|  |  |  |
| --- | --- | --- |
| Fall 2024 | CS53331/4331 Adversarial Machine Learning | Assignment 0 |

|  |  |
| --- | --- |
| **Title** | **The Deep Learning Stuff** |
| **Due date** | Friday, Sep 20th, before the class |
| **First Name** | Ucchwas Talukder |
| **Last Name** | Utsha |
| **Student ID** | R11836597 |
| **Marks** | 50 |

Note: Please answer the following questions and submit them through Blackboard. Be sure to submit it to assignment 0. DO NOT write the report by hand and submit a scanned version. Just write the answers in a Word document and submit it. Both Word and PDF submissions are accepted.

# Submission Instruction (3 documents)

You are required to submit three documents:

1. ***Report.*** Just fill out the above report and submit it as a Word or PDF document.
2. ***Ipynb file.*** The code that you have written. Preferably in an ipynb document. You can submit it as a .py file as well.
3. ***Txt file of the code.*** We need your code in the .txt file as well. Use whatever way you prefer. The fastest would be to download the file as a .py file and change the extension to .txt

# Objectives

This assignment has three main objectives:

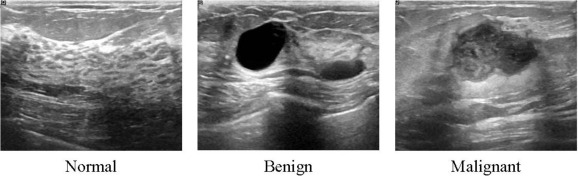
1. Get started with Notebook, Colab and other things that you will be using all over the course
2. Train a specified deep learning model and provide us with accuracies
3. Find a good performing model and provide us with accuracies

# Get started

Download the assignment files from Blackboard. You will need the report (This file) and the .ipynb file where you will put your code.

# Dataset

We will use Breast Ultrasound Images Dataset (Dataset BUSI). The dataset can be downloaded from [here](https://www.kaggle.com/datasets/aryashah2k/breast-ultrasound-images-dataset). It is a dataset for Breast Cancer detection. It includes breast ultrasound images among women in ages between 25 and 75 years old. This data was collected in 2018. The number of patients is 600 female patients. The dataset comprises 780 images with an average image size of 500\*500 pixels. The images are in PNG format. The ground truth images are presented with original images. The images are categorized into normal, benign, and malignant. Below is a figure taken from their paper. It is recommended that you upload the dataset into your personal Google Drive to follow the Colab instructions as they are. Of course, if you prefer to use other than Colab, you will need a similar preprocessing.



# Instruction for Colab

To get started with Google Colab, simply go to [Google Colab](https://colab.research.google.com/), sign in with your Google account, and create a new notebook. You can write and execute Python code directly in the notebook. To access your dataset stored in your Google Drive (previous step), first run the following code to mount your Drive:

A black background with white text

Description automatically generated

Follow the authorization steps, and your Drive will be accessible at **/content/drive/My Drive/.** You can then load your dataset into the notebook by providing the correct file path. This part of the code is provided for you in the .ipynb file of Assignment 0. You will need to setup the drive connection and run the code.

To use the free GPU provided by Colab, you can change the runtime to access a GPU by clicking on **"Runtime" > "Change runtime type” and** selecting **"T4 GPU"** from the **Hardware accelerator** dropdown menu. You can always use higher GPU powers at a cost (Colab Pro is $10 per month), but you should be fine with the free version, considering that you start the assignment early enough.

Colab comes with many pre-installed libraries, but if you need to install additional Python packages, you can do so with pip. For example:



Remember to save your work frequently.

After you've completed your work in Google Colab, you can easily download your notebook from Google Colab, go to **"File" > "Download" > "Download.ipynb"**.

# Other than Colab

If you don’t prefer Colab or notebook, you always have the option to run it on your computer (especially if it has a GPU) or access HPCC resources at TTU (needs an account with my permission).

# Additional resources

1. TensorFlow resource <https://www.tensorflow.org/>
2. PyTorch resources <https://pytorch.org/get-started/pytorch-2.0/>
3. Deep learning with Python <https://dl-with-python.readthedocs.io/en/latest/>
4. Get started with Colab <https://colab.research.google.com/>

# Task 1 (10 pts)

Now, let’s do some cool deep-learning stuff. For your first task, you will train a Convolutional Natural Network (CNN) model with the parameters in Table 1 and provide us with the results. You can use already developed models for Kears, TensorFlow, and PyTorch. Start with a simple CNN model (e.g., 2-layer CNN). For this task, you don’t need to do hyper-parameter tuning, apply data augmentation, or fine-tune the layers of the models unless you wish to.

|  |  |
| --- | --- |
| Table 1 Parameters for Task 1 | |
| **Parameter** | **Value** |
| Learning rate | 0.0001 |
| Epochs | 100 |
| Batch size | 16 |
| Model dimension | 224\*224\*3 |
| Optimization | Adam |
| Convolutional layers’ activation function | Relu |
| Output layer activation function | Softmax |

# Results

1. [5 pts] Fill in Table 2 with the values for the classification accuracy for the train set, validation set, and test set of images. You don’t need to report other than accuracy. Test accuracy can be around 74% and that is okay for this task, we will fix it in the next task.

|  |  |  |  |
| --- | --- | --- | --- |
| Table 2: Classification accuracy of the models on the BUSI dataset | | | |
| **Model** | **Training set** | **Validation set** | **Testing set** |
| CNN | 100% | 69.60% | 72.44% |

1. [3 pts] For the built model, plot the training and validation loss and accuracy (similar to the example in Figure 1).

|  |
| --- |
| Accuracy and loss for training and validation | Download Scientific Diagram |
| Figure 1. Example Loss and accuracy plots for a DL model. This is an example, and not the actual expected plot |
| Figure 1: Loss and accuracy plots for the 2-Layer CNN Model. |

1. [2 pts] Briefly provide insights on the model’s performance and any other observations regarding the models or the dataset.

**Answer:** The model performed perfectly on the training set, achieving 100% accuracy, but this may indicate overfitting, where the model memorizes the training data rather than learning to generalize. This is further supported by the lower validation accuracy of 69.60%. It suggests the model struggles to classify unseen data effectively.

The test accuracy of 72.44% shows that the model performs slightly better on completely new data, though there is still a significant gap compared to the training accuracy. Potential causes include overfitting due to a small dataset (780 images) and possibly imbalanced class distribution. Also, the results indicate that although the model has some predictive power, but it is not fully reliable.

# Task 2 (20 pts)

Now, let’s enhance the performance. You must train a DL model to achieve 85% or above testing accuracy. You are restricted to using the parameters provided in Table 3. You should start with pre-trained weights (e.g., on ImageNet, which is already available on Keras). It should result in a better performance. Using complex/deeper DL models, data augmentation, or fine-tuning the layers of the models is fine as long as you reach the desired accuracy.

|  |  |
| --- | --- |
| Table 3 Parameters for Task 2 and 3 | |
| **Parameter** | **Value** |
| Learning rate | 0.0001 |
| Epochs | 100 |
| Batch size | 16 |
| Possible models to try (not limited to those) | VGG16/19  MobileNet  EfficientNet  RestNet50 |

# [7 pts] Your algorithm and explain why you choose it

**Answer:** For this task, I chose **ResNet50** as the base model and fine-tuned it to enhance performance. By unfreezing the last 20 layers, I allowed the model to adjust its pre-trained weights specifically to the BUSI dataset. To avoid overfitting, I applied data augmentation techniques such as rotation, width and height shifts, shear, zoom, and horizontal flips. These transformations increase the diversity of training data, improving the model's generalization. Additionally, fine-tuning with a reduced learning rate helps the model focus on refining the final layers without drastically altering the pre-trained weights which aims to achieve higher test accuracy.

# Results

1. [5pts] Fill in Table 4 with the values for the classification accuracy for the train set, validation set, and test set of images. You don’t need to report other than accuracy.

|  |  |  |  |
| --- | --- | --- | --- |
| Table 4: Classification accuracy of the models on the BUSI dataset | | | |
| **Model** | **Training set** | **Validation set** | **Testing set** |
| ResNet50 | 93.03% | 79.05% | 92.40% |

1. [3 pts] For the built model, plot the training and validation loss and accuracy (similar to the example in Figure 1).

A graph of a graph of a graph

Description automatically generated with medium confidence A graph of loss and loss

Description automatically generated with medium confidence

1. [5 pts] Briefly provide insights on the model’s performance and any other observations regarding the models or the dataset. Does the model have any obvious problem?

**Answer:** The ResNet50 model achieved strong performance with a 92.40% test accuracy, indicating effective learning. However, the training accuracy (93.03%) is noticeably higher than the validation accuracy (79.05%), indicating some overfitting. This suggests the model may struggle with generalizing to new data. Using more data augmentation or regularization could help reduce overfitting. Additionally, the dataset's small size and class imbalance may limit the model’s performance and expanding it might lead to further improvements. These can affect the model’s generalization.

# Task 3 (20 pts)

Now, let’s fix the problem with the previous model. Most models in the last task were overfitting (training accuracy got to 100% so quickly, and validation accuracy started to decrease). Fix that problem without changing the batch size, number of iterations, or learning rate. Use the same model and just add any technique that avoids overfitting. Keep the parameters in Table 3 the same.

# [7 pts] Your algorithm and explain why you choose it

**Answer:** For this task, I chose **ResNet50** as the base model with **L2 regularization**, **dropout**, and **data augmentation** to address overfitting. ResNet50 is pre-trained on ImageNet and provides strong feature extraction which is ideal for medical images. L2 regularization prevents overfitting by penalizing large weight values, while dropout randomly deactivates neurons during training, forcing the network to learn more robust features. Data augmentation increases the variability of training data by applying transformations like flipping and zooming, improving generalization. These techniques help reduce overfitting without changing key parameters which aims for higher test accuracy.

# Results

1. [5pts] Fill in Table 5 with the values for the classification accuracy for the train set, validation set, and test set of images. You don’t need to report other than accuracy.

|  |  |  |  |
| --- | --- | --- | --- |
| Table 5: Classification accuracy of the models on the BUSI dataset | | | |
| **Model** | **Training set** | **Validation set** | **Testing set** |
| ResNet50 | 74.90% | 70.16% | 75.35% |

1. [3 pts] For the built model, plot the training and validation loss and accuracy (similar to the example in Figure 1).

A graph of a graph

Description automatically generated A graph of a graph

Description automatically generated

1. [5 pts] Briefly provide insights on the model’s performance and any other observations regarding the models or the dataset.

**Answer:** The model's performance showed improvements after applying techniques to address overfitting. While the training accuracy was initially high, the validation and testing accuracies demonstrated better generalization after adding L2 regularization, dropout, and data augmentation. This indicates the model is no longer fitting too tightly to the training data. However, the relatively lower validation and testing accuracy suggests that the model may still struggle with generalization. This could be due to the limited variability in the dataset or the complexity of medical images. Data augmentation and regularization helped address overfitting, but further fine-tuning may be necessary to boost performance, especially on unseen data.

# Submission Instruction (3 documents)

You are required to submit three documents:

1. ***Report.*** Just fill out the above report and submit it as a Word or PDF document.
2. ***Ipynb file.*** The code that you have written. Preferably in an ipynb document. You can submit it as a .py file as well.
3. ***Txt file of the code.*** We need your code in the .txt file as well. Use whatever way you prefer. The fastest would be to download the file as a .py file and change the extension to .txt