PML Final Project - Bone Fracture Multi-Region X-ray Data (image classification)

Created by: Maria Dolgaya (28168) and Mariana Coelho (25605)

Professor Manuel Lameiras de Figueiredo Campagnolo

MSc in Green Data Science, ISA, Portugal

Introduction

Machine learning (ML) is a subset of artificial intelligence that focuses on the development of computer algorithms to learn from data and make decisions/predictions without being directly programmed to do so. These algorithms are designed to improve their performance over time, becoming more accurate and effective as they process more data [1]. Practical Machine Learning consists of constructing practical techniques to deal with data applied to ML. This is important since real data is often not independently and identically distributed. From this, we can efficiently train ML models [2]. With this project, we aim to apply ML techniques learned over the semester in order to work on a problem chosen by us.

The problem is based on the Bone Fracture Multi-Region X-Ray Dataset. The goal of this project is to build an image classification model, similar to the corn leaf disease detection example given in class and, if possible, build a space in HuggingFace with it. This task is interesting because it has potential to improve diagnostic efficiency and accuracy in medical settings.

Some possible challenges that we will be facing throughout the development of the project are: computational resources, because there's a large quantity of imagery files in the dataset, which can be challenging for the model; similarities in colors, due to the grayscale nature of X-ray images which can cause some shading and mislead the model; quality and variability of the images, which is another challenge as the images represent different anatomical regions and presence of noise can also affect the model performance; and lastly, overfitting, caused by the high dimensionality of image data.

Data and Data Organization

For this project, we will be using the Bone Fracture Multi-Region X-ray Dataset, collected from Kaggle. This dataset is composed of only image data, specifically fractured and non-fracture x-ray images of all anatomical body regions (lower limbs, upper limbs, lumbar area, knees, etc).

The dataset is composed of 10580 radiographic images (X-ray) data. This data is then splitted into three sets: training data, with 9246 images; validation data, with 828 images and lastly testing data, with 506 images. This dataset already includes necessary metadata regarding licensing and author credits.

Data cleaning and transformation was required so, to start, we created some snippets of code to remove corrupt files and empty subdirectories. After copying the database in order to perform changes on the images used, we removed corrupted images from the train, validation and test directories. Afterwards, a transformation method was built, including resizing, flip and rotate the images, as well as improving the colors so that the model could have a better understanding of the images, and as a way to diminish the challenge regarding similarities in colors due to the grayscale nature of X-ray images.

Methods

As it was mentioned before, the inspiration behind this project came from the Corn Leaf Disease Detection model, which was studied during this semester. We started by taking some bits of code from that notebook, also available in Kaggle, and tried to apply it for our dataset to see if it was possible. After collecting all the workable and necessary snippets, we then started to construct our own notebook for the project.

We first started by importing all the necessary Python libraries such as numpy, pandas, matplotlib, seaborn and most importantly, Pytorch, for the construction of the ML model. We chose Pytorch over Tensorflow because it was easier for us considering we have more experience with it from the classes in the semester. From Pytorch, we then imported several utensils, as well as some torchvision tools that would be necessary (In order to know which tools exactly, please check the **Google Colab notebook**). Following this step, we then defined the device in which the model would be built in and mounted the notebook into Google Drive so that everything from the project would be stored into one place.

The runtime type was set to GPU in order to increase speed and efficiency but, as it will be noticeable if the user runs the notebook in these conditions, the loading of the data set will take some significant time. We believe this must be related to the size of the dataset and the complexity of the images. After we imported the dataset onto the notebook, we started applying the data transformation steps, beginning with removing the corrupted images, empty subdirectories and defining data transformation and augmentation methods. In order to get a better understanding of the transformation, we allowed the user to see the before and after before starting training the model (see “Results” section of the report).

Starting with the training, a Neural Network (NN) was defined. A NN in ML is a model inspired by the structure and function of biological neural networks in animal brains. These are trained through empirical risk minimization [3]. Afterwards, we were able to define the model and save it directly into Google Drive. We then defined the loss and optimization for the train and validation data so that these would be accounted for and then we finally began the training, validation and testing of the model.

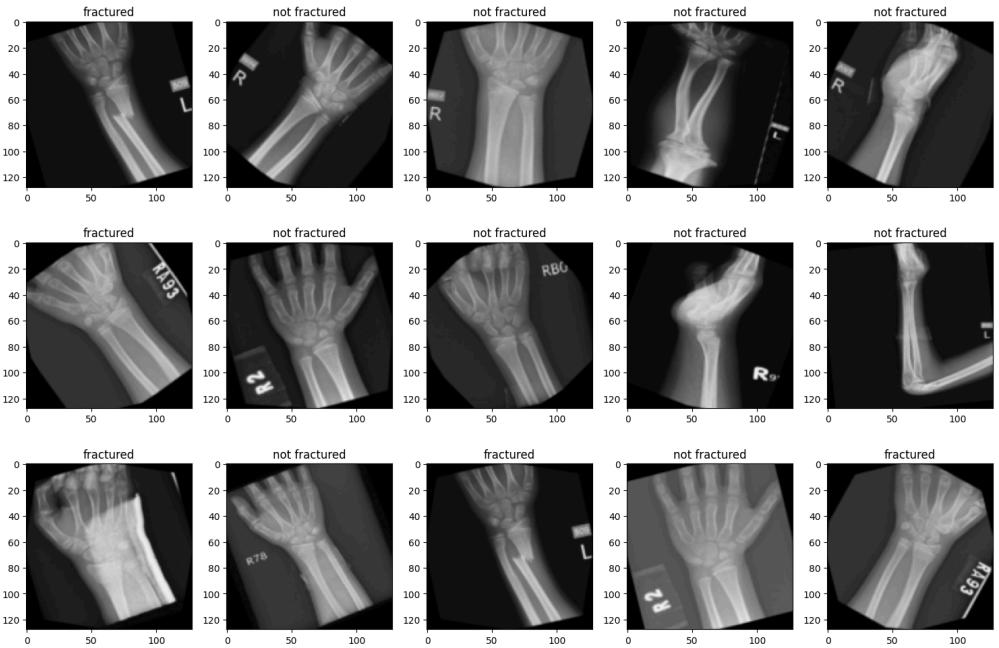
Two functions were constructed, one for the training and validation and one for the testing. Various tests were performed, all different from each other because of the number of epochs. In an earlier stage of the project, only three epochs were established, leading to a tes accuracy of around 94%. This of course wasn't enough so more tests were made and the number of epochs was increased. On a second try, we used ten epochs and got an accuracy of around 80%. This was more preferable but still more testing was made. So for the last test, the number of epochs was set to twenty and this one gave back an accuracy of around 90%. This might not be a perfect score but it's far from a bad result. It is also important to mention that when changing the number of epochs to run, it is crucial to clear the previous output. If this isn't performed, the accuracy score shown after the second trial, will be a cumulative result of the previous one with the most recent trial.

Lastly and as a way to better observe the results, we plotted a confusion matrix and two learning curves showing the loss and the accuracy of both the training and the validation data. All of these representations can be observed in the "Results" section of this report.

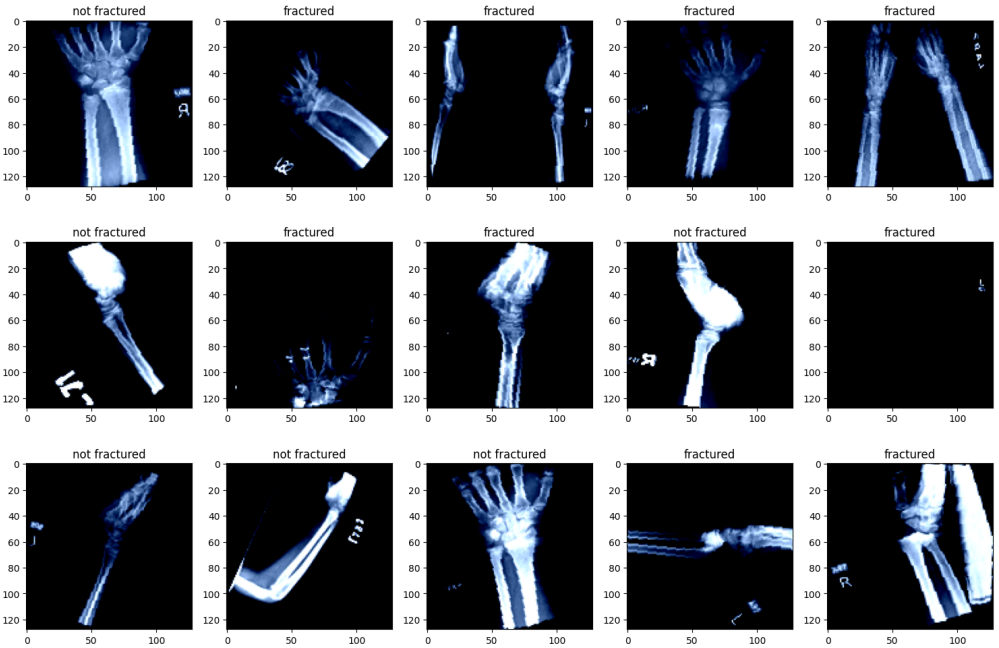
From this code, we were then able to build a HuggingFace Space very similar to the one built for the corn leaf disease detection model but this time with a bone fractures classifier trained using Resnet18. The basis of the code is the same, a pace created using gradio but now we have only two outputs (fractured and not fractured) instead of four. Besides the code, a text file was added titled requirements.txt, containing all the packages needed for the code to work, as well as the model file and some images to serve as input. We chose five images, three of fractured limbs and two with not fractured limbs to see how the model would react. The HuggingFace space can be accessed through **this link**: [Bone Fracture](). **.**

Results

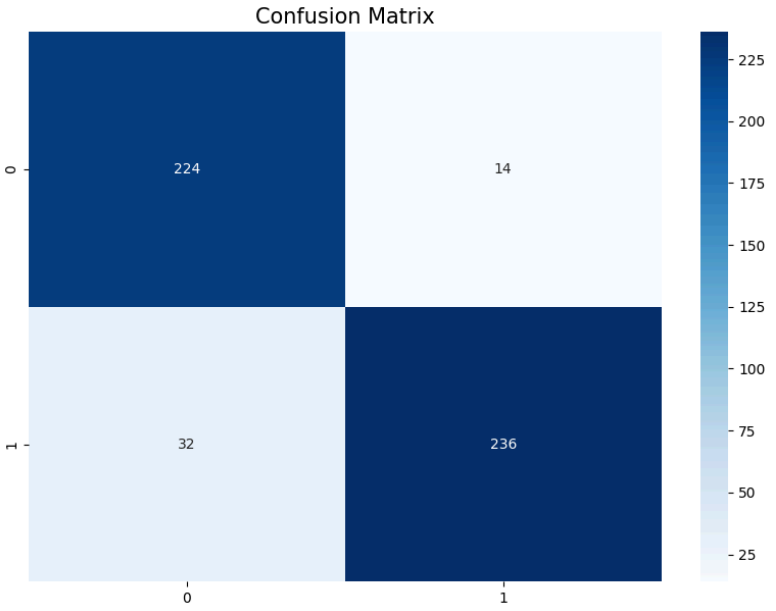
Images plot **before** data augmentation



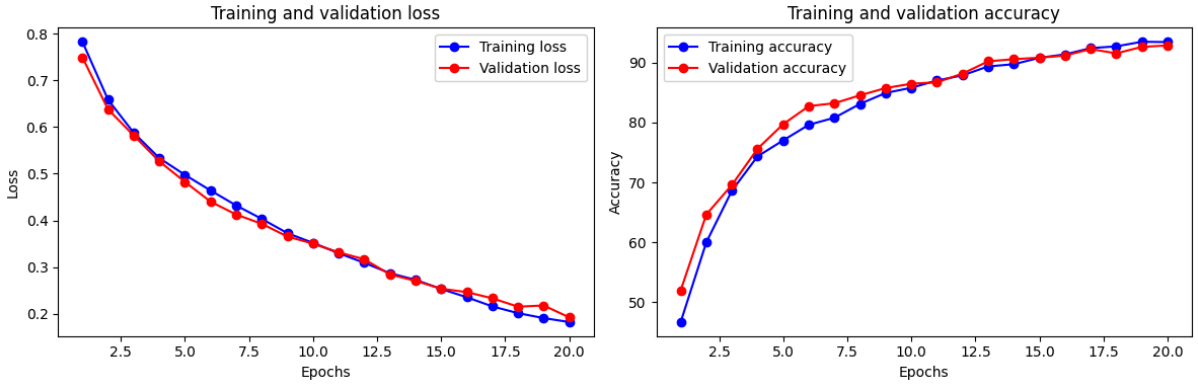
Images plots **after** data augmentation



Confusion Matrix



Learning Curves



Analysis

Building an image classification model is no easy task and although machine learning has been around for a while, it hasn't been applicable until more recent times, due to the functionalities of technology being in constant evolution.

Having a basis to work from was very important in our project because it gave us a clear idea of how we should approach the project from the beginning to the end, as well as the fundamentals learned throughout the semester, which were crucial to learn how to build the model.

The code shows good results, with high levels of accuracy even at the beginning when it wasn't fully functional and the number of epochs was very low. Although it would be probably better to run more trials, not all devices would be able to handle it. We could see that even with the runtime type set to GPU, the dataset was too heavy and could crash at any moment, compromising everything done up until that point and increasing the time spent working on the model.

Even though the results show a good accuracy when it comes to testing the model, the space created to test it out showed that there's still room for much improvement. Out of the five images used as examples, only three were correctly classified , although we're not sure if the two misclassified were because of the model or the images where the fractures aren't always perceivable. Another possible explanation for this to have occurred comes from the fact that the model.pth file used to create the HuggingFace space came from the usage of what's probably too many epochs (20). We believe that could be why the model was overfitted and one of the reasons why the HuggingFace space was misclassifying some of the images.

We believe that with hard work and determination, this model can be improved and get an accuracy of 100%, to then be applied to the medical field, in order to improve diagnostic efficiency and accuracy, so that this can work as a helpful tool for medical professionals.

References

[1] Crabtree, M. (2023) *What is Machine Learning? Definition, Types, Tools & More*. DataCamp. Available at: https://www.datacamp.com/blog/what-is-machine-learning?utm_source=google&utm_medium=paid_search&utm_campaignid=21374847033&utm_adgroupid=165153431722&utm_device=c&utm_keyword=&utm_matchtype=&utm_network=g&utm_adpostion=&utm_creative=703052951031&utm_targetid=dsa-2222697810918&utm_loc_interest_ms=&utm_loc_physical_ms=1011747&utm_content=DSA%7Eblog%7EMachine-Learning&utm_campaign=240617_1-sea%7Edsa%7Etofu_2-b2c_3-ptbr-lang-en_4-prc_5-na_6-na_7-le_8-pdsh-go_9-nb-e_10-na_11-na-june24&gad_source=1&gclid=CjwKCAjwps-zBhAiEiwALwsVYbfnImyRVlfbgc9vuORMYhbDl_6aLeSO2MCVSd-ilP5q3oJs26puxxoC_M0QAvD_BwE&dc_referrer=https%3A%2F%2Fwww.google.com%2F (Accessed: June 20th 2024).

[2] Practical Machine Learning — *Practical Machine Learning* (no date) c.d2l.ai. Available at: https://c.d2l.ai/stanford-cs329p/ (Accessed: June 20th 2024).

[3] Hardesty, L. (2017) Explained: Neural networks, MIT News. MIT. Available at: https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414 (Accessed: June 25th 2024)

Contributions

Code and HuggingFace Space: Maria Dolgaya (Github: https://github.com/DolgayaMaria)

Report: Mariana Coelho (GitHub: https://github.com/marianadc01)