**Liver Lesion Detection Using Feed-Forward Backpropagation (FFBP) Neural Networks**

Sumedh Guha, Jay Huang, Erin Murnane, & Raghav Ramachandran

Johns Hopkins University Engineering Program for Professionals | Neural Networks | Spring 2019

**Executive Summary**

A current trend in medical research is the development of universal Computer-aided Detection (CAD) frameworks capable of detecting anomalies in medical imagery. One example of successful CAD research is The National Institute of Health, Clinical Center application of Regional Convolutional Neural Networks (RCNN) on CT scans for lesion detection. Their RCNN-based CAD framework accurately detects lesions at 77.31% with 3 False Positives (FP) per image or 81.1% with 5 False Positives (FP) per image1.

Prior to development of this RCNN-based CAD framework, The National Institute of Health, Clinical Center investigated how lesions are traditionally identified by radiologists. An evaluation criteria for solid tumors, RECIST, is used by radiologists to identify and label anomalous features in CT scans. The radiologists will bookmark the anomalous feature with a bounding box for further review. Once a lesion is confirmed, the bookmarked CT scans are stored hospitals’ picture archiving and communication systems (PACS)1. Some examples of lesions are lung nodules, enlarged lymph nodes, and liver lesions.

The National Institute of Health, Clinical Center leveraged PACS in order to create a *DeepLesion* dataset. *DeepLesion* contains 32,735 lesions in 32,120 CT slices from 10,594 studies of 4,427 unique patients1. The CT slices are diverse, consisting of bone, abdomen, mediastinum, liver, lung kidney, soft tissue, and pelvis scans. The RCNN-based CAD framework draws from *DeepLesion* for training, validation, and test datasets. The dataset has been released in order to encourage other researchers to develop CAD frameworks, mine and study the relationship between different types of lesions, and more accurately and automatically measure sizes of all lesions6. This research has major implications to cancer research such as the ability to assess whole body cancer burden on patients.

The paper applies a Feed Forward Backpropagation (FFBP) neural network classification on a subset of *DeepLesion* liver CT scans. The paper seeks to determine optimal threshold value, training procedures, activation functions, and tradeoffs between false negatives and false positives within the FFBP framework. The baseline FFBP topology multilayer: two hidden layers with 7 nodes each and one output layer with one node. Grey scaled CT images are reduced to 32 by 32 subsets and transformed to a vector of pixel values. The vector of pixel values are inputs to FFBP neural network. Random weights and biases promote robustness and the activity value is a summation of the product of input pixel values and weights in addition to bias.

Variations of the the baseline FFBP model was explored by altering type of activation function used distorting the training set using various types of noises, and changing elements of the training method. Performance was evaluated based on error, accuracy, sensitivity, specificity, number of false positives, and runtime. When comparing the sigmoid function and hyperbolic tangent activation functions, the hyperbolic tangent produced more accurate results, however the sigmoid activation function had greater specificity and lower sensitivity. When Gaussian, Gamma, and mixed noise was applied to training dataset, Gamma noise produced a more accurate model and the number of false positives was smallest when Gaussian noise was applied. While reducing the training step and using a termination threshold runtime increased markedly, the accuracy and sensitivity improved significantly.

The training, validation, and test dataset preparation method is described in detail in Section II A. The baseline FFBP topology and performance of model variations are also described in Section II A - D. An analysis of the performance and future work is described in Section III. FFBP model development was created in Python and can be found on GitHub: <https://github.com/sumedhguha/DeepLesionClassifiers>

1. **Introduction**

Cancer is a collection of related diseases in which the body’s cells divide without stopping, spread into surrounding tissues, and form growths called tumors that are often solid (National Cancer Institute). Early detection of abnormal tissue and tumors is important for effective treatment and improved outcomes for patients. In cancer screenings, doctors may discover lesions and tumors. Lesions are any types of physical abnormalities and tumors are abnormal growth of a specific tissue or group of tissues. Examples of lesions include lung nodules, liver lesions, enlarged lymph nodes, and bone lesions1.

Computed tomography (CT) scans are medical diagnostic images used by doctors during cancer screenings. CT scans are a combination of X-ray images taken from different angles that provide cross-sectional images of bones, organs, and soft tissue within a patient. CT scans must be interpreted by radiologists, who mark and measure findings of interest. Interpreting and annotating CT scans can be complex for radiologists as exact locations and measurements are important to determine if tumors are growing.

For our project, a Feed Forward Backpropagation (FFBP) neural network is used to identify sub regions of CT images and determine whether the sub region contains a lesion or not. This determination of the existence of a lesion in a sub-region of an image is advantageous in locating a lesion within a full CT image. To determine the location of a lesion, one would recursively explore sub regions to determine which sub region contained a lesion and its position relative to the full CT image. Since locating lesions is a significant task for radiologists, this neural network can augment and support radiologists in determining whether a lesion exists in a sub-region of a CT image.

The dataset of 32,120 CT slices was compiled by the National Institutes of Health (NIH) from 4,427 unique patients. The dataset includes CT images of bone, abdomen, mediastinum, liver, lung, kidney, soft tissue and pelvis. For our project, 20 images of the liver were selected. Liver images are an optimal choice due to high noise-to-contrast ratios of liver lesions in comparison to lesions in other organs of the body. Images with bounding boxes (boxes identifying the area of the lesion) of size less than 32 by 32 pixels. The reasoning for 32 by 32 subregions input images is explained in detail in Section II A.

1. **Approach**
   1. **Data Preparation and Baseline Model**

For our baseline model, a FFBP neural network was implemented. The FFBP neural network is one of the first and simplest neural networks implemented. It provides a good baseline for building on top of. A single layer perceptron can take both continuous and binary inputs. They take in input values, compute a weighted sum of the inputs, subtract threshold and pass the resulting activity value through an activation function. The perceptron is most capable for separating a search space into two half spaces for classification. Multilayer perceptrons contain one or more hidden layers between the input and output layers. They can combine multiple hyperplanes to cover nonlinear classification regions. Two layer perceptrons can cover unbounded and bounded convex regions. This involves the intersection of multiple hyperplanes formed via the interactions of inputs and nodes in the first layer of the perceptron. Three layer perceptrons can form more complex regions, partitioning the search space into hypercubes, n-dimensional square regions which are combined for classification. Since three layer perceptrons are powerful enough to model almost any function, a three layer perceptron will be our baseline.7

The number of inputs to the input layer will be 1024: the program collects 32x32 image array samples from the 42 lesion images available. The number of nodes in the output layer will be 1 which will indicate the probability of classifying the image has having or not having a lesion. There are plenty of rule-of-thumbs, but no clear-cut method for determining the number of nodes in the hidden layers. Increasing the number of hidden layers and nodes will require more computational power, which may affect model convergence. In the study detecting breast cancer and thyroid disease, the authors found the number of nodes decreased the detection accuracy of the multi-layer perception. The suggested number of nodes for the sigmoid and hyperbolic tangent activation functions was 7 and 5, respectively (Appendix A). As number of nodes increased, accuracy performance decreased by a factor of 10. A similar study using FFBP for mammary gland images found best disease detection accuracy using 5-7 nodes per layer and random weights and bias4. Randomly initialized weights and biases is a common approach across various fields of image classifications FFBP neural networks such as aerospace, automotive, materials, manufacturing, petroleum, robotics, and communication5. Per these suggestions, our model is initiated with random weights, biases, and 7 nodes per layer. A rough determination of ideal number of nodes was considered by evaluating confusion matrices from multiple iterations. Please refer to Figure 2.

The step size is a hyperparameter for which there is no hard and fast rule for how to come up with the value. Again, the baseline step size was determined empirically. From some experimentation, a learning rate of 1.1 was established. Please refer to Figure 3.

The image values are between 0 and 255, hence, they may result in the activation function for input layer nodes being close to 1 for most of the image values. To resolve this issue, a normalization scheme was determined where 128 is subtracted from the values. The normalization results in negative values which resulted in overflow errors in the denominator. To address this, two formulas were made for the sigmoid function: one for dealing with negative input values, another for dealing with positive input values.

Early stopping time is a standard approach for terminating neural network training.8,9 It is easy to comprehend and to implement. The neural network is trained until the performance doesn’t improve. For our baseline, a metric was chosen to determine improvement. This metric is the difference between the output node of our neural network and the desired classification, 1.0 being an image containing a lesion, 0.0 being an image not containing a lesion. For classification, 0.5 is the boundary between classification of an image array as lesion or non-lesion. The dataset prior to splitting for training, validation, and testing is 42 images. The following training-validation-testing dataset split procedure is used:

1. Assign training, validation, and testing datasets. A 2-to-1 proportion between training and validation sets seems to be a desirable ratio. This proportion was determined empirically as well. The dataset is split randomly for 22 training, 10 validation, and 10 testing examples. Please refer to Figure 4.
2. Train on the training set using an online approach, evaluate the model based off of the validation examples.
3. Finish training once the validation example average error stops improving. Improvement is seen as the decrease in the metric value between successive training and validation epochs across the validation dataset, calculating the average error. A patience value of 10 epochs is used to account for the possibility that the function constantly improves after all epochs. The patience value was determined via some experimentation.
4. Use the weights in the final training as the final weights for the model.
5. Measure computation time and confusion matrices for getting accuracy, specificity, sensitivity, number of false positives, and runtime values.

|  | *Actual lesion (positive)* | *Actual non-lesion (negative)* |
| --- | --- | --- |
| *Predicted lesion (positive)* | True Positive (TP) | False Positive (FP) |
| *Predicted non-lesion (negative)* | False Negative (FN) | True Negative (TN) |

*Figure 1: Confusion matrix description for calculating performance metrics*

1. Accuracy = (TP+TN) / (TP+FP+FN+TN)
2. Sensitivity = TP / (TP+FN)
3. Specificity = TN / (TN+FP)
4. False Positives = FP

|  |  |
| --- | --- |
|  |  |

*Figure 2: Studied ROC curves of FFBP neural networks using 22 training, 10 validation, 10 testing for 10 maximum iterations, threshold of 0.5 between lesion and non-lesion image arrays. Step size of 1.1. Layers of size 7 seem to perform best in comparison to layers of size 5, 6.*

|  |  |
| --- | --- |
|  |  |

*Figure 3: Studied ROC curves of FFBP neural networks using 22 training, 10 validation, 10 testing for 10 maximum iterations, threshold of 0.5 between lesion and non-lesion image arrays. Layers of size 7. Narrowed in on 1.1 as being a decent step size.*

|  |  |
| --- | --- |
|  |  |

*Figure 4: Studied ROC curves of FFBP neural networks using 22 training, 10 validation, 10 testing for 10 maximum iterations, threshold of 0.5 between lesion and non-lesion image arrays. Layers of size 7. Narrowed in on 22 training, 10 validation, 10 testing as being baseline training-validation-testing breakdown .*

Monte Carlo simulations help visualize various outcomes in our FFBP framework. The FFBP neural network ran for 10 and 50 simulation, each time initializing random weights, biases, training, validation, and testing sets. Each simulation was timed and confusion matrices were calculated in order to compute accuracy, specificity, sensitivity, and number of false positives. Table 1 shows an average of all confusion matrix metrics over 10 and 50 simulations.

| *Model Description* | *Accuracy* | *Specificity* | *Sensitivity* | *Number of False Positives* | *Runtime (secs)* |
| --- | --- | --- | --- | --- | --- |
| *Sigmoid Activation Function* | 0.53  0.59 | 0.67  0.53 | 0.38  0.66 | 3.0  2.0 | 10 simulations, 73.09 s  50 simulations, 363.1 (6.05 min) |

*Table 1: Monte Carlo simulation results to evaluate performance of baseline perceptron implementing sigmoid activation functions. Recall, dataset sizes for training, validation, and test datasets are 22, 10 and 10 respectively. Specificity is also known as the True Negative Rate (TNR) whereas sensitivity is known as the True Positive Rate (TPR).*

* 1. **Variations in Activation Function**

The activation function maps various input signals to an output. The choice of activation function depends on the problem, the desired output and the expected model performance. For classification problems, activation functions should be continuous, differentiable and monotonically non-decreasing3. Various activation functions can be used in any layer of a neural network, though the general rule is to implement them in hidden layers. The sigmoid activation function is an “S” shaped curve, mapping inputs between the range of 0 and 1 (Appendix B). Sigmoid activation functions typically work better in the case of classifiers. In the application of anomaly detection in images, the sigmoid activation function was shown to have to the worst performance compared to other functions. A study classifying diseases in breast and thyroid images compared 8 activation functions and model accuracy 2 (Appendix B). Breast image scans contained normal and cancerous cells while thyroid scans contained normal, hyperthyroid, and hypothyroid cells. A multi-layered perceptron implementing FFBP was used to train the model. The study found using a hyperbolic tangent activation function resulted in a 97% accuracy in disease classification among images. The next best models included the sigmoid, sine, and exponential activation functions of 96.5%.

For this study, of the FFBP-based CAD for liver CT scans performance was evaluated using two activation functions: the sigmoid and hyperbolic tangent functions. Choice in activation function alters the change in error with respect to weights. For the sigmoid activation function change in weight is a function of step size, , and the input value. Delta, is depended on the change in activation function, *y*, with respect to the activity value. For the sigmoid activation function where *e* is the difference between the desired value and output (Appendix A). For the hyperbolic tangent activation function where *A* is the activity value. Thus, only two lines of code in our perceptron model had the be changed when alternating between activation functions: defining and All other elements of the baseline FFBP remained unchanged.

While the hyperbolic tangent activation produced more accurate results, the sigmoid activation function had greater specificity or true negative rate. Across simulations, the hyperbolic tangent activation function frequently produced zero true negative values. The true negative value in this application is the number of times the model correctly identified a non-lesion. Since the model whilst using a hyperbolic tangent was unable to classify non-lesions across simulations, the model is not specific. The hyperbolic tangent model produced over twice as many false positives then the sigmoid activation model. Of the sigmoid activation and hyperbolic tangent functions, the sigmoid activation is superior for FFBP-based CAD lesion detections.

| *Model Description* | *Accuracy* | *Specificity* | *Sensitivity* | *Number of False Positives* | *Runtime (secs)* |
| --- | --- | --- | --- | --- | --- |
| *Hyperbolic Tangent Activation Function* | 0.48  0.51 | 0.0  0.0 | 0.91  0.94 | 4.45  4.8 | 10 simulations, 29.6 s  50 simulations, 137 (2.28 min) |
| *Sigmoid Activation Function* | 0.53  0.59 | 0.67  0.53 | 0.38  0.66 | 3.0  2.0 | 10 simulations, 29.6 s  50 simulations, 139 (2.32 min) |

*Table 2: Monte Carlo simulation results to evaluate performance between baseline perceptron implementing sigmoid and hyperbolic tangent activation functions. Recall, dataset sizes for training, validation, and test datasets are 22, 10 and 10 respectively.*

* 1. **Variations in Data Preparation and Image Distortion**

As an alternative to the data preparation approach mentioned in (II)(a) to create our lesion image dataset, another dataset preparation method, mentioned below, was used for the set of experiments in this section:

1. Select 21 images with liver lesions and bounding boxes (lesion boundaries) smaller than 32x32 pixels.
2. Extract two 32x32 sub-regions from each image: one with a lesion and one without a lesion. This allows for fewer raw images to be processed, while not having much of an impact on the non-lesion data since the non-lesion regions are largely similar across the images in the dataset.
3. Of the 42 resulting images, allot 22 for the training dataset, 10 for validation datasets, and the remaining 10 for the testing datasets.
4. To increase the contrast in the images, process images according to instructions associated with the DeepLesion dataset1
   1. Convert the image to 32-bit format and subtract 32768 from the pixel intensities to obtain the original Hounsfield Unit (HU) values which generally vary between -1000 and 1000.
   2. Convert the intensities to the 0-255 range (“windowing”) to be able to view low contrast images. The “DICOM\_windows” field in the metadata associated with the *DeepLesion* dataset1 provides the default window for each image with min and max values represented as A and B respectively. The windowed intensity is then calculated as *I = min(255, max(0, (HU-A)\*255/(B-A)).*
   3. Save images to 8-bit image files.

We used an alternative data preparation approach for the following set of experiments since the image contrast needed to be adjusted before image distortion. The following image distortion procedures (adding noise or image rotation) were implemented prior to running the FFBP neural network with two hidden layers containing 7 nodes each. We chose to pre-process some of the images by adding noise or performing rotations based on the existing literature showing that such data augmentation techniques could potentially help improve the accuracy and robustness of image classifiers against overfitting.10-12 We ran the FFBP network on the following datasets distorted by noise or rotations:

1. 42 images: original or with Gaussian noise with a mean of 10 and a variance of 50
2. 42 images: original or with Gamma noise with *k=2*,=5 (mean=2\*5=10, variance =2\*52=50)
3. 42 images: original, with Gaussian noise with a mean of 10 and a variance of 50, or with Gamma noise with *k=2, =5*
4. 42 images: original or with Gaussian noise with a mean of 1 and a variance of 1
5. 42 images: original or with Gamma noise with *k=1, =1*
6. 42 images: original, with Gaussian noise with a mean of 1 and a variance of 1, or with Gamma noise with *k=1, =1*
7. 42 images: original or rotated by 45 degrees
8. 42 images: original or rotated by 90 degrees
9. 42 images: original or rotated by 180 degrees

Original Gaussian Noise (μ=10, σ2=50) Gamma Noise (k=2, θ=5)



*Figure 4: Various noise distortions of a 32 by 32 CT scan subregion with lesion*

We used a mix of non-distorted and distorted images (images with added noise or rotated image) in the training, validation, and testing sets, instead of the traditional approach of only training on distorted images, to help improve prediction accuracy. This strategy is similar to an approach known as test-time augmentation where distorted and “noisy” images are shown to the image classifier repeatedly during training and testing. 13, 14  The other parameters were kept the same as those in the baseline model. A number of classification metrics were used to quantify the performance of our neural network with respect to lesion image classification. These classification metrics are derived from a confusion matrix or a matrix comparing actual binary outcome values with the predicted outcome (Table 3). In our case, the outcome is the presence/absence of a lesion. Thus, true positives and true negatives are lesion/non-lesion images correctly identified as lesion/non-lesion images. False positives are non-lesion images identified as lesions and false negatives are lesion images identified as non-lesions as seen in Figure 1.

The results for 50 Monte Carlo simulations using image distortion are shown in Table 3 below and the results for 10 simulations are in Appendix C.

**50 simulations**

|  | *Accuracy* | *Sensitivity* | *Specificity* | *Number of False Positives* | *Runtime (secs)* |
| --- | --- | --- | --- | --- | --- |
| *Gn+Gm:*  *10 (50), 2 (5)* | 81.37% | 85.13% | 78.70% | 1.08 | 237.86 |
| *Gn:10 (50)* | 91.18% | 84.62% | 95.94% | 0.24 | 202.74 |
| *Gm: 2 (5)* | 81.37% | 58.43% | 94.22% | 0.39 | 221.57 |
| *Gn+Gm:*  *1 (1), 1 (1)* | 84.71% | 79.46% | 92.61% | 0.35 | 230.97 |
| *Gn: 1 (1)* | 82.35% | 62.81% | 96.56% | 0.20 | 229.03 |
| *Gm: 1 (1)* | 91.57% | 81.36% | 100% | 0.00 | 201.68 |
| *Rot (45˚)* | 62.75% | 89.57% | 15.36% | 2.98 | 234.01 |
| *Rot (90˚)* | 71.76% | 88.54% | 46.90% | 2.02 | 233.80 |
| *Rot (180˚)* | 76.08% | 69.72% | 86.00% | 0.73 | 230.97 |

*Table 3: Results for distorting input training set: Gn: Gaussian noise - mean (standard deviation); Gm: Gamma noise - shape parameter (scale parameter); Rot: Rotation (degrees by which whole image was rotated)*

As seen from results above, the performance metrics of the FFBP Neural Network model with transformed images (i.e. noise or rotations) are pretty similar across 10 and 50 simulations. The models performed particularly well on the Gaussian noise dataset (large mean) and the Gamma noise dataset (small mean). All the models with noisy datasets were reasonably accurate (> 75%) in predicting lesions/non-lesions. The model with some images rotated by 45˚ performed the worst, while the other models with rotated images performed on par with some of the noisy dataset models.

The results show that noise does not necessarily have a big impact on lesion detection, but rotations, particularly those at acute angles, can substantially affect the performance of our FFBP neural network. In the future, it would be interesting to train, validate, and test the FFBP neural network with a larger dataset and with other combinations of image transformations.

* 1. **Variations in Training Procedure**

The learning rate and iteration termination are two hyperparameters that can be tuned to improve the training process of the model. The learning rate affects the weight of the error that is applied in the backpropagation process. While increasing the learning rate can reduce the time it takes to train the model, having a large learning rate may lead to a sub-optimal model that does not converge towards the minimum in gradient descent. Similarly, determining when to stop training a model is important in determining whether the model is optimal or can still be improved with more training.

Our baseline model has a learning rate of 1.1 and terminates after 10 iterations (i.e. when the change in error between iterations was below 0.05). These baseline parameters were chosen based on literature reviews described in Section II A. In the hyperparameter training process, a lower learning rate was used and the process was forced to terminate only when a threshold was met.

Lowering the learning rate from 1.1 to 0.1 reduced the adjustment in the weights of the perceptrons during the backpropagation step. This slows the improvement of the model during each epoch, but improves the probability that the FFBP neural network will converge towards the optimal weights for the perceptrons given our training set.

Reducing the termination threshold of change in error between epochs from 0.05 to 0.0001 and removing the fixed number of iterations constraint, allows the model to continue learning until the improvement is negligible. Utilizing this new threshold ensures the model only terminates when it stops learning and the change in error between epochs is small.

These two changes in the training process significantly improved the sensitivity and the number of false positives while maintaining the accuracy and specificity when compared to the baseline model. Furthermore, the difference in the metrics between 10 Monte Carlo simulations and 50 Monte Carlo simulations was small, suggesting that this new training process would consistently generate a better and more optimal model.

While the model trained with the modified process increased sensitivity and reduced the number of false positives, the training process increased from 73s to 886s for 10 Monte Carlo simulations and from 363s to 4026s for 50 Monte Carlo simulations (Table 4). However, since a model is trained infrequently after a model is generated, the improvement in the model using these new parameters and process is a worthwhile tradeoff.

| *Model Description* | *Accuracy* | *Specificity* | *Sensitivity* | *Number of False Positives* | *Runtime (secs)* |
| --- | --- | --- | --- | --- | --- |
| *Learning Rate,* η*: 0.1*  *Delta Error Termination Threshold: 0.0001* | 0.56  0.59 | 0.53  0.65 | 0.6  0.51 | 2.6  1.84 | 10 simulations: 886s  50 simulations: 4026s |

*Table 4: Results for altering the training procedures*

1. **Results and Conclusion**

Best models were judged based on top results for accuracy, specificity, sensitivity, number of false positives, and runtime for 50 Monte Carlo simulations. An FFBP implementing a hyperbolic tangent activation function, sigmoid activation function, and a training dataset with gamme noise distortion and scale parameter 1 had the quickest runtimes of 137, 139, and 201. 68 respectively. It should be noted the variations in activation function experimentation was run on with a *Intel Xeon* W-2133 CPU at 3.60GHz with 32.0 GB of RAM. Other experimentations with variations in FFBP were run on slower processors. As expected, the slowest runtime was when the learning rate and delta error termination was reduced (Table 4).

The most accurate FFBP occured when the training dataset was distorted with Gamma (step size 1), Gaussian noise (step size 50), and a combination of the two noise types (both step size 1). Accuracy for these models ranged between 84 - 92%, far surpassing accuracy from the baseline, variations in activation function, and variations in training set models. This is due to the contrast adjustment procedure performed only during the data preparation for the image distortion experiments. When we ran the FFBP without contrast adjustment on the non-perturbed images, we got an accuracy of 43% compared to 82% with contrast adjustment, suggesting a need to appropriately adjust image contrast while testing image classifiers, particularly on noisy images.

The most specific models were FFBP using training sets distorted with Gamma (step size 1) and Gaussian noise (step size 1 and 50). The Gamma noise distortion model with step size 1 had a 100% specificity resulting in an obvious 0 number of false positives. When the training dataset was rotated 45 or 90 degrees, specificity dropped from 50 -60% to 15 - 47 %. A specificity value of 0.0 was found when using the hyperbolic tangent function and is due to the model’s inability to identify any true negatives (predicted non-lesion given there is no lesion).

For this application, the authors felt the sensitivity value was the most critical metric considering the implication a patient has or does not have a life-threatening lesion. The sensitivity metric is based on, if the patient actually has a lesion, how well does the model detect it? If the model has low sensitivity, there is a chance a patient has a lesion and our algorithm will frequently miss detection. The lowest values were 51%, 58.43%, and 62.81% for models with lower learning rate and delta error termination thresholds and distorting training images with Gamma (step size 5) and Gaussian (step size 1), respectively. Models with 89% sensitivity or better use the hyperbolic tangent function and rotating the training dataset 45 degrees prior to training, learning, and testing. While the hyperbolic tangent function was the most sensitive, 94%, it had the largest number or false positives.

Future work may include exploring combinations of these variations. It is clear distorting the training dataset with Gaussian and Gamma noise produces more accurate and specific results. Other activation functions such Gaussian or stochastic activation functions may be of interest. Using image distortion techniques in conjunction with the highly sensitive hyperbolic tangent model may produce a more robust model. As mentioned, FFBP-based CAD framework was used for only liver CT scans from *DeepLesion*. Other body CT scans such as bone and brain can be explore using our *DeepLesionClassifer* Python scripts on GitHub.

1. **Acknowledgements**

The following work was supported by Johns Hopkins Whiting School of Engineering, Engineering for Professionals program. The authors would like to thank Dr. Mark Fleischer for inspiring a passion for neural networks.

**References**

[1] Ke Yan, Xiaosong Wang, Le Lu, Ronald M. Summers, “DeepLesion: automated mining of large-scale lesion annotations and universal lesion detection with deep learning,” J. Med. Imag. 5(3), 036501 (2018),

doi: 10.1117/1.JMI.5.3.036501.

[2] S. Isa, Z. Saad, S. Omar, M. K. Osman, K. A. Ahmad, & H. A. Mat Sakim  (2010) “Suitable MLP Network Activation Functions for Breast Cancer and Thyroid Disease Detection.” IEEE Computer Society 978-0-7695-4262-1/10 $26.00 © 2010 IEEE DOI 10.1109/CIMSiM.2010.93

[3] S. Kumar, “Neural Networks A Classroom Approach”, McGraw Hill, International Edition, 2005.

[4] Saini, S.C., & Vijay, R. (2015). Mammogram Analysis Using Feed-Forward Back Propagation and Cascade-Forward Back Propagation Artificial Neural Network. *2015 Fifth International Conference on Communication Systems and Network Technologies*, 1177-1180.

[5] Ullah, Hadaate & Bhuiyan, Mohammad. (2018). Performance Evaluation of Feed Forward Neural Network for Image Classification. Journal of Science and Technology. 10. 10.30880/jst.2018.10.01.004.

[6] Summers, Ronald (2018) “*DeepLesion* Dataset” Created: Jun 27, 2018, 1:51 PM. Modified: Sep 5, 2018, 2:54 PM. Accessible via Web: <https://nihcc.app.box.com/v/DeepLesion>

[7] Lippman, Richard (1987), “An introduction to computing with neural nets.” IEEE ASSP Magazine.

[8] Prechelt, Lutz (1998), “Early Stopping - But When?” Orr, G.B. and Müller, K.-R.

[9] Brownlee, Jason (2018), “A Gentle Introduction to Early Stopping to Avoid Overtraining Deep Learning Neural Network Models.” <https://machinelearningmastery.com/early-stopping-to-avoid-overtraining-neural-network-models/>

[10] Minh, T. N., Sinn, M., Lam, H. T., & Wistuba, M. (2018). Automated Image Data Preprocessing with Deep Reinforcement Learning. *arXiv preprint arXiv:1806.05886*.

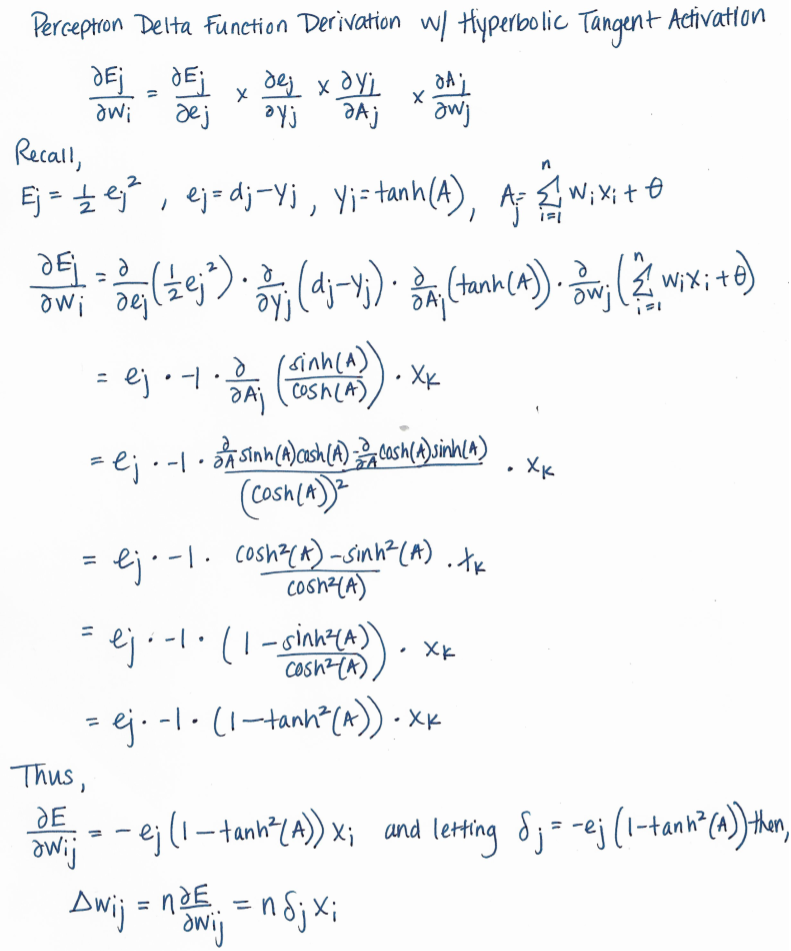
[11] Hussain, Z., Gimenez, F., Yi, D., & Rubin, D. (2018). Differential Data Augmentation Techniques for Medical Imaging Classification Tasks. *AMIA ... Annual Symposium proceedings. AMIA Symposium*, *2017*, 979–984.

[12] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

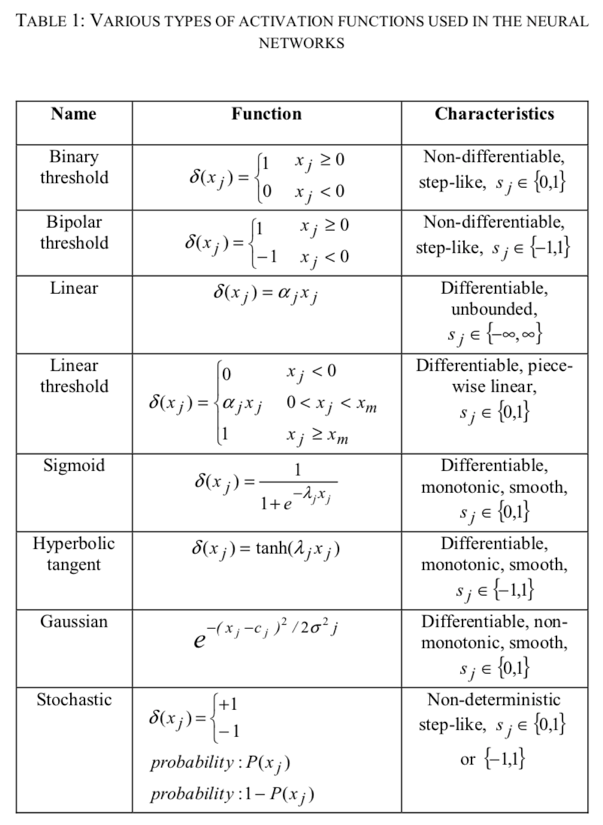
[13] Wang, G., Li, W., Aertsen, M., Deprest, J., Ourselin, S., & Vercauteren, T. (2019). Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks. *Neurocomputing*.

[14] Ayhan, M. S., & Berens, P. (2018). Test-time data augmentation for estimation of heteroscedastic aleatoric uncertainty in deep neural networks.

**Appendix A**

****

**Appendix B**



Adapted from Isa et. al. 2010

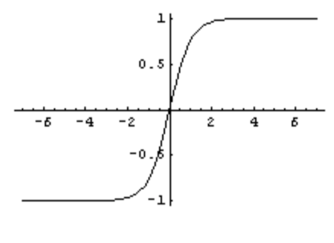
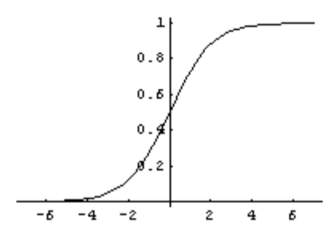
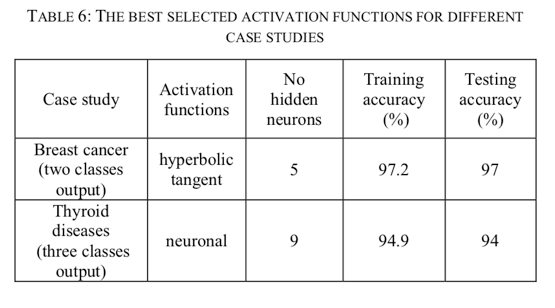


Figure 1: Sigmoid Activation function (left) vs. hyperbolic activation function (right)



Adapted from Isa et. al. 2010

**Appendix C (10 simulations)**

|  | *Accuracy* | *Sensitivity* | *Specificity* | *Number of False Positives* | *Runtime (s)* |
| --- | --- | --- | --- | --- | --- |
| *Gn+Gm:*  *10 (50), 2 (5)* | 77.27% | 80.49% | 75.06% | 1.09 | 50.48 |
| *Gn:*  *10 (50)* | 86.36% | 77.64% | 93.10% | 0.36 | 40.91 |
| *Gm: 2 (5)* | 76.36% | 38.18% | 97.69% | 0.18 | 51.27 |
| *Gn+Gm:*  *1 (1), 1 (1)* | 80.91% | 71.55% | 93.18% | 0.27 | 48.27 |
| *Gn: 1 (1)* | 82.73% | 73.33% | 88.99% | 0.54 | 48.32 |
| *Gm: 1 (1)* | 94.55% | 89.24% | 100% | 0.00 | 47.32 |
| *Rot (45˚)* | 62.73% | 88.17% | 18.18% | 2.82 | 53.00 |
| *Rot (90˚)* | 72.73% | 85.29% | 51.52% | 1.73 | 51.88 |
| *Rot (180˚)* | 76.36% | 68.96% | 90.91% | 0.64 | 49.29 |