Appendix A: Work's Progress Plan

A.1. Winter Semester

1 – 3rd Week	Analyzing and exploring similar work to our task
4th Week	Obtaining a dataset
5th Week	Full EDA on the obtained dataset
6th – 7th Week	Planning the solution workflow
8th – 9th Week	Building our initial custom model
10th Week	Premilitary Experiment and obtaining results after training the model
11th Week	Writing down theory about our task and adding State of the Art followed by the results of our initial experiment

Evaluation of the work's progress plan for the Winter semester:

In the first 5 weeks of implementing our plan to solve the task we have done a lot of research to find reliable resources and to find similar researches that matches our task, after that we had to find a good and big dataset that can be used to implement our task.

From 6th week we started to plan how our workflow will be like, and started building our model to train it, we faced a lot of troubles to build it, because we had to normalize the dataset and standardize it.

After we have normalized the dataset we passed to the model and obtained our results.

That being done, we spent the rest of the semester writing the document of BP1 and filling it with some theory about the topic and State of art followed by our initial solution that we built and our results.

A.2. Summer Semester

1 – 3rd Week	Analyzing and exploring similar work to our task
4th Week	Further research on how to normalize the dataset
5th Week	Building dataset balancer
6th – 7th Week	Planning the solution workflow for Binary classification task
8th – 9th Week	Planning the solution workflow for Multi-label classification task
10th Week	Building the models and implementing transfer learning
11th – 12th Week	Writing down theory about our task and adding State of the Art followed by the results of our experiment with detailed explanation of everything we have done

Evaluation of the work's progress plan for the Summer semester:

We spent most of the first 6th weeks normalizing the dataset and trying different methods to apply the most suitable techniques to create windows and stack them to create 3 channels, Later on we had to choose middle slices to create a dataset that does not vary that much.

Later on, we balanced the dataset using a hybrid method between down-sampling and over-sampling, after that in the following weeks we tried many pre-trained models to see which one that might fit our model, after that we built our model for both of binary classification task and multi-label task.

In the last two weeks we spent them finalizing the model and training the model and obtaining the results and visualizing them in illustration graphs. And finally, we worked to format our document of the task.

Appendix B: Technical Documentation

B.1. Instruction to run the code

To run the code, some steps need to be done first, they are as follows:

1. Install anaconda following the instruction in this link: https://docs.anaconda.com/anaconda/install/

2. In folder **env** there is an environment called "***environment.yml***" which is the same environment that we had with all the needed dependencies to run the code in Jupyter Notebook or Jupyter Lab, To create the env using our environment simply run the following command in terminal or cmd: ***conda env create -f environment.yml***

3. After Creating the environment go to the dir where our project is, and run the command in the terminal or cmd***: jupyter notebook*** or ***jupyter-lab***

4. Then head to **Notebooks** folder where all the notebooks can be found for our project

5. The results of the training and testing are being deployed with run time on wandb.ai so an account needed to connect the code with Wandb's platform

B.2. General walkthrough of the files and the folders

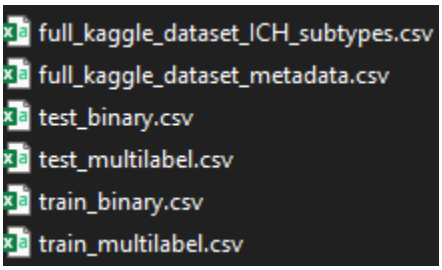
There are 5 folders for our project:

- Data
- Env
- Models
- Notebooks
- Documentations

In ***Data folder*** there are three subfolders:

- CSV
- Binary_Data
- MultiLabel_Data

In ***CSV folder*** there are all the needed csv that were used or generated to solve our task



To do EDA we used full_kaggle_dataset_ICH_subtypes.csv where we obtained it from Kaggle website with rest of the data set which can be found in this link:

https://www.kaggle.com/c/rsna-intracranial-hemorrhage-detection/data

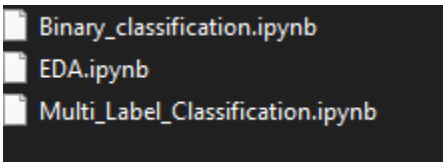
another important csv is full_kaggle_dataset_metadata.csv which we used to choose the middle slices of each Study Instance to get similar data set and from there we created 4 csv files for binary classification and then we isolated the ICH positive cases from those files and created two new csv files for multi-label classification.

In ***TRAIN and TEST*** folders we have just a sample of the dataset that we used to get the exact dataset first the dataset needs to be downloaded from the link above then just filter the data according to our csv files to create the same datasets.

In ***Models Folder*** we have the wights that we obtained after training the models for both of Binary classification task and Multi-label classification Task.

In ***env*** folder it has the anaconda environment “environment.yml” that has all the needed dependencies to run the code.

In ***Notebooks*** Folder, it contains 3 files:



In ***EDA.ipynb*** Notebooks there we have done EDA to get a better understanding of the dataset, all the functions can be seen inside of the notebook.

In ***Binary_classification.ipynb*** there we have developed the code to build a model to classify if a given image is ICH positive or negative.

The code consists of functions which are self-explanatory, and each function is named after its functionality.

In ***Multi_Label_classification.ipynb*** there we have developed the code to build a model to classify which subtype of ICH a given image it has.

The code consists of functions which are self-explanatory, and each function is named after its functionality.

In ***Documentation*** Folder it contains all the documentation for the work.

B.3. Important code explanation

In Multi-label classification Notebook we have many functions which will be described here

choose_mid_slices Function, It has a job of reading an input of the full_kaggle_metadata.csv and arrange the slices in ascending order according to the field “StudyInstance” and choose the middle slice of each “StudyInstance” and then choose the above 5 slices as our dataset where it takes the middle-chosen slice as for testing and the rest four slices for training.

move_files Function, It moves the files between the original folder and the one we want to move our new generated dataset from above.

create_targets Function originally was built to create targets for the dataset to make Multi class classification model but we have decided to move to multi-label classification model because we notice that many classes could be found in one image, so we excluded the use of this function

create_multi_lable Function, It generates a csv for the Multi-label classification task, by passing the chosen middle slices and then isolate the ICH positive cases and then get all the columns from the full_kaggle_dataset_ICH_subtypes.csv file for each “Slice name”.

sigmoid_bsb_window Function, It create each of “bone”, “soft” and “subdural” windows and stack them to create 3 channels image from dicom data array.

DicomDataset Class, It handles creating the dataset by assigning images from directory to its label and also apply all the preprocessing functions to the dataset and finally returns the image with its label.

Multi_classification_model_resnet152 Class, It creates our core model for Multi-label classification task by loading the model and deleting the classification layer and adding five parallel classification layers to each for each subtype of ICH subtypes.

ImbalancedDatasetSampler Class, We have used this class just in Binary classification task because we had a lot more ICH negative that ICH positive cases, It creates a balanced dataset using classes weights.

plot_confusion_matrix Function, It creates and plots the confusion matrix for further analyses.

binary_acc Function, It measures the accuracy between two arrays," Predicted" and "Actual" arrays for training and testing accuracies.

train_func Function, It passes the data with the labels to the model to be trained then it calculates the loss of the training.

test_func Function, It does similar job as train_func but it does not train the model it just evaluates the model after each epoch to track the test loss and the accuracy of the training and also creates some output samples of the predicted images with their results.

save_checkpoint and ***Load_checkpoints*** functions save a dictionary of the training weights and some other important hyperparameters that can be later loaded to proceed training or to test the trained model.

main Function, it basically runs all the previous classes and functions to train and evaluate the model.

Appendix C: Description Of The Digital Part Of The Work

Registration number of work in the information system: FIIT-100241-91983

Content of the digital part of the work (ZIP archive):

Folder	Description
\Final Bachelor	Contains all files and documents
\Data	sample of dataset with all used CSV files
\CSV	It contains all the csv files
\ full_kaggle_dataset_ICH_subtypes.csv	
\ full_kaggle_dataset_metadata.csv	
\ test_binary.csv	
\ test_multilabel.csv	
\ train_binary.csv	
\ train_multilabel.csv	
\Binary_Data	Contains Data for Binary classification
\TRAIN	Sample of binary training data
\TEST	Sample of binary testing data
\MultiLabel_Data	Contains Data for Multi-Label Classification
\TRAIN	Sample of multi-label training data
\TEST	Sample of multi-label testing data
\env	Anaconda environments with all needed dependencies
\ environment.yml	
\Models	Trained weights of both of Binary and Multi-label models
\NoteBooks	It contains all Notebook of EDA and Both classifier models
\Documents	Final BP document with all appendixes
\BP_AhmedLotfiAlqnatri.pdf	pdf Main part of final BP thesis
\ BP_prilohy_AhmedLotfiAlqnatri.pdf	pdf Text appendix of the BP thesis

Name of the uploaded archive: BP_prilohy_digital_AhmedLotfiAlqnatri.zip