Machine Learning Models for Interpretation of Knee MRI

Alexandru Licuriceanu
Computer Science Department
Politehnica University of Bucharest
Bucharest, Romania
alicuriceanu@stud.acs.upb.ro

Nelu-Rareș-Ionuț Cărăușu
Computer Science Department
Politehnica University of Bucharest
Bucharest, Romania
nelu_rares.carausu@stud.acs.upb.ro

I. Dataset Description

The MRNet dataset contains medical imaging data. Its purpose is to diagnose knee disorders. It comprises 2 directories with MRI of the knee from 1250 patients in total, one directory for data training and one for validation. There are 3 directories in each of these, containing the anatomical directions of MRI scans (sagittal, coronal and axial). Each file contains a volume of images (slices) with dimensions (256, 256). The files have the extension .npy, a binary file format in NumPy.

The training of the dataset considers the classification of each patient within three labels (ACL, meniscus and abnormalities). Within the dataset, there are also some csv files. Each of these has 2 columns, one for the patient ID and one categorized binary (0/1) according to whether the patient has the problem or not, where 0 means healthy and 1 means having the problem.

II. Preprocessing Data

The dataset was downloaded from Kaggle as a zip archive, and then unzipped. In the first part, we each created a dataframe for training and for validation containing all the information about each patient with respect to the 3 labels.

Next, we check if there are any missing values from any of the dataframes, and after this analysis, we see that there is no missing data. This is important because missing values can negatively affect the performance and interpretability of the model.

Another important step in data preprocessing is to check the class distribution. Within this, dataset there is an imbalance between classes. We tried to create an equilibrium between them by oversampling, but by multiplying the images within the minority class in a label, an even larger imbalance was created. Thus, we observe that an equal class distribution is not possible for all 3 labels simultaneously (acl, meniscus, abnormal).

Next, 3 columns corresponding to the 3 anatomical sections of the MRI scans (axial, coronal and sagittal) are added to the 3 dataframes. In this way, the row corresponding to a patient contains information about the 3 possible problems and the paths to the MRI images (on the axial column are the paths to the patients' images on the axial plane, etc.). Following the training of the model, we would also like to test the model to verify its quality in classifying images. Therefore, from the training set, a test dataset is taken out. This has a size of 130, which is quite close in value to the size of the validation set, 120.

To ensure the consistency and integrity of the dataset, we implemented a detailed image verification. By this, we ensured that all images have uniform dimensions. Different image sizes can introduce difficulties in model training. A solution to this problem could be to resize the minority of images to the dimensions of the majority of images. In addition, we checked if the pixel values are expressed on the same scales so that, the model can interpret the values correctly. One way to do this is to normalize the images. Upon checking, it was found that all images in all datasets have the same dimensions, namely (256, 256) pixels. At the same time, all of them need to be normalized since they do not have pixel values in the range [-1, 1].

In the next step, we applied a series of augmentations to the images. These augmentations are intended to diversify the existing images while increasing the size of the dataset. In this way, the model is trained to be robust to different variations, thus improving its performance and reducing the risk of overfitting. In reality, medical images may have poor quality, differences in angles, the model is trained to be able to analyze such images as well. In order for the model to be able to learn as well as possible, several transformations were applied on the training dataset.

The transformations applied are resizing the images to (128, 128) pixel dimensions (the reason for this is that (256, 256) images are resource expensive, producing a very large training time), random horizontal flipping with a probability of 50% (introduces variation in the images, simulating different perspectives), rotating the image with a random angle in the range [-15 degrees, 15 degrees] (in reality, MRI images can be slightly rotated due to the patient's position; typically, if the rotation angle is greater than 15 degrees, then the MRI is repeated), ColorJitter (produces variation in light intensity, changes contrast and changes color intensity, thus simulating real-life conditions), converting the images to PyTorch tensors and normalizing the images, by bringing the pixel values into the range [-1, 1] by setting the mean and standard deviation to 0.5 (faster convergence and balanced training).

The transformations within the validation and test datasets are identical. They involve resizing the images to (128, 128) pixel dimensions, converting the images to PyTorch tensors, and normalizing the images, resulting in pixel values in the range [-1, 1]. These datasets have only these transformations, because they are used in model evaluation and hyperparameter tuning (only in the case of the validation dataset).

III. Baseline Model

The chosen baseline is a simple CNN. The architecture consists of three main components: convolutional layers with rectified linear unit (ReLU) activation, max pooling layers for down-sampling, and fully connected layers for feature mapping and classification. The input to the network is a single-channel, grayscale image resized to 128x128.

The model begins with three convolutional layers. The first layer uses 32 filters, followed by 64 in the second, and 128 in the third. Each convolutional layer is followed by a ReLU activation function, which introduces non-linearity, enabling the network to learn more complex representations of the input data. After each convolutional operation, a max pooling layer with a 2x2 kernel is applied, halving the spatial dimensions of the feature maps, while also helping retain the most relevant features and discarding the irrelevant details, contributing to the network's ability to generalize effectively.

The output of the final convolutional layer is a 3D tensor with dimensions 16x16x128, which is flattened into a 1D vector of size 32768. This vector is then passed through a sequence of three fully connected layers. The first fully connected layer reduces the dimensionality to 512 neurons, capturing high-level features from the input data. The second layer further reduces this to 256 neurons, compressing the learned features into a more compact representation. Finally, the output layer contains three neurons, each corresponding to one of the target labels: ACL tear, meniscus tear, and abnormality. No activation function is applied to the final layer, as the network outputs logits (unnormalized scores) for each class.

The model is trained using binary cross-entropy loss with logits (BCEWithLogitsLoss), which combines a sigmoid activation function and binary cross-entropy loss. This loss function is particularly well-suited for multi-label classification tasks, as it treats each class independently and calculates the probability of each label being present. By predicting the presence or absence of each injury independently, the model can account for cases where multiple injuries coexist within the same sample. This characteristic is essential for the knee injury dataset, where injuries are not mutually exclusive.

IV. Evaluation Metrics

To assess the performance of the baseline model, we used a comprehensive set of evaluation metrics, including accuracy, loss, and receiver operating characteristic (ROC) area under the curve (AUC) scores, for both the training and validation datasets across multiple epochs to track the model's learning progress.

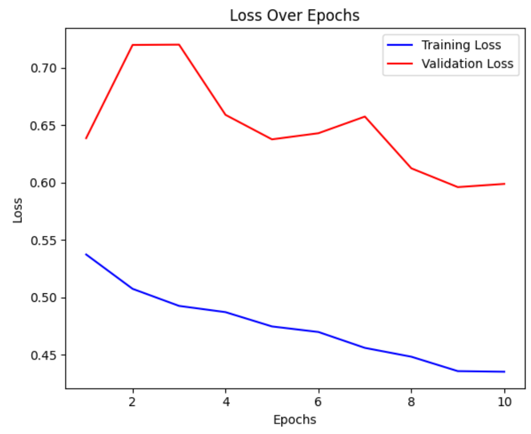

Fig. 1. Train and validation loss vs epoch.

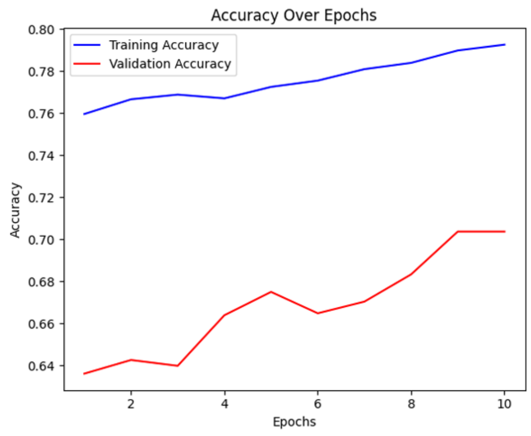

Fig. 2. Train and validation accuracy vs epoch.

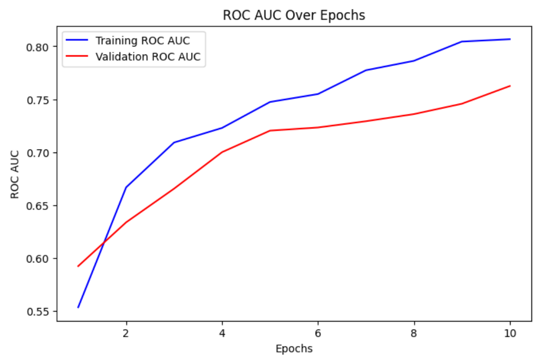

Fig. 3. ROC AUC vs epoch.

This simple model was also measured on the test dataset, where it obtained an accuracy of 0.7436, loss of 0.5125, and ROC AUC of 0.7825. We also visualized the results for each present combination of labels using a classification report:

TABLE I
CLASSIFICATION REPORT FOR LABEL COMBINATIONS

Label Combination	**Precision**	**Recall**	**F1-Score**	**Support**
Abnormalities	0.37	0.82	0.51	117
ACL, Abnormalities	0.30	0.33	0.32	63
Meniscus, Abnormalities	0.44	0.08	0.14	48
Meniscus, ACL, Abnormalities	0.41	0.18	0.25	66
No Issues	0.56	0.15	0.23	96
Accuracy			0.38	390
Macro Average	0.42	0.31	0.29	390
Weighted Average	0.42	0.38	0.32	390

The accuracy of 0.7825 is the per-label accuracy, measuring correctness per label and averaging it across all samples, whereas the accuracy of 0.38 shown in the classification report is the combined labels accuracy, where a sample is considered correct only if all predicted labels match exactly with the true labels.