Machine learning capstone project proposal.

Domain background:

Musculoskeletal conditions affect more than 1.7 billion people worldwide, and are the most common cause of severe, long-term pain and disability, with 30 million emergency department visits annually and increasing. So, Stanford ML group has come up with a dataset MURA (**mu**sculoskeletal **ra**diographs), which is a large dataset of bone X-rays of finger, wrist, elbow, forearm, hand, humerus, and shoulder studies. (Ref: https://arxiv.org/pdf/1712.06957.pdf).

My personal motivation is to build models to help in healthcare sector, which would identify most of false negatives which would remove untimely treatment and negligence out of the system. Accurately being able to predict the false positives would keep cost of health care down, by not subjecting the patients to a plethora of tests, which might not have been necessary to begin with and alleviate unnecessary stress on the patients.

Problem statement:

MURA, a large dataset of musculoskeletal radiographs containing 40,561 images from 14,863 studies, where each study is manually labeled by radiologists as either normal or abnormal. To evaluate models robustly and to get an estimate of radiologist performance, the Stanford group collected additional labels from six board-certified Stanford radiologists on the test set, consisting of 207 musculoskeletal studies. These 207 radiographs are considered as labeled data and will help in evaluating the performance of radiologist in labeling the rest of the radiographs. To keep the dataset manageable, we will consider the data for one study group, in my case -the wrist study.

Datasets and input:

As defined above, MURA is a dataset of musculoskeletal radiographs consisting of 14,982 `studies` from 12,251 `patients`, with a total of 40,895 `multi-view radiographic images`. Each `study` belongs to one of seven standard upper extremity radiographic `study types`: elbow, finger, forearm, hand, humerus, shoulder and wrist.

MURA dataset comes with `train` and `valid` folders containing corresponding datasets, `train.csv` and `valid.csv` containing paths of `radiographic images` and their labels. Each image is labeled as 1 (abnormal) or 0 (normal) based on whether its corresponding study is negative or positive, respectively , by the radiologists. Sometimes, these radiographic images are also referred as `views`.

The train and valid dir structure is as follows:
study_area/patientxxx/[study_negative|study_postive]/images[1-3]

Solution statement:

According to MURA paper:

The model should take as input one or more views for a study of an upper extremity. On each view, the model should predict the probability of abnormality. Then the overall probability of abnormality for the study is computed by taking the arithmetic mean of the abnormality probabilities output by the network for each image. The model makes the binary prediction of abnormal if the probability of abnormality for the study is greater than 0.5.

Benchmark Model

The one advantage of this data set is, it is baselined against a 169-layer convolutional neural network to detect and localize abnormalities. MURA paper states that:

"Our baseline uses a 169-layer convolutional neural network to detect and localize abnormalities. The model takes as input one or more views for a study of an upper extremity. On each view, our 169-layer convolutional neural network predicts the probability of abnormality."

Evaluation Metrics:

Accuracy (number of patients accurately classified/total number of patients for wrist study) could be a primary metric.

We could also use precision, recall, f1-score and also use the confusion metric to evaluate the model).

Project Design:

- Download the datasets from the MURA site, which would include views for all the study groups. As the data set should be <500MB zipped, select an area of study(wrist radiographs) and extract data out of the CSV files and train and valid dirs.
- Import the necessary libraries like pandas, numpy, matplotlib and scikit-learn, keras and tensorflow.
- Load datasets and categorize them into training files, training targets, validation files, validation targets and testing files and testing targets.
- Print the # of training, testing and validation files.
- Check the sanity of data, with undefined values.
- Check if the data needs transformation (size reduction and gray scaling) or augmentation.
  The paper says:

  "Before feeding images into the network, we normalized each image to have the same mean and standard deviation of images in the ImageNet training set. We then scaled the variable-sized images to 224×224. We augmented the data during training by applying random lateral inversions and rotations."

  Which means we need to scale the variable sized images to a standard size and apply augmentation as well.

- Check if bounding boxes are needed to isolate the image from the background.
- Building a model.
  The paper quotes:
  "We used a 169-layer convolutional neural network to predict the probability of abnormality for each image in a study. The network uses a Dense Convolutional Network architecture—detailed in Huang et al. (2016)—which connects each layer to every other layer in a feed-forward fashion to make the optimization of deep networks tractable. We replaced the final fully connected layer with one that has a single output, after which we applied a sigmoid nonlinearity."

“The weights of the network were initialized with weights from a model pretrained on ImageNet (Deng et al., 2009).”

Need to use a dense convoluted neural network. DenseNet is a fully connected neural network(Ref: https://arxiv.org/pdf/1608.06993.pdf), who’s output should be funneled through a final layer which outputs the probability of abnormality for the MURA datset. The output of the final layer funneled through a sigmoid function for predicting abnormality.

The model needs to be initialized with weights from a pretrained model on imageNet. Planning on using keras denseNet library and tensorflow for the backend processing. Need to check if a high number of layers as 169 can be run on the GPU on the local machine, else need to make a decision to switch to cloud computing services.

- The model needs to use modified Binary Cross Entropy Loss function as mentioned in the paper.
- Need to use an optimizer algorithm like adam(adaptive momentum estimation ,Ref: https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f) to minimize the loss.