Using two feed-forward layers for fault detection of hole

Yvonne Dalle Nyame¹, Peijiang Yuan and Abamou Kaoubara
BUAA: Beijing University of Aeronautics and Astronautics

¹ E-mail: yvonnedalle@buaa.edu.cn

Abstract. Automated assembly technology is a major component of modern aerospace manufacturing. Hole-Making Robot (HMRs) can significantly improve the efficiency of the aircraft manufacturing and can also ensure reliability of processing. The machining accuracy of the HMR depends on the precision of the mechanism, the accuracy of hole-positioning, and the vertical degree of the hole. Evaluating the precision of the hole system requires a complete set of inspection systems to measure the relevant hole parameters. Visual detection technology can provide more information on the object, which in theory is better suited to simulate real world applications. Due to the rapid development of current visual technology and the actual demand of aviation holes, this paper attempts to apply Convolution Neural Network (CNN) technology to the concrete task of visual inspection and improve the performance of visual inspection systems. Hole classification and hole flaw detection are realized via two feed-forward layers. We created a dataset of 1300 images of holes for testing. These were collected from various sources. The highest accuracy was 98.15% and was achieved using the two feed-forward neural network algorithm.

1. Introduction
Using basic visual inspection system to study the problem of classification and fault diagnosis, then by synthesizing a variety of image processing algorithms, a better processing effect was achieved. By employing the renowned CNN training and parameters adjustment using basic algorithms of machine learning, we attempted to optimize the existing image processing. This proved to be a valuable tool for smart robot especially as it utilizes a huge number of unsupervised data. Subsequently, it offers improvement in robot performances even with a substantial reduction in the need for inputs from human experts in retrieving features from every new data. This paper presents a cost effective approach to classify drilled holes into defectives and non-defectives. A novel description of an optimized smart drilling holes robot with visual capacity based on decision-making that fundamentally idealize the stipulated computational demand was achieved. A set of homogeneous and heterogeneous pool of features was investigated under this framework. A thorough test was performed and compared the results with state-of-the-art classifiers. The experimental results clearly highlight the improvements in the different features learned within this framework.

This paper also proposes a dataset of 1300 images that will be published online for further research. To classify holes, a proposed solution was implemented using python with Tensorflow running on a Linux desktop. The aim is to understand features representative of images that are appropriate for classification of defective holes. Our work is as follows: In part 2, holistic view and evolution of this study is presented. Part 3 is an outline of the proposed architecture outlining its major processing steps and how they are formed to develop an all-encompassing framework of deep learning. Part 4 includes results obtained by applying this framework on our dataset for defective hole classification. Part 5
discusses possible implementations that allow this framework to be adjusted and be suitable for very large datasets and concludes this report and suggests additional features that need to be performed for the HMR.

1.1. Related work
It is imperative to discuss the fundamentals of deep learning—its present state and future projections. This will involve outlining of the neurons functions and proffer a broad presentation of its variants and applications. The first analogy is with the brain; it has different sections that perform various tasks [1]. Firstly it learns from being exposed to data as it is not pre-programmed. To resolve numerical solutions to various robotic issues is a critical aspect in robotics, this makes deep learning an essential tool as it is highly utilized in this aspect. Therefore the parameters of neuron from data need to be studied—Descent gradient is a general method of achieving this study. Descent gradient method essentially quantifies the errors (popularly known as the loss function) that our neuron performs by training the dataset of images which is the exact thing used for logistic and linear regression to obtain a closed-form solution. Iteratively the parameters so called weights are updated during the training. The learning rate should be chosen properly. The cost function is obtained from the likelihood of learning which is then computed in batches to improve training time. The human brain function has been a great source of interest for Scientists for ages; the pioneering application based on artificial neural network was made by McCulloch and Pitts in 1940 [2]. The paper described the concept of the neuron buttressing its connections for receiving inputs, processing data, and generating outputs. Then Minsky et al. detailed the limits of a simple perceptron [3].

Two types of HMR exist: manual and automatic. The automatic is known for its flexibility and relative cost efficiency that estimates and control an automatic HMR whereas the Manual HMR are inflexible and required much space. More so, for automatic HMR, the Dedicated drilling end-effector have been significantly utilized in manufacturing [4-8] as can be seen in the very first generation by Electroimpact for a HMR of the wing trailing edge flaps of the Boeing’s F/A-18E/F Super Hornet fighter aircraft [5] using the one-sided cell end effector (ONCE) system for drilling, countersinking, and inspection.

ONCE system utilized a one-up assembly of aerospace structures where drilling, countersinking, inspection, and fastening were completed in a single step, thus the disassembly of parts in the traditional way of deburring, cleaning, and sealing operations were eliminated [6]. Many researchers [7-9] showed that robotic drilling has the potential to drill precision holes on aerospace structures. Typical aerospace structures (e.g., large aircraft panels) have thousands of fastener holes; hence, automatic generation of program for robotic drilling is necessary. However, Computer graphics-based off-line programming systems, which support simulation, collision detection, and user interaction, are widely used to reduce idle time of the robots and improve the production efficiency [10-14]. On the same hand, Arao M. and S [13] have demonstrated that the use of hybrid combinations of Neural and Fuzzy as RNFN (Recurrent Neuro Fuzzy Network) and RFNN (Recurrent Fuzzy Neuro Network) in automatic feed drilling system has a significance of guidance to improve the performance of a HMR. Walter et al. [15] performed a security alert before the hole being drilled on rock strata.

In modern times, there’s a high preference for appearance based methods for object detection. Lee et al [16] obtained a high classification accuracy in bearing fault detection using CNN on a signal dataset. Zheng et al. [17] proposed a CNN application in which a multivariate time-series dataset in feature extraction was conducted followed by a classification task with the extracted features.

However, our goal is to optimize the off-line programming by applying CNN due to their good classification and detection properties based on similarity, high adaptability and fault tolerance [13]. In this case the CNNs can easily extract feature and diagnose a defective hole of a HMR.
2. Proposed architecture

Our own dataset is created and adjusted the neural network to meet our needs. Also, different tools and layer have been used to achieve our results such as LMDB which is a light and efficient key-value databases. Versus hdfs5 (another type of data layer), it uses memory-mapped files and does not need to load the entire dataset into the memory. Thus it is suitable for large datasets.

The cost function—also known as the cross-entropy cost function—for so-called regularization methods (L1 and L2 dropout and artificial expansion of the training data) which proffers a more desirable method for initializing the weights in the network. Therefore the weights presents a set of heuristics to aid in determining better hyper-parameters for the network [17].

The convolutional layer decides the trade-off between speed and accuracy of the learning rate. The learning rate is one of the parameters of gradient descent. It sets the biased learning rate of the filters and offers better convergence rates.

The pooling layer is a procedure that takes input over a certain area and reduces it to a single value according to the type of applied operation. Examples include max-pooling, average-pooling, etc. The fully connected layer is an initialization layer that impacts non-convex optimization algorithms such as stochastic gradient descent, the Xavier algorithm, etc. Gaussian or uniform distributions usually have fairly arbitrarily set variances.

Loss layer & RELU: In LeNet, Relu is used as an activation function, and Softmax regression functions as loss of function.

Our procedure is as follows:

1. Pre-processing step: In this step, it firstly involves the creation of our own dataset converted into a tensorflow record format. With the use of a drill holes were created which forms our images albeit some have flaws or are defective hence need to be identified. then, analyzed our data by modifying the way the images are displayed. This step includes executing brightness and contrast normalization as well as whitening. All features are remapped to provide improved results.

2. Processing step: To carry this out, patches are pulling out from the original dataset and from inputs for unsupervised learning procedure. Then, extracting 10,000 7x7 random patches from 1,300 32x32 images from the training data set. Each image contains a hole. Unsupervised learning via K-Means is carried out here, which attempts to learn a dictionary of image attributes to elaborate a new representation of the original dataset. This procedure is as a result of the insights from [18] concerning K-Means implementation as well as the necessary number of clusters and processing steps.

3. Training step: Images use a dictionary with data pre-processing by a discriminative method as described earlier. The features are encoded and include the height, width, colour space, channels, format, label, text, and filename. Next, Mapping the original features (pixels) of the dataset into a new feature space is a function of the learned dictionary. Subsequently, execution of a new feature representation of the original dataset with the hope of obtaining a linear solution. The soft threshold encoding scheme for K-Means and an implementation in Python were performed, which runs much better with limited memory resources. In this step, each original 32 x 32 pixel image represents a 1 x 1024 vector and is now encoded in a 1 x 6400 vector. Lastly, the classification executed using standard linear classification algorithms. Then the SVM implementation provided by [19] was used for testing.
3. Materials and method
Tensorflow is a powerful tool for deep learning. Our dataset is converted into the TFrecord format of TensorFlow. Our images are collected from a drilling robot and from the Internet, shown in figure 1 and figure 2. The holes have different positions and are classified into two classes (simple hole and countersink hole). These are sub-classified into two classes (good and fault). The images were trained with a resolution of 32x32. Increasing the resolution to 128x128 results in longer training time but no better results.

3.1. Preprocessing phase
Our dataset directory contains 04 subfolders (Train/class 1, train/class 2, validate/class 1, validate/class 2).
A quarter (1/4) is placed in each of the subfolders (Validate/class 1, validate/class 2).
A file called Libelles.txt is created and write the names of our two classes in the root directory.
During the pre-processing step, our images are trained with the cross-entropy loss function.

3.2. Training Phase
At the beginning of the training, the error is defined and corrected for prediction and accuracy: Tf.train.shuffle_batch is used to get a randomly selected batch of 100 images from the dataset.
Initialize the function of conversion png to jpeg for consistency. This function is then decoded from jpeg to RGB.
The hierarchical SoftMax module speeds up training over an extremely large number of classes.
Cross-map pooling (sometimes known as MaxOut) is used for certain types of visual and text models.

3.3. Algorithm
Step 1: initialize weights and biases. Nfeatures1=32 features for each patch of size x5, same weights are used for all patches.
Step 2: reshape raw image data to 4D tensor.
Step 3: hidden layer one used pooling which reduces each dim by factor of 2; pool (convolution(wx +b)
Step 4: hidden layer two with nfeatures2 features per 5x5 patch
Step 5: densely connected layer operating on entire image (rather than patch). A large number of neurons (1024) in each dimension (32) is used and make sure dimensions are multiple of 4
Step 6: flatten output from previous layer
Step 7: reduce overfitting by applying dropout. For each neuron is kept with a probability keep_probab

Figure 1. Defective hole vs Non-defective hole.
Figure 2. The Hole-Making Robot used during experiments.
Step 8: create a readout layer which outputs to nclass categories

Step 9: define output calc(for each class) with y=softmax(wx+b). softmax gives probability distribution across all classes.

\(\varepsilon = 10^{-5}, \eta = 0.5, \beta = 0.9\)

\(E = \varepsilon + 1, M, N\)

initialisation

\(m = 0, n = 1\)

begin

do
input_training_sample_set

calculate_output_each_layer

calculate_network_output_error

\[E = \frac{1}{2} \sum_{n=1}^{N} (y^n - O^n)^2\]

until \((n < M \cdot n + 1)\)

Do(calculate_error_signal_of_hidden_layer_and_hidden_layer_weights

adjust_the_output_layer_and_hidden_layer_weights

increment \(_m(m + 1)\)

until \((E \leq \varepsilon, M \leq m)\)

\[y = \text{tf.nn.softmax}(\text{tf.matmul}(h_{fc1 \_ drop}, w_{fc2}) + b_{fc2})\]  

(1)

Cross_entropy = reduce_mean(reduce_sum(y \* \log(y), r=[1]))

(2)

train_step = AdamOptimizer(\text{le-4}).minimize(cross_entropy)

(3)

\[\text{Predict} = \text{equal}((\text{arg max}(y, 1)), \text{argmax}(y, 1))\]

(4)

Accuracy = reduce_mean(cast(Pr edict, \text{float32}))

(5)

\[w_{convolution1} = \text{weight \_ variable}([5, 5, 1, nfeatures1])\]

(6)

\[b_{convolution1} = \text{bias \_ variable}([nfeatures1])\]

(7)

\[x\_image = \text{tf. reshape}(x, [-1, width, height, 1])\]

(8)

\[h\_convolution1 = \text{tf.nn.relu}(\text{conv2d}(x\_image, w\_convolution1) + b\_convolution1)\]

(9)

\[w\_convolution2 = \text{weight \_ variable}([5, 5, nfeatures1, nfeatures2])\]

(10)

\[b\_convolution2 = \text{bias \_ variable}([nfeatures2])\]

(11)

\[h\_convolution2 = \text{tf.nn.relu}(\text{conv2d}(h\_pool1, w\_convolution2) + b\_convolution2)\]

(12)

\[h\_pool2 = \text{max} \_ pool \_ 2x2(h\_convolution2)\]

(13)

\[h\_fc1 = \text{tf.nn.relu}(\text{tf.matmul}(h\_pool2 \_ flat, w\_fc1) + b\_fc1)\]

(14)

\[w\_fc2 = \text{weight \_ variable}([nNeuronsfc, nfeatures])\]

(15)

\[b\_fc2 = \text{bias \_ variable}([nclass])\]

(16)

keep_probab = tf.placeholder(tf.float32)

(17)

\[h\_fc1\_dropout = \text{tf.nn.dropout}(h\_fc1, \text{keep \_ probab})\]

(18)

\[w\_fc2 = \text{weight \_ variable}([nNeuronsfc, nclass])\]

(19)

\[b\_fc2 = \text{bias \_ variable}([nclass])\]

(20)

4. Results and discussion

The database of holes images used to test this framework is composed of 1,300 labeled training images and 10,000 labeled test images. Utilizing the available training data and extracting 10,000 7x7
patches which in turn was used to test the K-Means. The encoding and classification steps used 1/3 of both training and test sets that were obtained randomly from their respective complete sets. Afterwards by executing an SVM classifier using a standard linear kernel, the results are obtained in table 1.

One possible approach for implementing this is a scale-out processing framework such as kibana [20] which have a great community.

1. Pre-processing step: executing this step is straightforward. Kibana supports parallelization which process jobs into several chunks.
2. Processing step: Extracting random patches and feature encoding run smoothly. In the map step, centroids are assigned to the data set. In the reduce step, new centroids are computed.
3. Training step: A linear classifier—logistic regression through gradient descent is used for its predictive performance. This uses numerous partial gradients would be computed in parallel.

| Hidden layer unit | Network error (x10^-6) | Training times (ms) |
|-------------------|------------------------|---------------------|
| 2                 | 1.91                   | 120                 |
| 3                 | 1.91                   | 140                 |
| 4                 | 1.91                   | 605                 |
| 5                 | 2.17                   | 240                 |
| 6                 | 1.96                   | 180                 |
| 7                 | 1.92                   | 132                 |
| 8                 | 1.98                   | 171                 |
| 9                 | 1.95                   | 420                 |
| 10                | 1.93                   | 110                 |
| 11                | 1.95                   | 256                 |
| 12                | 1.94                   | 186                 |
| 13                | 1.96                   | 181                 |
| 14                | 1.95                   | 184                 |

5. Conclusions
Though having similar accuracy, smaller CNN architectures gives at least 3 major advantages [21]:

Smaller CNNs utilizes less bandwidth to export a new model from the cloud.
Smaller CNNs utilizes less communication across servers during distributed training.
Smaller CNNs are the most feasible to deploy on FP-GAs and other hardware with limited memory.

We have shown that a visual inspection system using CNN can significantly improve the performances of a HMR. This paper have presented the simplest model between training performance and Hidden layer unit. Our next paper will present a theoretical analysis for optimizing the target detection of such a HMR using a CNN.

Acknowledgment
This research is partially supported by the National natural Science Foundation of China (No.61375085).

References
[1] Lashley K S 1929 Brain mechanisms and intelligence Chicago: Chicago University Press
[2] McCulloch, W.S. and Pitts 1940 A Logical Calculus of the Ideas Imminent in Nervous Activity The Bulletin of Mathematical Biophysics pp 115-133

[3] Minsky M. L. and Papert S. A. 1969 Perceptrons Cambridge, MA: MIT Press

[4] Bi S and Liang J 2011 Robotic drilling system for titanium structures The International Journal Advanced Manufacturing Technology 54 pp 767–774

[5] DeVlieg R, Sitton K, Feikert E and Inman J 2002 ONCE (one sided cell end effector) robotic drilling system In: SAE Automated Fastening Conference & Exposition, Chester, ENGLA pp 01-2626

[6] Devlieg R and Feikert E 2008 One-up assembly with robots In: SAE 2008 Aerospace Manufacturing and Automated Fastening Conference & Exhibition, North Charleston, SC, USA. SAE Technical Papers 01 pp 2297

[7] Atkinson J, Hartmann J, Jones S and Gleeson P 2007 Robotic drilling system for 737 Aileron In: SAE 2007 AeroTech Congress & Exhibition, Los Angeles, CA, USA 01 pp 3821

[8] Liang J and Bi S 2010 Design and experimental study of an end effector for robotic drilling The International Journal Advanced Manufacturing Technology 50 pp 399–407

[9] Webb P, Eastwood S, Jayaweera N and Chen Y 2005 Automated aero structure assembly Industrial Robot 532 pp 383–387

[10] Craig J 2004 Introduction to robotics: mechanics and control 3rd ed Pearson Prentice Hall, Upper Saddle River, NJ

[11] Mitsi S, Bouzakis K, Mansour G, Sagris D and Maliaris G 2005 Off-line programming of an industrial robot for manufacturing The International Journal Advanced Manufacturing Technology 26 pp 262–67

[12] Pan Z, Polden J, Larkin N, Duin S and Norrish J 2011 Recent Progress on Programming Methods for Industrial Robots Robot Computing Integrated Manufacturing vol 28(2) pp 87–94

[13] Arao M and Kawaji S 2001 Identification of drilling system using evolving recurrent fuzzy neural networks, IEEE International Conference on Systems, (Man and Cybernetics vol 3) pp 1384-89

[14] Zhu W, Qu W, Cao L, Yang D and Ke Y. 2013 An off-line programming system for robotic drilling in aerospace manufacturing The International Journal of Advanced manufacturing Technology Oct vol 68(9-12) pp 2535–45

[15] Walter K. 2000 Method for characterization of rock strata in drilling operations

[16] Lee D, Siu V, Cruz R and Yetman C 2016 Convolutional neural net and bearing fault analysis Proc. Int. Conf. Data Min pp 194-200

[17] Zheng Y, Liu Q, Chen E, Ge Y and Zhao J 2014 Time series classification using multi-channels deep convolutional neural networks Proc. 15th Int. Conf. Web-Age Inf. Manag pp. 298-310

[18] Maoguo G. et al 2017 Feature learning and change feature classification based on deep learning for ternary change detection in SAR images, ISPRS Elsevier 129 pp 212-225

[19] Peres C. et al. 1999 Fuzzy model and hierarchical fuzzy control integration: an approach for milling process optimization Computer in Industry 39 pp 199-207

[20] https://www.elastic.co/products/kibana

[21] Forrest N. et al 2016 SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size, ICLR