(U) A Deep Learning Algorithm for the Detection of Image Quality Indicators

Walter Rose

US Army CCDC
Armaments Center

Building 908
Picatinny Arsenal, NJ 07806

Walter.s.rose5.civ@mail.mil

(U) Abstract: In this paper we describe a deep learning algorithm for the detection of image quality indicators. Given large data sets, being able to retrieve radiographs with certain desirable characteristics can prove very useful when auditing a production facility.

(U) Research Innovation and Objective(s): Transfer Learning is demonstrated in the context of Industrial Radiography. This is the first time machine learning has been used successfully in the context of industrial radiography, here at CCDC Armaments Center.

(U) Impacts on Warfighter Mission: Auditing production is a central theme of our mission at the Radiographic Laboratory. Being able to expedite the process with high accuracy ensures the best quality munitions are delivered to the warfighter.

(U) Keywords: Industrial Radiography, Supervised Learning, AlexNet, Transfer Learning, Rectified Linear Unity (ReLU), Machine Learning, Deep Learning, Convolutional Neural Network (CNN), MATLAB®, IQI, M1 Shell

1. (U) Introduction

(U) In this paper we demonstrate the use of advanced machine learning algorithms for the detection of image quality indicators, IQI. In recent years, there has been a lot of progress in using machine learning algorithms in the context of computer vision tasks. Machine Learning is defined as the use of algorithms to perform a specific task without using explicit instructions. [1] In contrast to this, there is what I call classical automatic defect recognition or classical ADR. In classical ADR, a set of explicit computer instructions, often including hard coded thresholds, is used to make decisions. Classical ADR has many flaws. Inherently, classical ADR posseses lack of versatility across systems. Machine learning algorithms have the potential to increase versatility. This paper will outline the detection of image quality indictators, IQI's, in a radiograph using a machine learning framework.

(U) There are two learning paradigms in machine learning. Supervised learning is a scheme where a human first separates the data into categories. This is called the training data set. The computer then "learns", developing optimal parameters based on the categorical data given. After the computer is done optimizing over variables, the training phase, the algorithm can be tested on data that is unlabelled. This unlabelled dataset is called a validation set. The other learning paradigm is called unsupervised learning and will not be explored further here. [2]

(U) There are many different types of machine learning algorithms. Fundamentally, a radiograph is a matrix of grey values. Imagine we have a dataset of images, where each image contains only two pixels. Further assume these pictures are either a dog or a cat. The objective is to solve the classification problem, wherby the computer will label each image as dog or cat.

(U) The computer takes these two pixels as inputs and feeds them into a machine learning algorithm. With enough data, a decision boundary in $\mathbb{R}^2$, the familiar euclidean space, can be derived. Note that the shape of the decision boundary can be highly nonlinear, as shown in figure 1 below.

(U) Advanced machine learning algorithms have been very successful at deriving these highly nonlinear decision boundaries. Fundamentally, this is why machine learning has gained so much popularity in recent years over the more familiar techniques of statistical regression. It is important to keep in mind that the ideas behind statistical regression, form the core of all machine learning algorithms. In fact, one can state that statistical regression is a special case of machine learning.

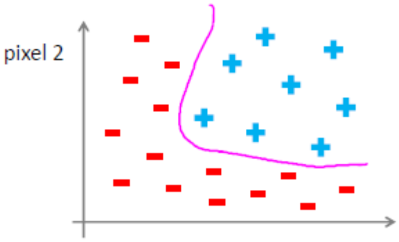

Figure 1. A plot of two pixels with a decision boundary shown in purple. The minus signs indicate this datapoint represented a cat, while the plus signs represented a dog. [9]

(U) For the more practical applications that follow, visualizing these decision boundaries is impossible. For most computer vision tasks, hundreds of pixels are used as inputs and thus the decision boundary exists in a high dimensional space.

(U) For computer vision tasks, successes have come from learning models called artificial neural networks (ANN). Artificial neural networks are mathematical models that are based off of biological neural networks. The basic unit of the ANN, is the neuron. Consider the network architecture depicted in figure 2 below.

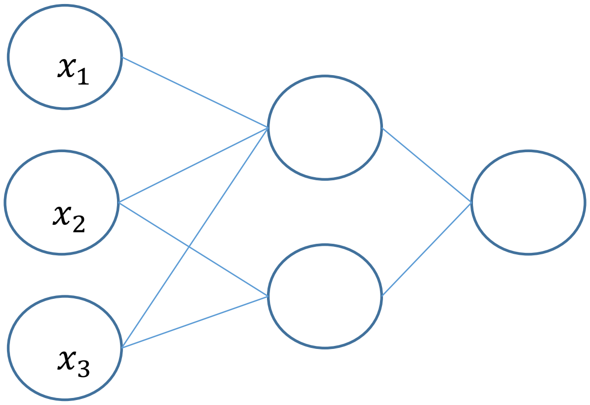

Figure 2. A simple schematic of a neural network with three layers: one input layer, one hidden layer, and the output layer.

(U) The neural network depicted in figure 1 shows an input layer with three input variables. For image processing tasks, the input layer will contain the greylevel values of each pixel. The next layer is what is called the "hidden" layer. This hidden layer, takes the inputs and does some mathematical operation on them. Lastly, the final layer computes a decision. For the classification problem, this is usually a simple vector containing zero's and a one. For example, with our earlier example of detecting dogs and cats, our output layer will be a two dimensional vector representing the two different categories. A vector of [0, 1] would represent a cat while a vector of [1, 0] would represent a dog. In column notation, this would be represented by,

$\begin{pmatrix} dog \\ cat \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. [9]

(U) We now turn to a more mathematical treatment of Neural Networks. Let us demonstrate the AND logic gate in the context of a simple neural network. Consider an input layer, with two variables, x_1 and x_2. These two variables can assume one of two values, 0 or 1. The neural architecture is shown in figure 3 below. Consider the following function, for the output, h(z). [9]

$$h_\theta(z) = g(z) = g(-30 + 20x_1 + 20x_2)$$

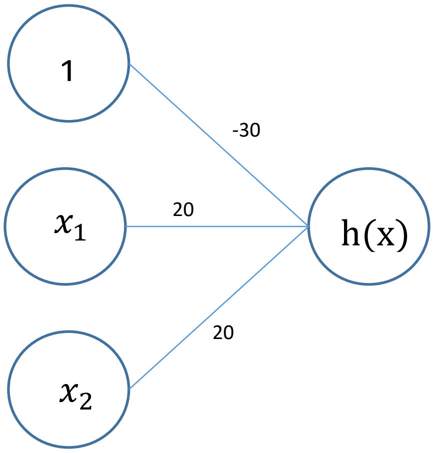

Figure 3. Neural Network for the AND logic [9]

(U) We now calculate all values of this neural network and tabulate in the table below. [9]

x_1	x_2	z	h(z)
0	0	-30	0
0	1	-10	0
1	0	-10	0
1	1	10	1

(U) We see that the above example has recovered the familiar AND logic gate. An important observation is the fact that this function seems to output zero for all negative values, but one for positive values. In the example above, the function g(z) was rather arbitrary. We now turn to the question of how to train a neural network to learn the function, g(z). [9]

(U) As we saw in the neural network for the AND logic gate, the output was based on a linear combination of the input variables. This is also true of a general neural network. In the general neural network depicted in figure (4), each line segement represents a different weight. Each subsequent layer of the network is computed based on the weights and the results of the previous layer.

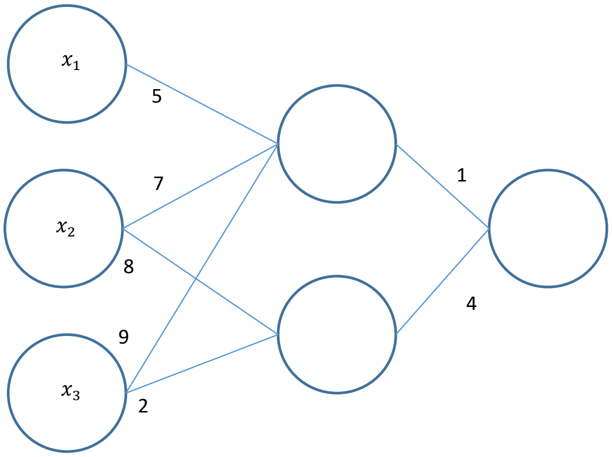

Figure 4. A neural network, with weights and bias.

(U) We update the weights and biases of the neurons on the basis of the error of the output. This is known in the literature as backpropogation. One can show that the non-linearity of the decision boundary stems from the presence of activation functions. It turns out that a neural network without activation functions are essentially, linear regression models. The activation function performs the non-linear transformation of the input, making it possible to learn more complex decision boundaries. An activation function simply determines if a neuron is on or off.

(U) Research has shown that rectified linear units (ReLU) have been very successful in the context of computer vision. [4] These are the activation functions that will be used in this paper. The mathematical expression for ReLU is,

$$f(x) = \max(0, x)$$

where x is the input to a neuron. A plot of the ReLU activation function is shown in figure 5.

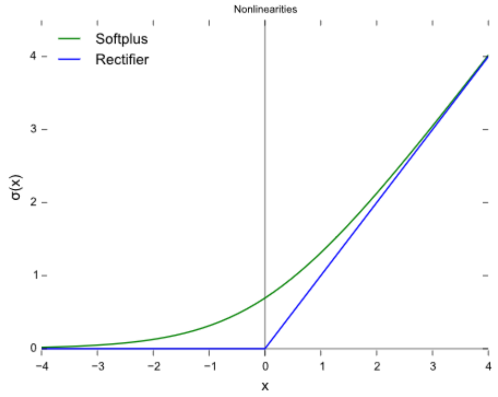

Figure 5. Graph of ReLU activation function in blue. [4]

(U) We now explain how a neural network learns over the training dataset. Similar to logistic regression, the objective is to find the minimum of a cost function. Recall from calculus, that to find the minimum of a function, one must find the derivative of the function, or the gradient and set it equal to zero. The actual formula of the cost function is omitted, as it is not instructive for our purposes.

(U) To train a neural network, one first randomly initializes the weights. One then implements forward propogation to get the output for an arbitrary input. Next, the value of the cost function is obtained. After this, one implements the backpropogation algorithm to compute the gradient of the cost function. Lastly, one uses gradient descent or other optimization algorithms to try to minimize the cost function.

(U) There has been a lot of success in using convolutional neural networks, or CNN, towards computer vision problems. [3] Images present a unique challenge when designing a neural network. When performing computer vision tasks the spatial correlation of the pixels becomes important.

(U) I will briefly explain the key components of a CNN and refer the reader to any textbook on machine learning for a more mathematical treatment. Spatial Pixel correlations are measured by convolving image patches with a filter. An image patch is simply a submatrix, 3x3 or 5x5, of pixels of the original image. In classical image processing, image filters can be used for all sorts of tasks such as edge detection, or sharpening, to name a few. In the context of machine learning, these filters are learned, based on the task at hand. With each iteration, the computer updates the filter to recognize features of interest. In our application, the filters will recognize what an IQI looks like in the spatial domain (matrix of pixel values).

(U) After the filter has been convolved with all image patches, a feature map is generated. An activation function is then applied to each feature map which decides if a given feature is present or not. You might also come across the idea of a "pooling layer" in the literature. A pooling layer extracts the largest value from a feature map in order to use these as inputs in subsequent layers.

(U) The problem discussed in this paper is an image classification problem. We have two categories. For a given radiograph, two choices are possible. Either an IQI is present, or an IQI is not present. It will be the computer's job to label each radiograph.

(U) In this paper we demonstrate transfer learning on M1 shells using AlexNet, a convolutional neural network designed by Alex Krizhevsky. Transfer learning is the process whereby a trained learning algorithm is applied to a different but related problem. Alex Krizhevsky's paper's primary result was that the depth of the model was essential for its high performance, which was computationally expensive, but made feasible due to the utilization of graphics processing units (GPUs) during training. [5] AlexNet is trained on more than a million images and can classify images into 1000 object categories. The network has an input image size of 227x227 pixels. More on the image input will be described below.

(U) It is my hope that this paper outlines a pipeline for future research and deployment of machine learning algorithms across CCDC armaments center.

2. (U) Method

(U) The first step was processing our radiographs to be of acceptable size and type for use in AlexNet. AlexNet was designed for type RGB, 227x227 pixels. The radiographs of our M1 Shells are greyscale images of size 1745x900.

(U) Our radiographs come from a production facility. Our task is far easier than most computer vision problems since the IQI's are placed in the same locations on every radiograph. Since AlexNet was designed for images with a very specific size, we first cropped each radiograph. We chose the region near the subcharge arbitrarily and could have chosen any of the IQI locations for testing. Consider figure 6 shown below. Our dataset consisted of 150,000 radiographs of M1 shells. I cropped the region near the subcharge, forming a new dataset that consisted of only the region directly below the supplementary charge. It was given the same serial number and was convereted to file type, tiff. The program for this is shown in the appendix, section 6.1. There was another preprocessing step before using these radiographs as inputs to AlexNet. Originally AlexNet was trained on RGB images. A simple conversion to RGB was necessary, but the images retained their true greyescale nature. This code is included in the appendix section 6.2 for the interested reader.

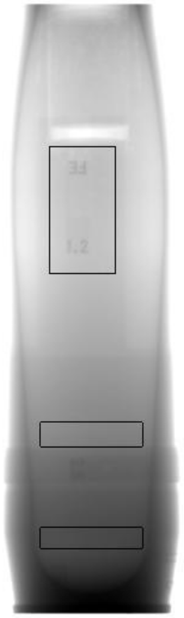

Figure 6. A Radiograph of an M1 Shell with 3 IQI's.

(U) AlexNet has a total of 25 layers. All 25 layers, with a description of each layer is included in the appendix, section 6.4. A few minor changes were made to AlexNet in order for it to perform our task. The output layer was reprogrammed to distinguish between two categories, IQI and *no* IQI. The Implementation of AlexNet is shown in the appendix in section 6.5.

(U) Images were sorted by CCDC Armament Center radiographers. All images that contained an IQI were put into one folder. The other images that did not contain an IQI were placed in a different folder. Traditionally, it is recommended that 80% of the data is used as "training data" and the remaining 20% of data is used as a validation set. There were only about 950 radiographs with an IQI. It is recommended that AlexNet has access to at least 1000 images of each category in order for the algorithm to perform well. Therefore, I conformed to the traditional guidlines of 80/20 split. More data would be needed to futher train and validate the algorithm.

3. (U) Results and Discussion

(U) AlexNet was extremely successful in determining which radiographs contained an IQI. We describe some of the performance statistics below.

Training took approximately one hour to achieve.

1918 iterations were performed. The statistics of these iterations are included in the appendix, section 6.5, for the reader.

An accuracy of 99.58% was achieved on the validation set. More testing and data would be required to further validate the algorithm.

A high degree of accuracy was achieved in a relatively short amount of time. In figure 7, a plot of accuracy versus iteration number is shown. (next page)

You can also see a graph of the loss function, representing the error in predicition as a function of iteration number in figure 8. (next page)

Using a combination of classical ADR and machine learning techniques, image audits can be done quickly and more easily. This allows for radiographers and project managers to become aware of critical problems sooner.

(U) When performing an audit, there are a few critical metrics that can now be gathered relatively quickly using machine learning and classical ADR. This helps radiographers at CCDC Armaments Center focus on critical issues sooner.

(U) Image Quality is the first line of defense. If image quality fails, the audit stops, and an investigation must be conducted to understand why image quality has not been met. Military specifications often dictate a placement frequency of IQI's on a radiograph. The algorithm in this paper can check IQI frequency quickly with high accuracy. After IQI placement frequency has been verified, a measurement of the Signal to Noise ratio (SNR), can be made. We refer the reader to our previous work to see how this can be automated. [6]

(U) A combination of machine learning and classical ADR can then check to see if the remainder of IQI requirements have been satisfied.

(U) Finally, a classical ADR algorithm can be run to check for gross anomalies in the dataset. We refer the reader to the following paper for a description and performance of such an algoirthm. [7]

4. (U) Conclusion

(U) Machine learning has been very successful in recent years. Research from Google AI has shown that machine learning algorithms can detect breast cancer more reliably than doctors in many scenarios. [8]

(U) As machine learning becomes more prevalent, there will be many benefits for use at CCDC Armaments Center. machine learning via convolutional neural networks can and will aid radiographers across the industrial base ensuring the highest quality munitions are delivered to the warfighter.

(U) It is my hope that this paper outlines a pipeline for future research and deployment of machine learning algorithms across CCDC armaments center.

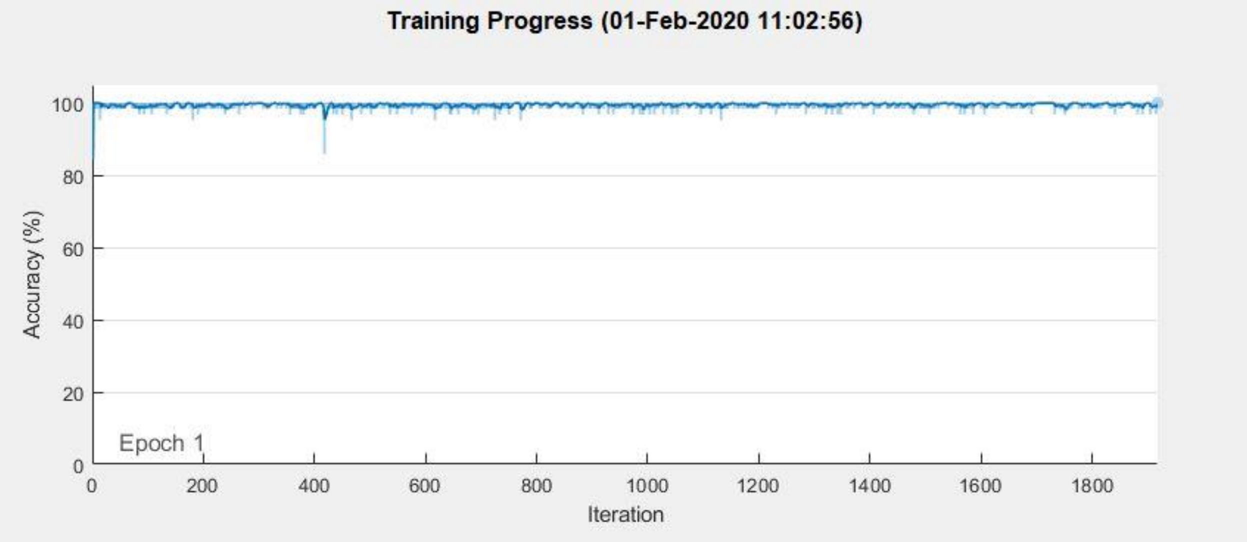

Figure 7. Training progress of AlexNet to detect IQI's in a radiograph. Accuracy is plotted against iteration. The darker line represents the smoothed version. Convergence was very fast.

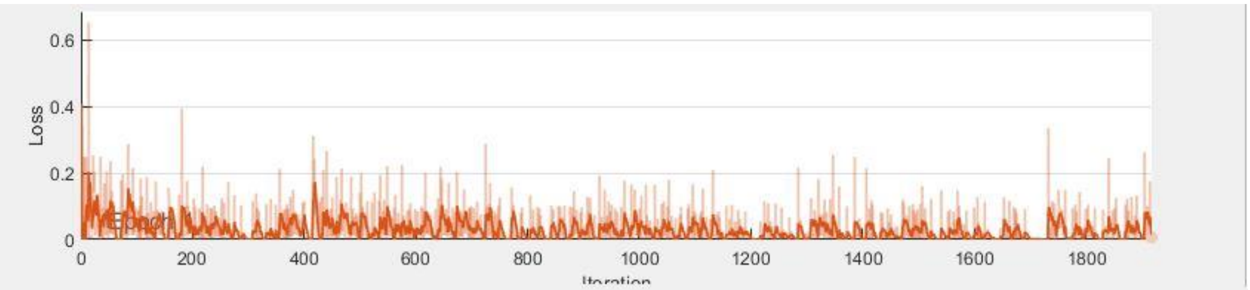

Figure 8. Training progress of AlexNet to detect IQI's in a radiograph. The darker line represents the smoothed verison.

6. (U) Appendix

(U) 6.1 Cropping Procedure for M1 Shells

```
cd 'D:\MCAAP M1 Recap\'

I = dir('D:\MCAAP M1 Recap\*.dcm');

nfiles = length(I);

for i = 1:nfiles

        filename = strcat('D:\MCAAP M1 Recap\', I(i).name);

        X = dicomread(I(i).name); %Read image and store as Matrix X

        %1.5 IQI
        %Submatrix Window
        ymin = 480; %Starting row
        ymax = 706; %Ending row
        xmin = 230; %Starting Column
        xmax = 456; %Ending Column
        W = X(ymin:ymax,xmin:xmax);

        name = strcat('C:\Users\walter.s.rose5\Documents\Images\M1
TrainingData\', I(i).name);
        imwrite(W, [name,'.tif']);

end
```

6.2 Converson of Greyscale to RGB

```
cd 'C:\Users\walter.s.rose5\Documents\Images\M1 TrainingData\'

I = dir('C:\Users\walter.s.rose5\Documents\Images\M1 TrainingData\*.tif');

nfiles = length(I);

for i = 1:nfiles

        filename = strcat('C:\Users\walter.s.rose5\Documents\Images\M1
TrainingData\', I(i).name);

        X = imread(I(i).name); %Read image and store as Matrix X

        Y = repmat(X,1,1,3);

        name = strcat('C:\Users\walter.s.rose5\Documents\Images\M1
TrainingData\Test\', I(i).name);
        imwrite(Y, [name,'.tif']);
end
```

6.3 AlexNet Run Program

```
cd 'C:\Users\walter.s.rose5\Documents\Images\'

allImages = imageDatastore('M1 TrainingData', 'IncludeSubfolders', true,
'LabelSource', 'foldernames');

[trainingImages, testImages] = splitEachLabel(allImages, 0.8, 'randomize');

%load a pre-trained deep convolutional neural network
Net = alexnet;
layers = Net.Layers;
layers(23) = fullyConnectedLayer(2);
layers(25) = classificationLayer

options = trainingOptions('sgdm', 'InitialLearnRate', 0.01, 'MaxEpochs', 1,
'MiniBatchSize', 64, 'Plots', 'training-progress');
myNet = trainNetwork(trainingImages, layers, options);
```

```
predictedLabels = classify(myNet, testImages);
accuracy = mean(predictedLabels == testImages.Labels);
```

6.4 Internal Structure of AlexNet for IQI Detection

```
25x1 Layer array with layers:

   1   'data'    Image Input                    227x227x3 images with 'zerocenter' normalization
   2   'conv1'   Convolution                    96 11x11x3 convolutions with stride [4  4] and padding [0  0  0  0]
   3   'relu1'   ReLU                           ReLU
   4   'norm1'   Cross Channel Normalization    cross channel normalization with 5 channels per element
   5   'pool1'   Max Pooling                    3x3 max pooling with stride [2  2] and padding [0  0  0  0]
   6   'conv2'   Convolution                    256 5x5x48 convolutions with stride [1  1] and padding [2  2  2  2]
   7   'relu2'   ReLU                           ReLU
   8   'norm2'   Cross Channel Normalization    cross channel normalization with 5 channels per element
   9   'pool2'   Max Pooling                    3x3 max pooling with stride [2  2] and padding [0  0  0  0]
  10   'conv3'   Convolution                    384 3x3x256 convolutions with stride [1  1] and padding [1  1  1  1]
  11   'relu3'   ReLU                           ReLU
  12   'conv4'   Convolution                    384 3x3x192 convolutions with stride [1  1] and padding [1  1  1  1]
  13   'relu4'   ReLU                           ReLU
  14   'conv5'   Convolution                    256 3x3x192 convolutions with stride [1  1] and padding [1  1  1  1]
  15   'relu5'   ReLU                           ReLU
  16   'pool5'   Max Pooling                    3x3 max pooling with stride [2  2] and padding [0  0  0  0]
  17   'fc6'     Fully Connected                4096 fully connected layer
  18   'relu6'   ReLU                           ReLU
  19   'drop6'   Dropout                        50% dropout
  20   'fc7'     Fully Connected                4096 fully connected layer
  21   'relu7'   ReLU                           ReLU
  22   'drop7'   Dropout                        50% dropout
  23   "         Fully Connected                2 fully connected layer
  24   'prob'    Softmax                        softmax
  25   "         Classification Output          crossentropyex
```

6.5 RunTime Statistics for AlexNet M1 IQI Detection

```
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |

|         |             |   (hh:mm:ss)   |   Accuracy   |     Loss     |      Rate       |
||=============================================================================
=================|

|       1 |           1 |       00:00:05 |       84.38% |       0.4085 |          0.0100 |

|       1 |          50 |       00:01:44 |       98.44% |       0.0944 |          0.0100 |

|       1 |         100 |       00:03:23 |      100.00% |       0.0108 |          0.0100 |

|       1 |         150 |       00:05:02 |      100.00% |       0.0035 |          0.0100 |

|       1 |         200 |       00:06:41 |       98.44% |       0.0441 |          0.0100 |

|       1 |         250 |       00:08:20 |      100.00% |       0.0070 |          0.0100 |

|       1 |         300 |       00:10:01 |      100.00% |       0.0013 |          0.0100 |

|       1 |         350 |       00:11:40 |      100.00% |       0.0011 |          0.0100 |

|       1 |         400 |       00:13:20 |       98.44% |       0.1150 |          0.0100 |

|       1 |         450 |       00:15:01 |       96.88% |       0.1273 |          0.0100 |

|       1 |         500 |       00:16:41 |       98.44% |       0.1412 |          0.0100 |

|       1 |         550 |       00:18:21 |      100.00% |       0.0092 |          0.0100 |

|       1 |         600 |       00:20:01 |      100.00% |       0.0030 |          0.0100 |

|       1 |         650 |       00:21:42 |      100.00% |       0.0163 |          0.0100 |

|       1 |         700 |       00:23:22 |      100.00% |       0.0044 |          0.0100 |

|       1 |         750 |       00:25:02 |       96.88% |       0.1277 |          0.0100 |

|       1 |         800 |       00:26:42 |      100.00% |       0.0013 |          0.0100 |

|       1 |         850 |       00:28:22 |      100.00% |       0.0037 |          0.0100 |

|       1 |         900 |       00:30:02 |      100.00% |       0.0086 |          0.0100 |

|       1 |         950 |       00:31:42 |       98.44% |       0.1051 |          0.0100 |

|       1 |        1000 |       00:33:22 |       98.44% |       0.0699 |          0.0100 |

|       1 |        1050 |       00:35:02 |      100.00% |       0.0023 |          0.0100 |

|       1 |        1100 |       00:36:41 |      100.00% |       0.0057 |          0.0100 |

|       1 |        1150 |       00:38:21 |      100.00% |       0.0017 |          0.0100 |

|       1 |        1200 |       00:40:01 |      100.00% |       0.0023 |          0.0100 |

|       1 |        1250 |       00:41:40 |      100.00% |       0.0008 |          0.0100 |

|       1 |        1300 |       00:43:20 |      100.00% |       0.0015 |          0.0100 |

|       1 |        1350 |       00:45:01 |      100.00% |       0.0043 |          0.0100 |

|       1 |        1400 |       00:46:41 |      100.00% |       0.0015 |          0.0100 |

|       1 |        1450 |       00:48:21 |      100.00% |       0.0036 |          0.0100 |

|       1 |        1500 |       00:50:01 |      100.00% |       0.0051 |          0.0100 |

|       1 |        1550 |       00:51:41 |      100.00% |       0.0011 |          0.0100 |
|       1 |        1600 |       00:53:21 |       98.44% |       0.0964 |          0.0100 |
|       1 |        1650 |       00:55:01 |      100.00% |       0.0023 |          0.0100 |
|       1 |        1700 |       00:56:41 |      100.00% |       0.0020 |          0.0100 |
|       1 |        1750 |       00:58:21 |      100.00% |       0.0023 |          0.0100 |
|       1 |        1800 |       01:00:01 |       98.44% |       0.0928 |          0.0100 |
|       1 |        1850 |       01:01:41 |      100.00% |       0.0159 |          0.0100 |
|       1 |        1900 |       01:03:21 |      100.00% |       0.0009 |          0.0100 |
|       1 |        1918 |       01:03:57 |      100.00% |       0.0066 |          0.0100 |
|========================================================================
================|
```

References

[1] Bishop, C. M. (2006), Pattern Recognition and Machine Learning, Springer

[2] Hinton, Geoffrey; Sejnowski, Terrence (1999). Unsupervised Learning: Foundations of Neural Computation. MIT Press.

[3] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, Backpropagation Applied to Handwritten Zip Code Recognition; AT&T Bell Laboratories

[4] Xavier Glorot, Antoine Bordes and Yoshua Bengio (2011). Deep sparse rectifier neural networks (PDF). AISTATS.

[5] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." Advances in neural information processing systems. 2012.

[6] Rose et al. Automated SNR Search to Support an Image Audit. DTIC Technical Reports, February 2020

[7] Rose et al. Automated Analysis of Digital Radiographs to Support an Image Audit. DTIC Technical Reports, November 2019

[8] McKinney et al. International evaluation of an AI system for breast cancer screening, Nature, 577, 89-94 (2020)

[9] Ng Andrew, Notes from Machine Learning, Stanford Online, 2019