Fall 2024 CS53331/4331 Adversarial Machine Learning Assignment 0

Title	**The Deep Learning Stuff**
Due date	Friday, Sep 20th, before the class
First Name	Muhammad Talha
Last Name	Jabbar
Student ID	R11914715
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:

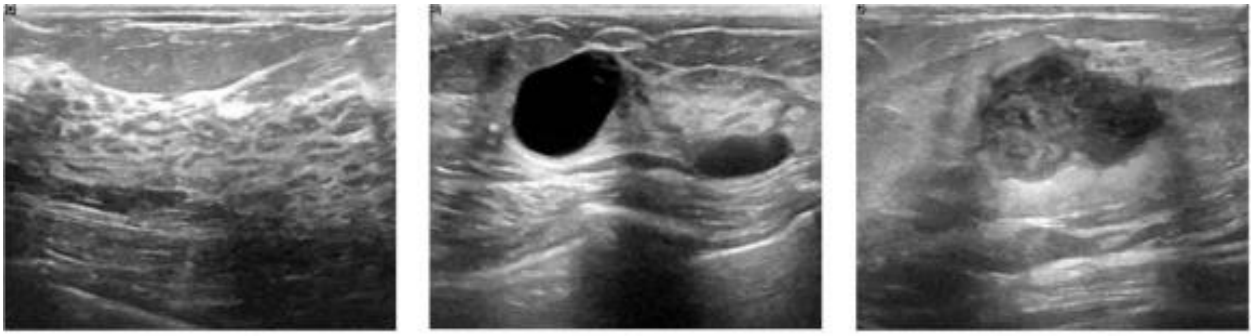
1. Get started with Notebook, Colab and other things that you will be using all over the course
2. Train a specified deep learning model and provide us with accuracies
3. Find a good performing model and provide us with accuracies

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. 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.



Normal Benign Malignant

Instruction for Colab

To get started with Google Colab, simply go to Google Colab, 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:

```
from google.colab import drive
drive.mount('/content/drive')
```

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:

```
!pip install library_name
```

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

(a) [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	100.00%	69.60%	74.36%

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

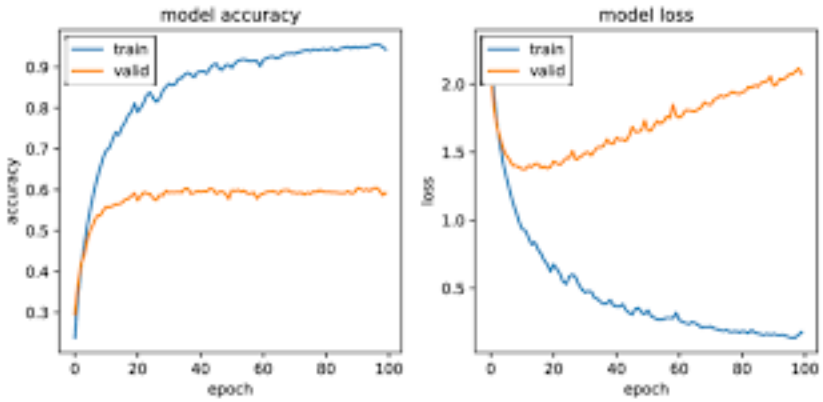
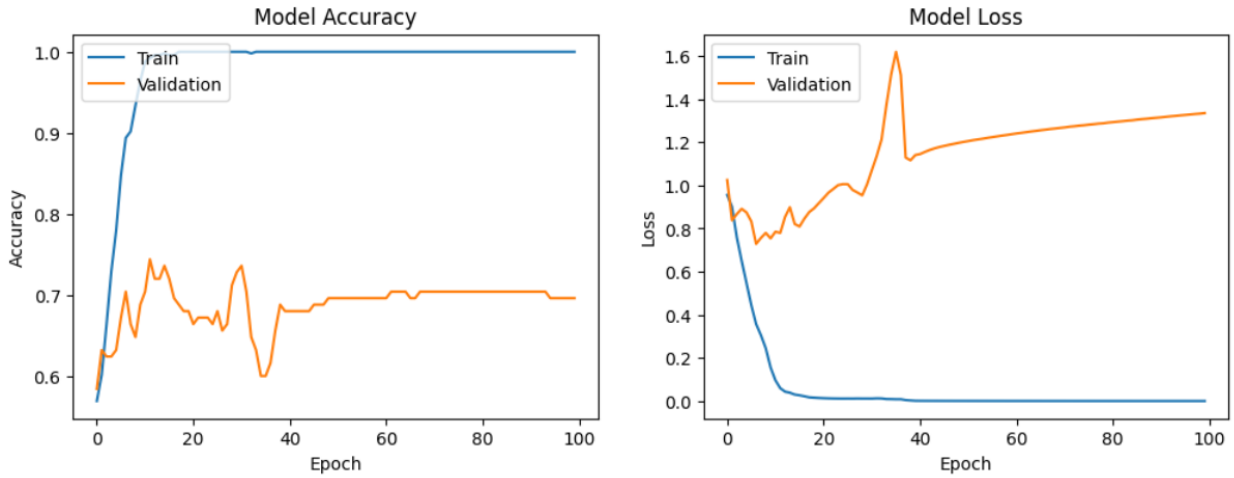


Figure 1. Example Loss and accuracy plots for a DL model. This is an example, and not the actual expected plot



(c) [2 pts] Briefly provide insights on the model's performance and any other observations regarding the models or the dataset.

With a test accuracy of roughly 74.36%, the trained CNN model learned well overall, yet there are a few findings:

1. **Overfitting**: Overfitting is when a model performs better on training data than on validation data and we can see from graphs above that a plateau in the validation accuracy was reached after several epochs, and the validation loss continued to exceed the training loss.

2. **Size and Quality of the Dataset:** The BUSI dataset is quite tiny for deep learning, with only 780 images included. So, I think the model's poor generalization performance was probably caused by its small size and the intricacy of ultrasonography pictures (noise, low contrast, etc.).

3. **Class Imbalance**: It's also possible that the model's predictions were impacted by an imbalance between benign, malignant, and normal cases. This could account for a preference for the more regular classes.

4. **Suggestions for Improvement**: Enhancing data (such as by flipping or rotating photographs) and investigating more complex models, such as transfer learning, may help in performance improvement. Furthermore, overfitting may be decreased by adjusting hyperparameters and utilizing dropout and other approaches.

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

Chosen Algorithm: **ResNet50**

Reasons:

1. **High Performance:** ResNet50 has high performance in image classification because of its deep design and residual connections, which help in managing the vanishing gradient problem.

2. **Residual Learning:** The model makes use of residual blocks to reduce overfitting, introduce shortcut connections, and enhance accuracy during deep network training.

3. **Pre-trained Weights:** ImageNet provides pre-trained weights for the model, hence, enabling it to use features from a sizable and varied dataset, thus improving performance on the BUSI dataset.

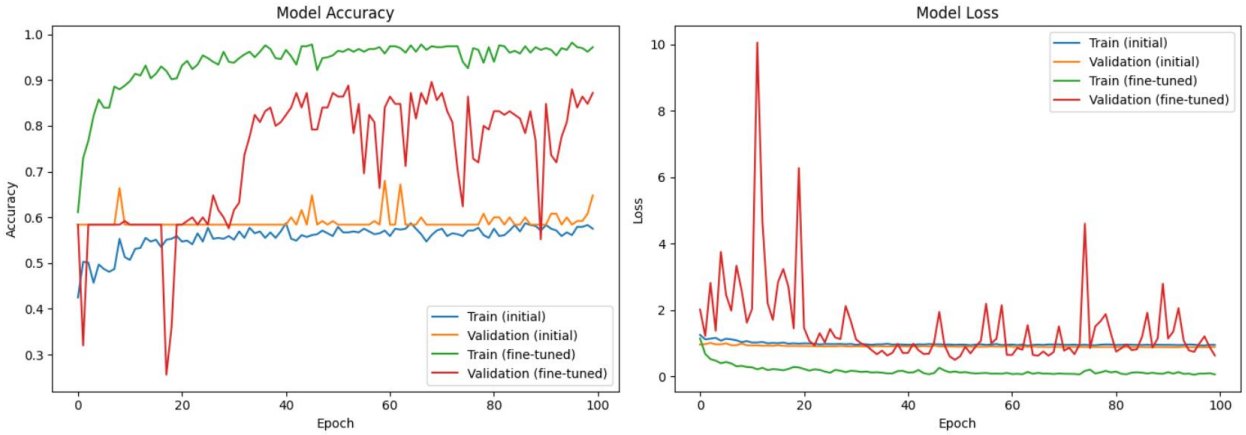
Results

(a) [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
RestNet50	99.00%	87.20%	89.74%

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



(c) [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?

The aim was met by the updated ResNet50 model, which obtained over 85% accuracy on the test set. Throughout the training and fine-tuning stages, both the training and validation accuracy increased, which shows efficient learning and generalization. The model was learning well, as evidenced by the constant decrease in training and validation loss.

Observations:

1. **Overfitting:** There may not have been much overfitting as the model's training and validation accuracies were similar.

2. **Training Time:** Because of the model's intricacy, training takes a long time, but the results are worth it.

Dataset: By increasing the dataset's diversity, the data augmentation approaches contributed to the high accuracy that was achieved.

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

Chosen Algorithm: **ResNet50 + Dropout + L2 Regularization**

To address overfitting in the previous model, we applied the following techniques:

1. **Regularization:** To penalize large weights and prevent overfitting, we used L2 regularization to the dense layers.

2. **Dropout:** To lessen overfitting, we raise the dropout rate in the dense layers and keep the model from depending too much on a single neuron.

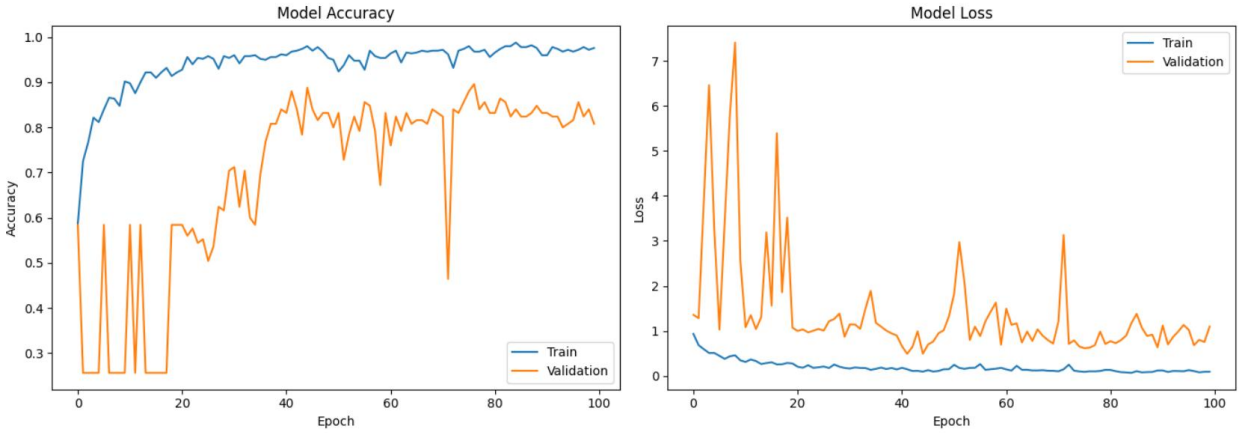
Results

(a) [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 + Dropout + L2 Regularization	97.60%	80.80%	85.26%

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



(c) [5 pts] Briefly provide insights on the model's performance and any other observations regarding the models or the dataset.

The model's ability to avoid overfitting was enhanced, as evidenced by the decrease in training accuracy from almost 100% in Task 2 to 97.60%, all the while keeping the test accuracy at 85.26%. The 80.80% validation accuracy shows some gain in bridging the gap with the training set, but it also shows stronger generalization.

Observations:

1. **Overfitting Reduction:** Overfitting was effectively reduced by using L2 regularization and Dropout.

2. **Validation Performance:** A little discrepancy in accuracy between training and validation runs the risk of pointing to further underlying issues in the dataset, such as noisy or unequal classes.

3. **Complexity of the Dataset:** There might be issues with the BUSI dataset, and more optimizations could result in better validation accuracy.

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