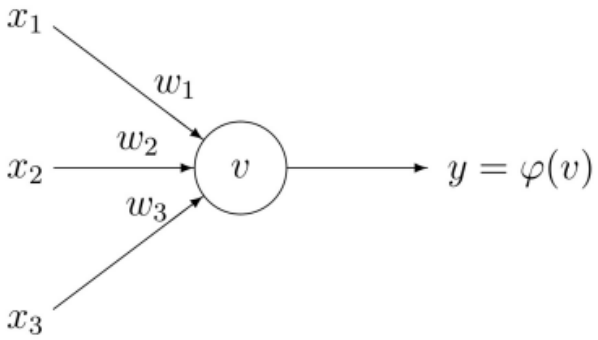
PART I: Theory Questions

1. Consider the convolutional neural network defined by the layers in the left column below. Fill in the shape of the output volume and the number of parameters at each layer. You can write the shapes in the numpy format (e.g. (64,64,3))

   - CONV3-10 denotes a convolutional layer with 10 filter with 3x3 size. Padding is 0, and stride is 1.
   - POOL-3 denotes a 3x3 max-pooling layer and stride is 2.
   - POOL-2 denotes a 2x2 max-pooling layer and stride is 2.
   - FC-20 denotes a fully-connected layer with 20 neurons.
   - FC-10 denotes a fully-connected layer with 10 neurons.

Layer	**Output Volume Shape**	**Number of parameter**
Input	127x127x4	0
CONV3-10		
POOL-3		
CONV3-10		
POOL-2		
FC-20		
FC-10		

2. Consider the simple neuron structure below:



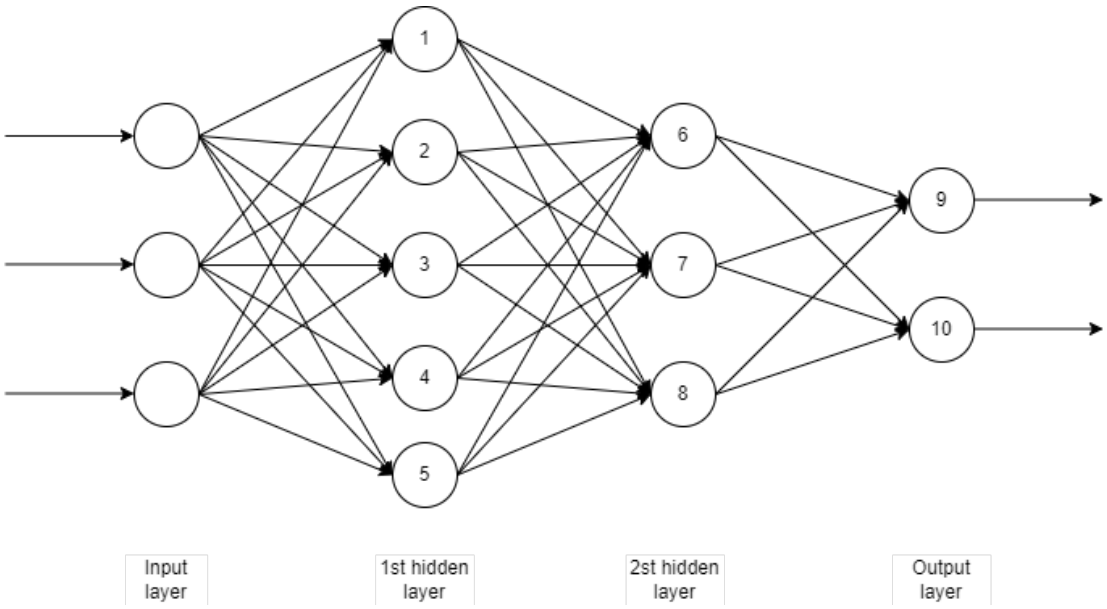
Assume that the weights for the neuron are $w_1 = 3$, $w_2 = -5$, and $w_3 = 2$ with activation function below:

$$\varphi(v) = \begin{cases} 1 & v \geq 0 \\ 0 & otherwise \end{cases}$$

find the output y values for the input patterns below:

Input	I_1	I_2	I_3	I_4
x_1	1	0	1	1
x_2	0	1	0	1
x_3	0	1	1	1

3. Consider the multi-layer neural network below:



- Find how many weight variables the network has in total (Ignore bias values). Show your calculations.
- Find how many weight variables the network has in total if the network is considered as fully connected (Ignore bias values). Show your calculations.
- State the dependency information for nodes given number values, which are about which node takes information from which previous node. State also these dependencies for both forward and back-propagation streams.

PART II: Classification of Mel Spectrogram 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 the age of the owner of the voice image in the Mel Spectrogram Image Dataset mentioned below.

Mel Spectrogram Image Dataset

A mel spectrogram is a representation of the spectrum of a signal as it varies with time. It is commonly used in signal processing and audio analysis, particularly in the field of speech and music processing. The term "mel" refers to the mel scale, which is a perceptual scale of pitches that approximates the human ear's response to different frequencies.

The dataset you will use in this assignment is the audio files in the Common Voice dataset prepared by Mozilla [1], converted into image data using Mel Spectrogram. You can see an example of the Mel Spectrogram image in Figure 1.

- You can download the dataset from given link.
- This data set contains 9600 train and 2400 test images. The class to which each image belongs is specified in the train_data.csv and test_data.csv files. The classes are given in the "age" column.
- Classes in the dataset are categorical data with equal distribution. Categorical classes:
  - Class 1: teens
  - Class 2: twenties
  - Class 3: thirties
  - Class 4: fourties
  - Class 5: fifties
  - Class 6: sixties
- You should use the given train and test split.

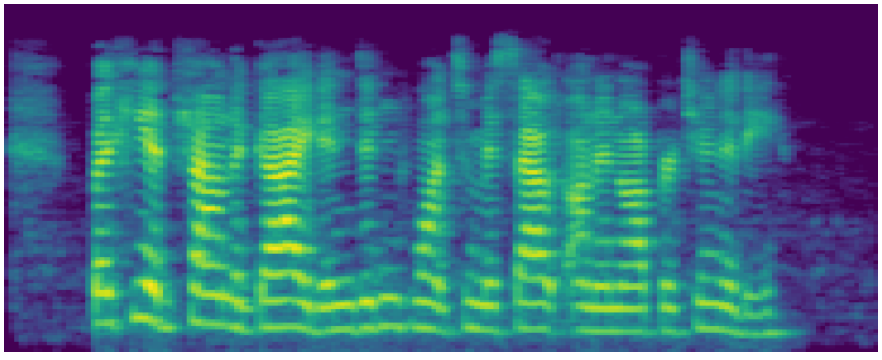


Figure 1: An example of Mel Spectrogram image of a voice that is classified as thirties class.

Multi Layer Neural Network

In this part of the assignment, you have to implement 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 propagations 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 2 and train the network feeding by the given training set as gray-level image values. It is important to normalize image values $(0 - 255)$ to 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 gradient descent algorithm. You will implement a function that computes the derivative of the loss function with respect to the parameters. To make sure your function is correct, you must also implement numerical approximation of gradients.

Write a function to minimize your cost function using mini-batch gradient descent. You should try different learning rates (between $0.005 - 0.02$) and batch sizes (between $16 - 128$). 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 of each set of parameters that connect to O_0, O_1, . . . , O_n.

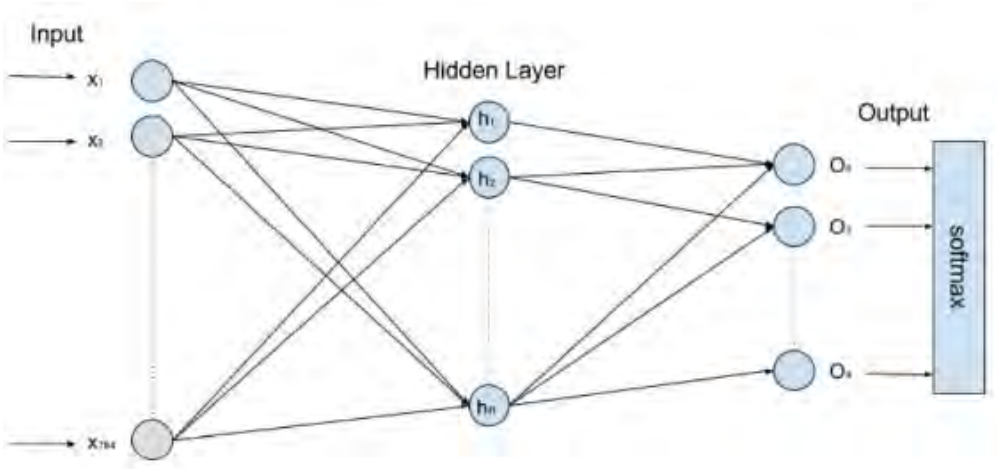


Figure 2: Multi layer neural network.

Convolutional Neural Network

In this part of the assignment, you have to implement a Convolutional Neural Network(CNN) for classification (Figure 3). 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.

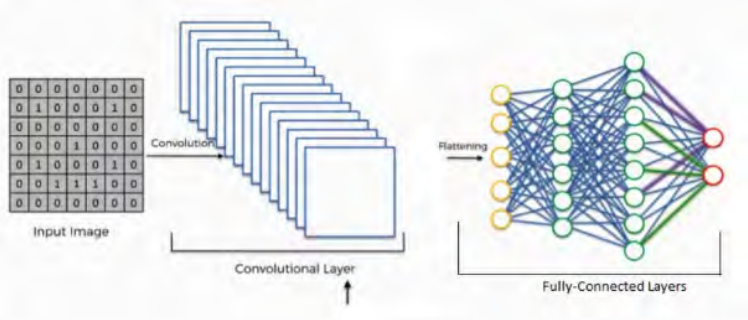


Figure 3: An example convolutional neural network.

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, tanh, ReLU, etc.
- You can use different objective functions: Cross-entropy error, the sum of squared error (SSE), etc.
- You can control your implementation by plotting loss. 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 TensorFlow [2] and you can use transfer learning which means you can fine-tune a pre-trained CNN model for the classification.
- You can use different tactics for fixing overfitting and underfitting problems if necessary.

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	...	...	Accuracy
1	64x64x1	CNN							
2	64x64x1	MLP							
3									
4									

- You will implement a single-layer neural network(a neural network with only has an output layer) 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. **Obligatory**
- Then you will change your architecture and use a network that contains two hidden layers. Repeat the same experiments and comment on the results. **Obligatory**
- 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. **Obligatory**
- 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. **Obligatory**
- 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...).

- 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 a n-layer network and learn the classifier. (For MLP and CNN)
- 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 the accuracy, parameter (weight size) size, and training-test error curve plots and you must state every experiment you accomplish and related proper explanation/conclusion about why you obtain such a result in your analysis report in order to get full points on the analysis report.

.

Submit

You are required to submit all your code in a Jupyter Notebook, along with a report in ipynb format, which should also be prepared using Jupyter Notebook. The code you submit should be thoroughly commented on. Your report should be self-contained and include a concise overview of the problem and the details of your implemented solution. Feel free to include pseudocode or figures to highlight or clarify specific aspects of your solution. Finally, prepare a ZIP file named name-surname-a3.zip containing:

- assignment_3.ipynb (including your report and code)
- assignment_3.py (py file version of your ipynb file)
- If you used pictures in Part I, please send the pictures as well.
- Do not send the dataset.

The ZIP file will be submitted via Google Classroom. Click here to accept your Assignment 3.

Grading

- **Code (60):** 30 points for multi-layer neural network implementations, 20 points for convolutional neural network implementations, and 10 points for other code parts.
- **Report (40):** Theory part: 10 points, Analysis of the results for classification: 30 points.

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

Late Policy

You may use up to four extension days (in total) over the course of the semester for the three problem sets you will take. Any additional unapproved late submissions will be weighted by 0.5. You have to submit your solution in (the rest of your late submission days + 4 days), otherwise, it will not be evaluated.

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.

References

[1] Common Voice, https://www.kaggle.com/datasets/mozillaorg/common-voice/
[2] TensorFlow CNN, https://www.tensorflow.org/tutorials/images/cnn