Machine Learning Engineer Nanodegree

Capstone Project

Divyank Vijayvergiya
April 22nd, 2019

I. Definition

Project Overview

Bone Fracture is a problem that affects people all around the world. As computer are getting better at understanding images due to advances in computer vision and neural networks, the concept of Bone X-Ray detection which detects the upper extremity abnormality including the shoulder, humerus, elbow, forearm, wrist, hand and finger. **MURA** is one of the one of the largest public radiographic image datasets which has been used in this project.

Problem Statement

Given a dataset of X-Ray radiographs images, an algorithm needs to be detect an abnormalities in the bones and determines whether they are fractured or not automatically with better than human performance?

- Download and preprocess the MURA dataset.
- Train a classifier which determine if an X-Ray is abnormal or not.
- The final application is expected to be useful for detecting X-Ray whether bone is abnormal or not.

Metrics

For each image X of study type T in the training set, you must optimized the weighted binary cross entropy loss.

$L(X,y)=-w_{T,1} \cdot y \log p(Y=1|X)$
$-w_{T,0} \cdot (1-y) \log p(Y=0|X),$

where y is the label of the study, $p(Y = i|X)$ is the probability that the network assigns to the label i, $w_{T,1} = |N_T|/(|A_T| + |N_T|)$, and

$w_{T,0} = |A_T|/(|A_T| + |N_T|)$ where $|A_T|$ and $|N_T|$ are the number of abnormal images and normal images of study type T in the training set, respectively.

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 256×256. We augmented the data during training by applying random lateral inversions and rotations of up to 30 degrees.

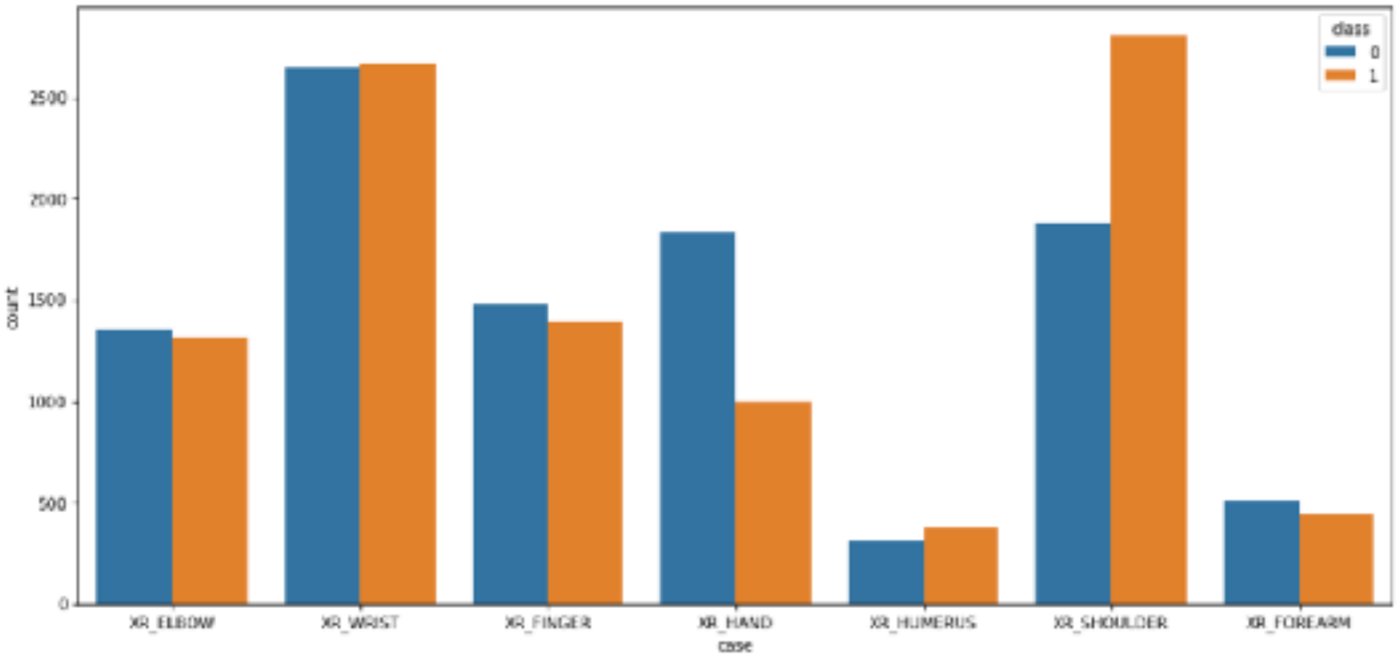
II. Analysis

Data Exploration And Visualization

Large high-quality datasets have played a critical role in driving progress of fields with deep learning methods . MURA, a large dataset of radiographs, containing 14,863 musculoskeletal studies of the upper extremity. Each study contains one or more views (images) and is manually labeled by radiologists as either normal or abnormal. MURA, contains 9,045 normal and 5,818 abnormal musculoskeletal radiographic studies of the upper extremity including the shoulder, humerus, elbow, forearm, wrist, hand, and finger. MURA is one of the largest public radiographic image datasets.

STUDY	TRAIN		VALIDATION		TOTAL
	Normal	Abnormal	Normal	Abnormal	
Elbow	1094	660	92	66	1912
Finger	1280	655	92	83	2110
Hand	1497	521	101	66	2185
Humerus	321	271	68	67	727
Forearm	590	287	69	64	1010
Shoulder	1364	1457	99	95	3015
Wrist	2134	1326	140	97	3697
Total no. of studies	8280	5177	661	538	14656

A dataset of musculoskeletal radiographs consisting of 14,863 studies from 12,173 patients, with a total of 40,561 multi-view radiographic images. Each belongs to one of seven standard upper extremity radiographic study types: elbow, finger, forearm, hand, humerus, shoulder, and wrist. Table summarizes the distribution of normal and abnormal studies. Each study was manually labeled as normal or abnormal by board-certified radiologists from the Stanford Hospital at the time of clinical radiographic interpretation in the diagnostic radiology environment between 2001 and 2012. The labeling was performed during interpretation on DICOM images presented on at least 3 megapixel PACS medical grade display with max luminance 400 cd/m2 and min luminance 1 cd/m2 with pixel size of 0.2 and native resolution of 1500 x 2000 pixels. The clinical images vary in resolution and in aspect ratios. We split the dataset into training (11,184 patients, 13,457 studies, 36,808 images), validation (783 patients, 1,199 studies, 3,197 images), and test (206 patients, 207 studies, 556 images) sets. There is no overlap in patients between any of the sets.



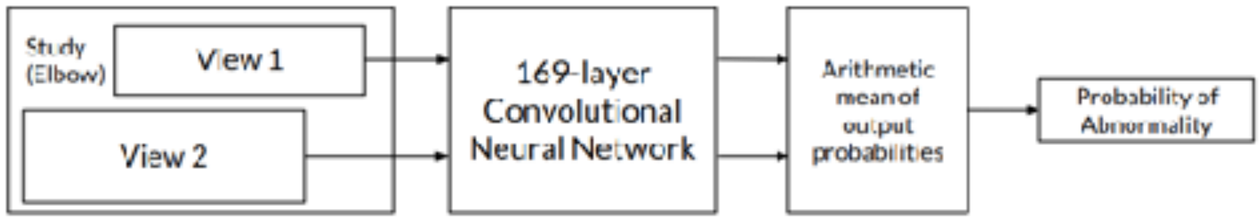
Histogram represents normal and abnormal distribution in each category.

Algorithms and Techniques

A deep learning algorithm will be developed using Tensorflow/Keras and will be trained with training data and validation data will be used to evaluate how model is doing. Specifically a CNN will be implemented in Tensorflow/Keras.The model takes as input one or more views for a study. On each view, Convolutional neural network predicts the probability of abnormality. We compute the overall probability of abnormality for the study 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.

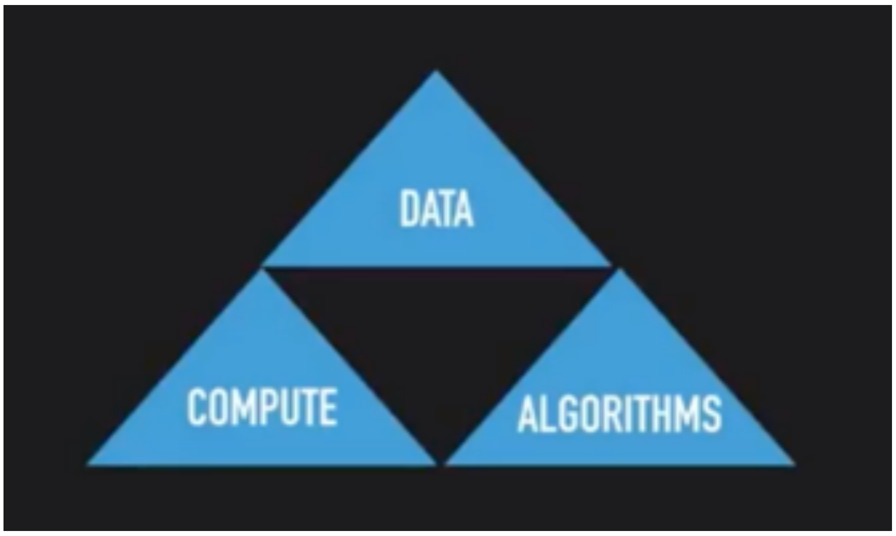
Benchmark

The model with public leaderboard with kappa score of 0.778 will be used as benchmark model. They train a 169-layer DenseNet baseline model to detect and localize abnormalities.


Study (Elbow)
View 1
View 2
169-layer Convolutional Neural Network
Arithmetic mean of output probabilities
Probability of Abnormality

III. Methodology

Data Preprocessing



- A lot of progress is in supervised learning = need for labelled data .
- Common example: Trained model on (x,y) pairs; predict probability of y_new given x_new.

MURA dataset comes with train, valid and test folders containing corresponding datasets, train.csv and valid.csv contain 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. Sometimes, these radiographic images are also referred as views.

Components of train and valid set

- Train set consists of seven study types namely:
    - XR_ELBOW
    - XR_FINGER
    - XR_FOREARM
    - XR_HAND
    - XR_HUMERUS
    - XR_SHOULDER
    - XR_WRIST
- Each study type contains several folders named like:
  Patient12110, patient12116, patient12122, patient12128 …
- These folders are named after patient ids, each of these folders contain one or more study, named like: study1_negative, study2_negative, study3_positive ..
- Each of these study contains one or more radiographs (views or images), named like: image1.png, image2.png…
- Each view (image) is RGB with pixel range [0, 255] and varies in dimensions.
- Merging the training path images and labeled images in a training dataframe
- Merging the Validation path images and valid labeled images in a testing dataframe

Implementation.

- 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.

- We compute the overall probability of abnormality for the study 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.
- 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 256×256. We augmented the data during training by applying random lateral inversions and rotations.
- Loss Function:

For each image X of study type T in the training set, we optimized the weighted binary cross entropy loss

$$L(X, y) = - wT ,1 \cdot y \log p(Y = 1|X) - wT ,0 \cdot (1 - y) \log p(Y = 0|X),$$

- Used LeakyRelu activation function to classify them properly.
- Four times I have Convo2D for errors loss.
- Used Dropout to overcome overfitting
- Early stopping has also used to overcome overfitting

Refinement

I have used Early Stopping algorithm to overcome overfitting and augmented the training data till 30 degrees so that model can learn each image precisely.
Used Used Dropout to overcome overfitting and used Sigmoid activation function in end so that output in between 0 and 1.

IV. Results

Model Evaluation and Validation

During development, a validation set was used to evaluate a model.

The final architecture and hyperparameters were chosen because they performed the best among the tried combinations.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 256, 256, 32)      320
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 256, 256, 32)      0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 128, 128, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 128, 128, 32)      9248
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 128, 128, 32)      0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 64, 64, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 64, 64, 64)        18496
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 64, 64, 64)        0
```

```
_________________________________________________________
________
max_pooling2d_3 (MaxPooling2 (None, 32, 32, 64)        0
_________________________________________________________
________
conv2d_4 (Conv2D)            (None, 32, 32, 128)
73856
_________________________________________________________
________
leaky_re_lu_4 (LeakyReLU)    (None, 32, 32, 128)       0
_________________________________________________________
________
max_pooling2d_4 (MaxPooling2 (None, 16, 16, 128)       0
_________________________________________________________
________
flatten_1 (Flatten)          (None, 32768)             0
_________________________________________________________
________
dense_1 (Dense)              (None, 256)
8388864
_________________________________________________________
________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________
________
dense_2 (Dense)              (None, 256)
65792
_________________________________________________________
________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________
________
dense_3 (Dense)              (None, 1)
257
=========================================================
========
Total params: 8,556,833
Trainable params: 8,556,833
Non-trainable params: 0
```

- The shape of the filters of the convolutional layer is 3*3.
- The first two convolutional layer learns 32 filters and third and fourth convolutional layer learn 64 and 128 filters respectively.

- The convolutional layers have stride of 1.
- The weights of the convolutional layers are initialised by sampling a normal distributions with a standard deviation of $10^{-4.}$
- The training run over 36,000 iterations.
- After 11 epochs training has finished to overcome overfitting.
- Evaluate model on testing set which has been created from merging of validation path and study labels.

Justification

For each experiment only the best model was saved along with their weights(a model only gets saved per epochs if it shows higher validation accuracy than the previous epoch)

I have got 87% accuracy in training set and approx 61% in validation and testing set and got a kappa score to 0.26360753294171424 which is not so good but its comparable with benchmarks model kappa score.

It can be seen that application is giving good accuracy over wrist and finger dataset but it’s not performing well over shoulder images.

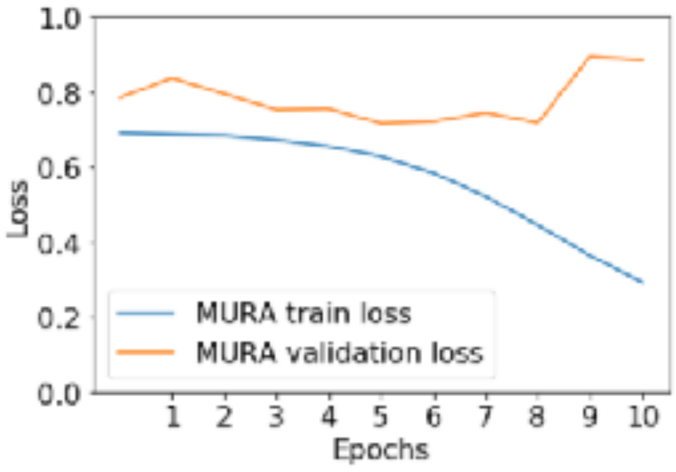
In summary, the application is useful in a limited domain, but to solve bigger problem(giving normal person a power to check their X-Ray whether it is normal or not).

V. Conclusion

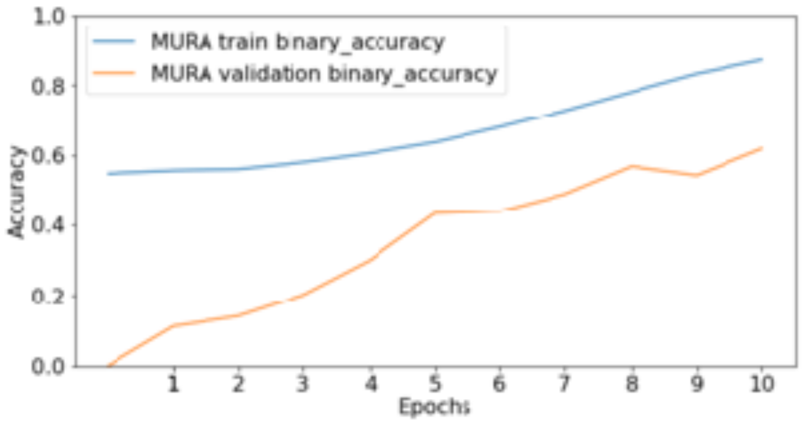
Free-Form Visualization

As I’ve recorded the accuracy and loss of models per epoch, the final model can be compared to the second best alternative. Here’s the accuracy/loss graph of the model with batch normalisation and augmentation of data upto 30 degrees.

Model loss



Model Accuracy



As we can see the training accuracy is near 90% in the diagram and loss is nearly 0.2. Similarly, the validation accuracy is nearly 60% and loss is around 0.8 as seen in the diagram. We also see the trend where the validation loss is keeps decreasing initially but after 10 epochs it starts increasing which is the sign of overfitting that's why I have used algorithm early stopping so that model can overcome overfitting.

The model performance or accuracy on the test set is also nearly 60% which is worse than the final model performance.

Reflection

For reaching into this end to end solution, I've tried to progressively to use very complex models to classify the images.
I tried to augment the training data till 30 degrees of rotation
Loading of dataset is most critical part for me Initially I was trying to import whole dataset which is giving me memory error because my CPU is not able to handle memory of this dataset. So, I import few sample images and trained my model over them.

I can improve my model by importing dataset using ‘Image Data Generator’ method and by loading images in batches.
I used Early Stopping algorithm to improve the model so that model do not overfit the data. Yes, According to me model and solution fits my expectations but kappa score can improve more in my model because in my benchmark model kappa score is 0.7 and I have got 0.26 which is far less than the final expectations but in model accuracy it is comparable with benchmark model. The Kappa statistic (or value) is a metric that compares an **Observed Accuracy** with an **Expected Accuracy** (random chance). The kappa statistic is used not only to evaluate a single classifier, but also to evaluate classifiers amongst themselves.

Improvement

Due to time and computational costing it is not possible for me to run more experiments using different approaches like transfer learning using VGG-16, RESNET, Inception V-3 for this dataset.

If I had used ImageDataGenerator method to import dataset then I can be able to import the whole dataset which will definitely improve the performance of model. In current model I am augmented the all images upto 30 degrees but if I divided the index by 2 and only augmented the images of those index then also it will increase the model performance. There are so many improvements which I can do if I have a Machine with more than 8GB Ram or using Gpu.

If I used final solution as the new benchmark, then yes better model are exists. Stanford model is the best model yet.