Automated Detection of Breaks and Fractures in X-Ray Bone Images

Maithreyi Gopalakrishnan, Jaymee Sheng, Maurizio Valesani

Abstract—The focus of this research is to develop a model to accurately classify normative vs. anomalous images of X-ray data of the humerus from many different patients. We co mpare the accuracy of four different supervised and unsupervised models: logistic regression, logistic regression with L2 regularization, k-nearest neighbors, and a convolutional neural network (CNN). Upon comparing accuracy of these models on the test data, we find that k-nearest neighbors yields the highest accuracy. We reason that this could be due to a combination of the CNN taking in different data inputs from the supervised learning approaches (un-normalized and normalized respectively), the reduction in noise, and most accurate identification of relevant data points.

Index Terms—Classification, k-nearest neighbors, logistic regression, regularization, convolutional neural network, pass, machine learning, X-ray image, humerus

1 Introduction

Currently, when a radiologist diagnoses a break or fracture in a bone, he or she simply does so by visually inspecting radiographs. This method could be prone to error, especially when the bone being imaged has multiple or smaller fractures. The motivation of this project is to utilize machine learning techniques to automatically detect breaks and fractures in an X-ray image of a bone. This can serve as an initial flagging system or as a secondary check to verify that the radiologist has identified all breaks and fractures. Eventually, such an algorithm can be extended to identify breaks on a 3-dimensional scale, i.e. with CT scan data.

We compared multiple methods to identify anomalies in X-ray images, using two different approaches: first, we input normalized histograms of images to perform logistic regression (with and without regularization) and *k*-nearest neighbors, which we anticipate will separate images showing fractures and/or anomalies from healthy ones; second, we input images directly into a convolutional neural network, which we trained to classify anomalous (positive) vs. normative (negative) data.

2 Related work

The medical field is extremely active as far as research on automation of image recognition goes, as more and more images are acquired every day and requiring extremely specialized personnel to check all of them manually is a slow and costly process. Specifically, radiology is one of the branches of medical science that shows the most applications of machine learning techniques, as mentioned by Wang and Summers who focus their attention on "*medical image segmentation, registration, computer aided detection and diagnosis, brain function or activity analysis and neurological disease diagnosis from fMR images, content-based image retrieval systems for CT or MRI images, and text analysis of radiology reports using natural language processing (NLP) and natural language understanding (NLU)*" [17].

The goal of the industry in this regard is to move towards intelligent systems able to classify most images autonomously, e.g. detecting whether a tumor is present and whether it is benign or malign, or identifying the severity and type of skeletal damage in specific parts of the body, thus having to rely on expert judgement only in the limited set of ambiguous edge cases. As there are many applications, we decided to focus our attention on the existing literature on the analysis of long bones, as it fits our target.

Yap et al., similarly to other researchers in this field, make use of Bayesian classifiers and SVMs. SVMs in particular are mentioned multiple times in the literature as they allow, thanks to the use of kernels, the search of hyper-planes in higher dimensions that can act as decision boundaries for the problems at hand, which is in line with the complexity of automatic fracture detection [5][8][9][10][11][13][16][18]. Another strategy exploited multiple times as mentioned in the existing literature is the use of Convolutional Neural Networks (CNNs) [17]: convolutional layers are well suited to deal with images as, contrary to standard neural networks, they do not require the flattening of the image and thus can "*successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters*" by limiting the amount of parameters used and reusing the computed weights [14].

3 Dataset and features

The dataset used for this project can be accessed at https://stanfordmlgroup.github.io/competitions/mura/

To simplify the scope of the problem at hand, we focused on data from the humerus only, ignoring the other types of bones. The resolutions of these images varied, but one dimension of the images was generally 512 pixels, and the

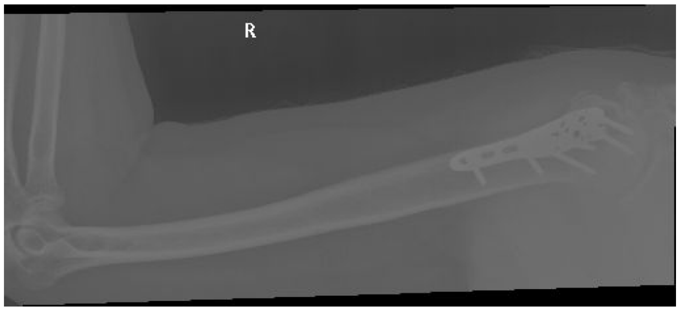


Fig. 1. Example image from training set.

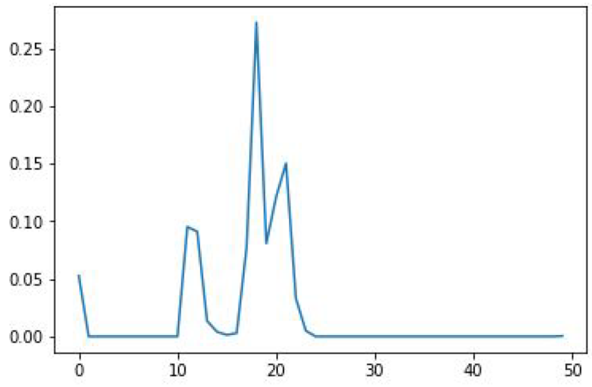


Fig. 2. Normalized histogram of example image from figure 1.

other dimension ranged from 200-600 pixels.

There was very limited pre-processing of the data for the neural network, as the only operation performed consisted in converting all images to the same size by padding with pixels of the same colour of the mode of the image (this solution was favored over using black pixels as, the background being non perfectly black, this alternative would have generated very inconsistent edges that could negatively affect performance).

For logistic regression and k-nearest neighbors, the pre-processing consisted of converting each image into a normalized histogram so it could be directly compared with histograms of other images. An example of an image from the training set and its normalized histogram is shown in figures 1 and 2.

4 Methods

4.1 Supervised learning

4.1.1 Logistic regression

To establish baseline, we implemented a logistic regression model that maximizes the log likelihood of seeing the labels given the histogram data of the images in the training set and the parameters for the logistic function. We then considered regularization using the L2 norm with the following loss function:

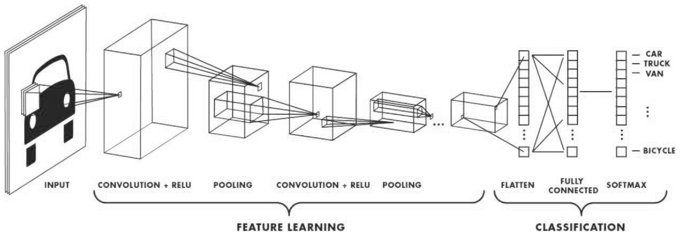


Fig. 3. Example structure of a CNN for multi-class classification [14].

$$J(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right) + \lambda||\theta||^2 \quad (1)$$

This effectively penalizes features with large norms and shrinks the coefficients which helps prevent over-fitting to the training data.

4.1.2 K-nearest neighbors

The idea behind k-nearest neighbors is simple: given a feature vector X, find its k nearest neighbors and classify it by a majority vote. For example, suppose we pick the 10 nearest neighbors to classify a new image. In our case, this is done by computing pairwise Euclidean distances between the histogram of the new image and the histograms of the images in the training set, which are then used to find the closest 10 training examples. We can then calculate the mean of the labels of those 10 examples and classify the new image as abnormal if the mean is greater than 0.5, i.e. if more than half of its 10 nearest neighbors are abnormal.

4.2 Convolutional neural network

The final strategy that we exploited to classify the images and separate the ones showing healthy humeri from the ones with broken/non-nominal bones was a Convolutional Neural Network (CNN). The advantage of CNNs over regular neural networks when dealing with images resides in the ability of convolutional layers to operate without needing to flatten the image. Thus, it is possible to retain all the information related to the spatial positioning of the different pixels in the image, which allows us to avoid fully connected dense layers. This means that the number of parameters and weights the algorithm needs to learn in order to extract characteristic features is notably reduced, resulting in improved performance. Examples of a typical CNN structure for a multi-class classification task and a regular neural network are shown in figures 3 and 4.

In addition to convolutional layers, CNNs make also use of pooling layers in their feature learning section. Pooling layers are used as the feature maps obtained by applying convolutional layers are sensitive to the location of the features in the input image. Pooling layers address this sensitivity by down-sampling the feature maps. This

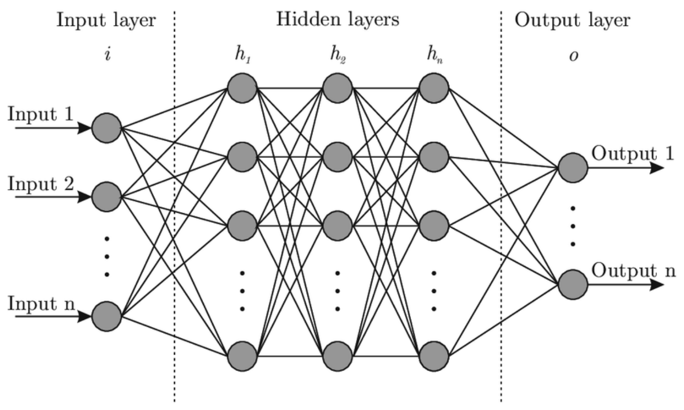


Fig. 4. Example structure of a neural network with dense, fully connected layers [15].

TABLE 1
Summary of the results for different methods.

Model	Accuracy
Logistic regression	0.61
Logistic regression with L2 regularization	0.64
k-nearest neighbours ($k = 10$)	0.67
Convolutional neural network	0.60

reduction in the spatial size of the representation helps further reduce the amount of operations and the parameters to be learnt. Usually *max pooling* is the go-to approach, which retains the maximum value among the pixels under consideration. Other alternatives are also available, such as *average pooling* in which the average value of the pixels under consideration is retained. *Max-pooling* was used in this study.

The classification section of the CNN makes use of dense layers, as the ones described above, following a flattening layer that acts as the bridge between the two sections by converting a three-dimensional tensor (2D image with more than one channel as input, with the third dimension increasing in size according to the filters used in the convolutional layers) into a one-dimensional tensor (a vector). In our case of binary classification, the output layer consists of a single neuron, returning the probability of the sample being positive (in the case of multi-class classification, n neurons are used where n is the number of classes).

5 Results

A summary of results obtained for each model[1] in terms of accuracy is shown in table 1. Note that accuracy was selected as the measurement of performance under the assumption that false positives and false negatives have the same importance and no error is inherently worse than the other.

1. The code is accessible at https://drive.google.com/open?id=1ubzTJ42CLeY7E2UOwHh1YIlMu8rdjW5x.

5.1 Supervised learning

5.1.1 Logistic regression

Our baseline logistic regression model yielded an accuracy of 0.61 on the test set. Performance was improved to 0.64 by adding a L2 regularization term to the objective function with a λ value of 0.001. We experimented with different values of λ and saw trade-offs between bias and variance. A large λ (e.g. $\lambda = 0.01$) reduces the number of features that are relevant and thus leads to smaller variance and faster training, but also results in worse fit and does not generalize very well to new data. A λ that is too small (e.g. $\lambda = 0.00001$), on the other hand, does little in terms of regularizing the model even though it produces a better fit on the training set, but is also less generalizable. The extreme case of the latter is when $\lambda = 0$, i.e. the baseline model, where the loss on training data was indeed the lowest across all λ's but the accuracy on the test set was also the lowest. Through trial and error, we found that $\lambda = 0.001$ is the sweet spot that led to the best performance on test data as well as reasonable run-time.

5.1.2 K-nearest neighbors

For k-nearest neighbors, we found that the optimal k is 10, which yielded an accuracy of 0.67. As shown in figure 5, accuracy on the test set increased drastically for small values of k and began to decline for k greater than 10, albeit with quite a bit of volatility, before eventually stabilizing around 0.61 at $k = 90$.

Note that we could have chosen a different threshold for classification. So instead of a simple majority, we could require three fourths of the k nearest neighbors to be positive, for example, in order to classify a new image as positive. The choice of this ratio given fixed k is essentially a trade-off between false positives and false negatives. A higher threshold reduces the chance of false positive error but also increases the chance of false negative error, and vice versa for a lower threshold. At $k = 10$, we found that a threshold of 0.5 gave the highest accuracy, our main evaluation metric. However, if we had a different or secondary objective such as minimizing false negatives, i.e. erring on the side of being more generous when identifying anomalies, which is not unreasonable given we want the algorithm to recognize fractures that may not be caught by the human eye, we might choose a slightly lower threshold without giving up too much accuracy.

5.2 Convolutional neural network

5.2.1 Structure

As for the CNN, we tried different approaches both in terms of minimal pre-processing and the structure of the CNN itself.

Regarding pre-processing, the only operation performed consisted in padding the images to convert them to the same size, as required by the CNN. Initially, we tried padding with black pixels. However, this would result in sharp edges appearing in those images whose background is not perfectly black. Thus, we tried implementing padding

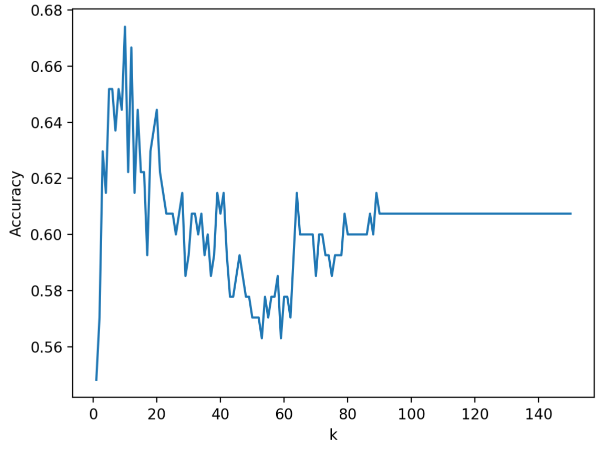


Fig. 5. Classification accuracy for k-nearest neighbors.

using either the minimum value of the pixels in the image or the mode (under the assumption that the mode would correspond to the background color); no significant difference was observed between the two approaches in terms of performance, so we elected to use the last implementation with the mode. An extract from the code used to import and pad images is shown below.

```
for i in os.scandir(path_positive):
  if len(data) >= threshold[0]:
    break
  if i.is_dir():
    for j in os.scandir(i.path):
      if len(data) >= threshold[0]:
        break
      if j.is_dir():
        for k in os.scandir(j.path):
          if k.is_file:
            im = cv2.imread(k.path)
            ones = np.ones(shape, dtype=np.int32)\
                * stats.mode(im, axis = None)[0]
            ones[:im.shape[0], :im.shape[1],\
                :im.shape[2]] = im
            data.append(ones)
            labels.append(1)
            if len(data) >= threshold[0]:
              break
```

Another aspect to notice about the training set is related to how skewed it is: of the ~ 1300 samples available, 83% were positive. We decided to add a threshold when reading images in to balance positive and negative samples, in order to avoid having a CNN always predicting *positive*.

As far as the CNN is concerned, the initial structure that was tested consisted of thirteen layers. After the input convolutional layer, four more convolutional layers followed with four pooling layers in between; then a flattening layer and three dense layers, with the last one consisting of a single neuron (as already discussed in section 4.2). This structure, however, resulted in a noticeable overfit on the training data, with loss on the validation set initially fluctuating wildly and then increasing at every epoch. To address this problem different strategies were tested and implemented. Dropout layers were added to compensate for the overfit. After different tests, we settled on a dropout rate of 0.1 for the ones after convolutional layers and 0.4 for the ones after dense layers as it seemed to provide the best results. The size of the CNN was also reduced to address the issue of overfit, limiting its feature learning section to three convolutional layers, each followed by a pooling layer and a dropout layer (the overall number of layers increased but reducing the amount of convolutional layers in favour of dropout layers helped counteract the overfit). This helped improve the accuracy on the test dataset from $\sim 35\%$ to $\sim 40\%$. The final structure of the CNN is shown in figure 6.

The optimizer that performed best was *Adam* with a learning rate of 0.001.

Finally, multiple tests were performed on the parameters of the other layers to achieve the best performance.

5.2.1.1 Convolutional layers: The ReLU function was used as activation function in all convolutional layers.

$$f(x) = x^+ = \max(0, x) \tag{2}$$

The stride was left at the default value of 1, meaning the convolution windows shift one pixel at a time. Padding was set at *'same'* to maintain the size of the image. The kernel size was set to $(3, 3)$, as it provided a good balance between accuracy of the results and computational performance.
The number of filters was steadily increased: 5 at the first convolutional layer, 7 at the second, and 10 at the final one.

5.2.1.2 Dense layers: Dense layers used the ReLU function as well, apart form the last one where the sigmoid function was used, since the goal is binary classification.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

Two layers of 100 neurons were used right after the feature learning section of the CNN, followed by a 2-neuron layer and a single-neuron one as output.

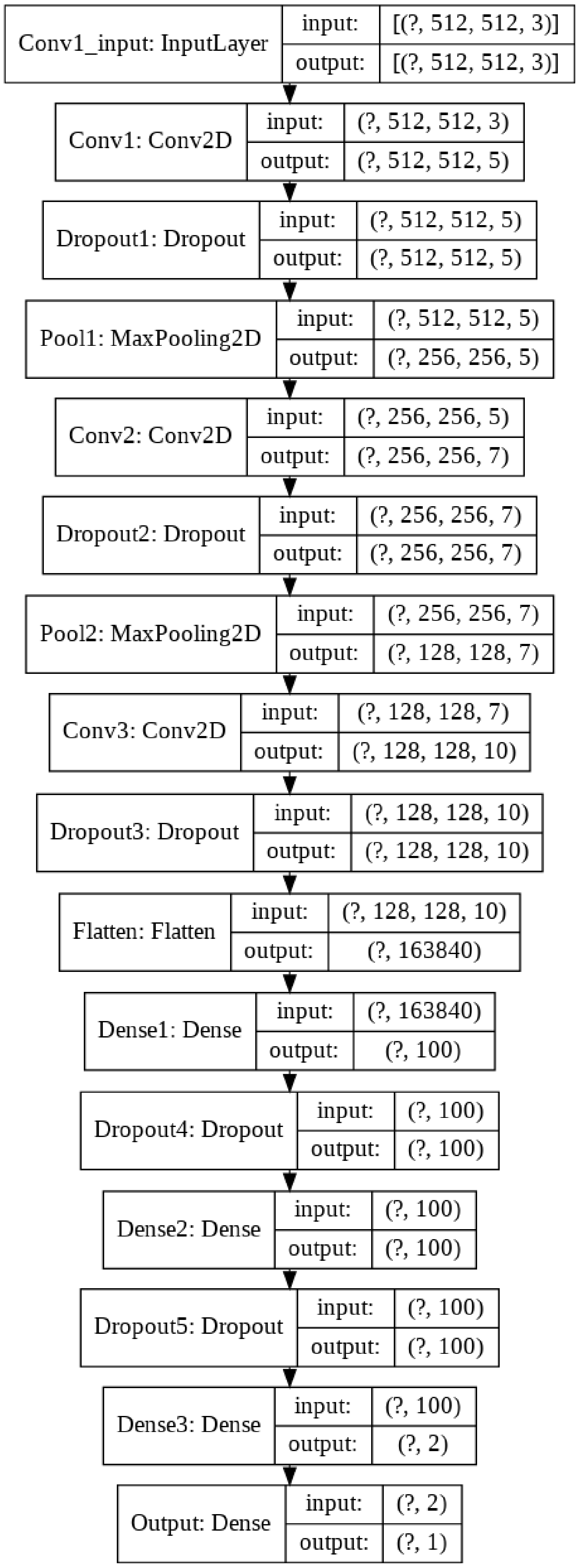
5.2.2 Performance

The accuracy of the CNN on the training and validation set varying with the epochs of training is shown in figure 7. It is evident how overfit is still an issue for this algorithm as accuracy on the training set is very close to 1, while on the validation set approaches 70%.

On the test set, the algorithm achieved an accuracy of 60.01% with the following confusion matrix:

$$\begin{bmatrix} 72 & 76 \\ 39 & 101 \end{bmatrix}$$

The obtained accuracy is definitely lower than what could be actually achievable. One of the main reasons for this result is the inconsistency of the images fed to the CNN. It would be extremely useful to perform a more aggressive form of pre-processing on the images before feeding them to the network. Furthermore, applying data augmentation



Conv1_input: InputLayer	input:	[(?, 512, 512, 3)]
	output:	[(?, 512, 512, 3)]
Conv1: Conv2D	input:	(?, 512, 512, 3)
	output:	(?, 512, 512, 5)
Dropout1: Dropout	input:	(?, 512, 512, 5)
	output:	(?, 512, 512, 5)
Pool1: MaxPooling2D	input:	(?, 512, 512, 5)
	output:	(?, 256, 256, 5)
Conv2: Conv2D	input:	(?, 256, 256, 5)
	output:	(?, 256, 256, 7)
Dropout2: Dropout	input:	(?, 256, 256, 7)
	output:	(?, 256, 256, 7)
Pool2: MaxPooling2D	input:	(?, 256, 256, 7)
	output:	(?, 128, 128, 7)
Conv3: Conv2D	input:	(?, 128, 128, 7)
	output:	(?, 128, 128, 10)
Dropout3: Dropout	input:	(?, 128, 128, 10)
	output:	(?, 128, 128, 10)
Flatten: Flatten	input:	(?, 128, 128, 10)
	output:	(?, 163840)
Dense1: Dense	input:	(?, 163840)
	output:	(?, 100)
Dropout4: Dropout	input:	(?, 100)
	output:	(?, 100)
Dense2: Dense	input:	(?, 100)
	output:	(?, 100)
Dropout5: Dropout	input:	(?, 100)
	output:	(?, 100)
Dense3: Dense	input:	(?, 100)
	output:	(?, 2)
Output: Dense	input:	(?, 2)
	output:	(?, 1)

Fig. 6. Final structure of the CNN.

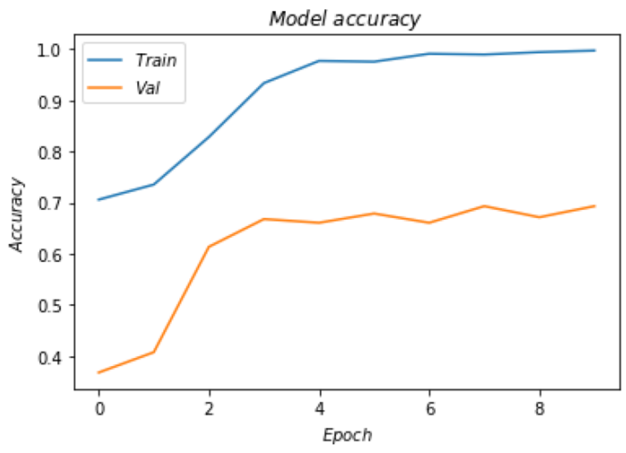


Fig. 7. Accuracy of the CNN on training and validation sets.

(rotating images, flipping them, and slightly modifying them in general) to increase the size of the training set should be one of the tasks to cover in future work in order to improve performance.

6 Conclusion

The problem statement was to explore the accuracy of different supervised and unsupervised models that classify normative vs. anomalous X-ray images of humeri (anomalous being humeri that are broken, fractured, or have implants), given labeled training and validation images. We pre-processed the images to extract normalized histograms of each image that could be fed into the supervised learning algorithms. Ultimately, we evaluated the accuracy of logistic regression, logistic regression with regularization, k-nearest neighbors, and a CNN. We found that, contrary to our original hypothesis that the CNN would perform best, k-nearest neighbors yielded the most successful results. This algorithm didn't require as much training data as the CNN to produce a robust model, and struck a balance between noise reduction and identifying the most relevant data for this particular problem. Additionally, our results told us that the pre-processing step of normalizing the images may have helped us achieve higher accuracy.

For future work, we would perform pre-processing on images prior to inputting them into the CNN. We would also explore more pre-processing steps, such as adding masks to the images and applying well-defined transforms, to make the currently extremely varied data more uniform in contrast, orientation, and scale. This would make feature extraction simpler and would consequently significantly improve the accuracy of our models across the board.

7 Contribution

Maithreyi found and analyzed the data set, developed the problem statement, and performed all pre-processing steps to convert images to be easily input into supervised learning algorithms. Jaymee worked on the logistic regression, logistic regression with regularization, and k-nearest neighbors algorithms and tuned the models to improve accuracy on test data. Maurizio developed the CNN and determined the accuracy of the final CNN on the test data.

References

[1] Martin Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: http://tensorflow.org/.

[2] Mahmoud Al-Ayyoub, Ismail Hmeidi, and Haya Rababah. "Detecting Hand Bone Fractures in X-Ray Images." In: *JMPT* 4.3 (2013), pp. 155–168.

[3] Hum Y. Chai et al. "Gray-level co-occurrence matrix bone fracture detection". In: *American Journal of Applied Sciences* 8.1 (2011), p. 26.

[4] François Chollet et al. *Keras*. https://keras.io. 2015.

[5] Joshua Congfu He, Wee Kheng Leow, and Tet Sen Howe. "Hierarchical classifiers for detection of fractures in x-ray images". In: *International Conference on Computer Analysis of Images and Patterns*. Springer. 2007, pp. 962–969.

[6] Seong-Hoon Kim et al. "X-ray image classification using random forests with local binary patterns". In: *2010 International Conference on Machine Learning and Cybernetics*. Vol. 6. IEEE. 2010, pp. 3190–3194.

[7] Thomas M. Lehmann et al. "Automatic categorization of medical images for content-based retrieval and data mining". In: *Computerized Medical Imaging and Graphics* 29.2-3 (2005), pp. 143–155.

[8] Sher Ee Lim et al. "Detection of femur and radius fractures in x-ray images". In: *Proc. 2nd Int. Conf. on Advances in Medical Signal and Info. Proc*. Vol. 65. 2004.

[9] Vineta Lai Fun Lum et al. "Combining classifiers for bone fracture detection in X-ray images". In: *IEEE International Conference on Image Processing 2005*. Vol. 1. IEEE. 2005, pp. I–1149.

[10] S. K. Mahendran and S. Santhosh Baboo. "An enhanced tibia fracture detection tool using image processing and classification fusion techniques in X-ray images". In: *Global Journal of Computer Science and Technology* 11.14 (2011), pp. 22–28.

[11] A. Mueen, Sapiyan Baba, and Roziati Zainuddin. "Multilevel feature extraction and X-ray image classification". In: *Journal of Applied Sciences* 7.8 (2007), pp. 1224–1229.

[12] Tian Tai Peng et al. "Detection of Femur Fractures in X-ray images". In: *Master of Science Thesis, National University of Singapore* (2002).

[13] Md Mahmudur Rahman, Prabir Bhattacharya, and Bipin C. Desai. "A framework for medical image retrieval using machine learning and statistical similarity matching techniques with relevance feedback". In: *IEEE transactions on Information Technology in Biomedicine* 11.1 (2007), pp. 58–69.

[14] Sumit Saha. *A Comprehensive Guide to Convolutional Neural Networks*. Ed. by Towards Data Science. 2018. URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[15] Lavanya Shukla. *Designing Your Neural Networks*. Ed. by Towards Data Science. 2019. URL: https://towardsdatascience.com/designing-your-neural-networks-a5e4617027ed.

[16] Guangjian Tian, Hong Fu, and David Dagan Feng. "Automatic medical image categorization and annotation using LBP and MPEG-7 edge histograms". In: *2008 International Conference on Information Technology and Applications in Biomedicine*. IEEE. 2008, pp. 51–53.

[17] Shijun Wang and Ronald M. Summers. "Machine learning and radiology". In: *Medical image analysis* 16.5 (2012), pp. 933–951.

[18] Dennis Wen-Hsiang Yap et al. "Detecting femur fractures by texture analysis of trabeculae". In: *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.* Vol. 3. IEEE. 2004, pp. 730–733.