Innovative Remote Device Services Using Deep Learning

Abstract—Remote Device Services is a technology used by equipment manufacturers to monitor and manage systems and products remotely via the Internet. This service has revolutionized the service industry making repairs seamless or nearly seamless from the customer’s perspective thereby enhancing their experience with the product which in turn affects the revenues and profits of the manufacturers. Working in collaboration with Hewlett Packard Inc., a novel remote device service solution is described in detail and implements various image processing tools and deep learning algorithms to detect defects in scanned Print Quality (PQ) test page documents. The detected defects are then compared against a database of defects in order to highlight potential shortfalls within the device environment and also correlate to the root cause of the defect. This content is redacted as the work is covered by a non-disclosure agreement. Please contact the RIT Computer Science Graduate Program Coordinator for more information.

Index Terms—Smart Device Services; Defect Detection; Print Quality (PQ) Test Page; Convolutional Neural Networks (CNN)

I. INTRODUCTION

With improved cloud document management services and advanced computer platforms, manufacturers such as Hewlett Packard Inc., have started to place more emphasis on proactive remote device management also known as smart device services (SDS). Instead of a traditional model where the customer initiates a service call to a support center, and the support center, in turn dispatches a technician to repair the product, an alternative way is to develop an SDS model that can monitor the product remotely on an on-going basis, perform predictive analysis throughout the lifetime of the machine and dispatch a technician once the product cycle reaches a certain lifespan.

Current SDS models involve the use of sensors that are dedicated to monitor the device and uploads information about the product to a cloud server. This information is then used by Hewlett Packard Inc. to analyze the machine and performs repairs if needed. In this paper, we develop an innovative SDS model that implements image processing tools and deep learning algorithms to detect defects on scanned Print Quality (PQ) test page documents thereby eliminating the use of extra sensors which in turn increases the company’s profits.

Defect detection or anomaly detection is an unsupervised learning task where we identify abnormal patterns in data that are infrequent or rare events [1]. The goal of anomaly detection is to define or learn a feature representation that captures spatial appearance patterns. Traditional machine learning algorithms make use of Feature Engineering [2] which is a process of putting domain knowledge into the creation of feature extractors to reduce the complexity of the data and make patterns more visible. This process is difficult in terms of time and expertise and may lead to errors as most of the applied features need to be identified by an expert and then hand-coded as per the domain and data type. Deep learning algorithms have shown to overcome this problem by trying to learn high-level features automatically from the data thereby eliminating or reducing the task of developing a new feature extractor for every problem [2] [3].

In this paper, we use a Convolutional Neural Network (CNN) to build a classifier for detecting defects in PQ test pages. CNNs are widely used in image recognition [4] [5], ii) they require fewer computations due to the sparsely connected neurons and iii) are capable of differentiating a large number of classes [6]. The paper is structured as follows. In Section II, we review some related work on anomaly detection in images. In Section III, we describe in detail the datasets as well as provide information on the structure and operation of the CNN that is used to detect defects. Section IV, presents the experimental results together with a comparison of different training strategies. Finally, Section V, concludes the paper with a brief discussion.

II. BACKGROUND

HP Inc. Smart Device Services (SDS) is a service platform that provides remote management and predictive services for HP managed product dealers. This SDS platform is useful to HP managed product dealers by allowing them to: i) monitor the product on an ongoing basis, ii) perform predictive analysis throughout the lifetime of the machine and iii) dispatch a technician if repairs are necessary once the product cycle reaches a certain lifespan.

The HP Smart Device Services (SDS) consists of three components: i) A HP JetAdvantage Management Connector, installed on a machine at the customer’s end that communicates with devices and with the HP JetAdvantage Management platform, ii) A Managed Print Services (MPS) management tool that has SDS functionality enabled and iii) HP Smart Device Agent to allow monitoring of USB connected printers [7].

The HP JetAdvantage management platform hosted on AWS servers, maintains the data, settings and business logic of a fleet of HP products. The SDS platform integrates with the HP JetAdvantage management platform to enable an extended set of capabilities for these managed fleets of HP products. The capabilities range from device-based functionality such as remote reboot, firmware upgrade, diagnostics and configuration to minimize the number of on-site service visits by HP
managed product service technicians to more advanced predictive service capabilities such as part replacements, training on demand and time required to perform a service so HP managed product dealers can optimize their service visits and maximize their first time fix rate [7].

III. RELATED WORK

Algorithms detecting anomalies in images [8] can be divided into two categories:- reference-free and reference-based algorithms.

Reference-free algorithms do not use a template image to compare with and instead detect anomalies by computing features that discriminate between normal and anomalous regions. For example, D.Carrera et.al [9] detects defects in SEM images of nanofibrous materials by computing features that provide unusual responses to any region in the image.

Reference-based algorithms like the one considered here, detect anomalies by comparing against a test image that do not contain any anomalies and can be used as a template. For example M.Zontak et.al [10] provides a solution for detecting defects in patterned semiconductor chip wafers. The method involves calculating features in the source image by estimating the weighted sum of neighboring features in a reference image. Any feature from defective regions are shown to be unreconstructible from the reference image.

IV. METHODOLOGY

A. Generating the Dataset

The first step in our approach is to generate the training, testing and validation datasets respectively from a sample of 5103x7018 sized PQ test page document images provided by HP Inc. An example of this image is shown in Fig. 2b. Each PQ test page image has a corresponding ground truth image associated with it which highlights the defective regions of the original PQ document as shown in Fig.2a. Since the PQ test pages are too large to be fed in as input to the neural network, the solution is to feed the neural network with small image patches of size 128x128 as shown in Fig.1 of the defective regions only. This way the neural network will learn to detect only the defects in a PQ test page.

The training dataset is generated as shown in Algorithm 1 which is similar to the one discussed in [11]. First we go over each row and column of the ground truth image and obtain the locations of pixels marked as defective in the this image. Next we crop out 128x128 sized image patches from the original image based on the location of pixels obtained in the earlier step and save it in a directory, thus creating our training dataset. Using this algorithm we are able to generate 69,000 training samples, 26,000 validation and testing samples respectively with approximately equal distribution of data among nine defect classes.

Since the performance of deep learning algorithms depends on the size of the datasets, it is important that we generate a sufficiently large dataset of images. Compared to traditional machine learning approaches, performance of deep learning algorithms scale up as the scale of data increases. This is due to the fact that deep learning algorithms need a large amount of data to be able to extrapolate features from a limited set of features in a training set [2] [3].

Algorithm 1 Generating the Training Dataset

Require: A set of scanned original and corresponding ground truth PQ test page images

1: Read in the original and Ground truth PQ test page image
2: while rows < imageheight do
3:     while columns < imagewidth do
4:         Obtain a 128x128 sized patch of the original and ground truth image
5:         if alpha_channel of ground_truth image patch <= 30 then
6:             Save corresponding original image patch in a directory
7:         end if
8:     end while
9: end while

B. Convolutional Neural Network Architecture

The Convolutional Neural Network consists of a succession of convolutional, max-pooling and fully connected layers [12]. It is a hierarchal feature extractor that maps pixels of an input image into a feature vector to be classified by several fully connected layers. Among the various types of CNN architectures, we have used the LeNet architecture, introduced by LeCun et al. [13]. This is due to the fact that the architecture is simple, small, straightforward and a specialized neural network for processing data that has a grid-like topology [14].

CNNs are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers [14]. A typical layer of a CNN consists of three stages. In the first stage, the layer performs several 2D convolutions of an input image with a square filter to produce a set of linear activations [12]. The term convolution is a specialized
kind of linear operation and can be defined as an integral function that expresses the amount of overlap $g$ that is shifted over another function $f$ [14]. In the second stage, the linear activations obtained in the first stage are summed up and passed through a non-linear activation function called Rectified Linear Unit (ReLU) to obtain the activations of the output maps. In the third stage, the output of the layer is modified even further using a pooling layer. The pooling function replaces the output of the net at a certain location with a summary statistic of the nearby outputs. It is useful to reduce the size of the neural network and also make the representation approximately invariant to small translations of the input. After the stages of the convolutional and pooling layers, several fully connected layers combine the outputs into a 1D feature vector. The output layer is always a fully connected layer with one neuron per class. Using a softmax activation for the last layer, we can ensure that each neuron’s output activation can be interpreted as the probability of a particular input image belonging to that class [12].

Our architecture comprises of three convolutional layers. The first convolutional layer contains 32 filters and accepts image patches of size 128x128 as input. The second and third convolutional layer contains 32 and 64 filters respectively and is arranged in a hierarchical order i.e., it accepts as input, the output of the previous layer. Each convolutional layer is followed by a max pooling layer with a window size of 2x2. The convolutional and pooling layers are connected to the fully connected layer containing 128 neurons and finally the output layer contains the softmax activation function that gives the probability of an image belonging to a particular class.

C. Identifying the Defect Type and Cause

Once we obtain a neural network model that is trained to detect defects, the final step is to use this model to predict the defect types and its root cause on a whole PQ test page image. The algorithm to predict a whole image is similar to the dataset generation algorithm and can be divided into two phases. The first phase is to identify the defect type and the second phase is to identify the defect cause. Since the model is trained to predict image patches, we need to feed it with patches. This can be done by going over each row and column of the image, cropping a 128x128 image patch and then getting the probability score for each defect class for that patch. The defect class with a probability of 1.0 predicted the maximum number of times is the overall predicted result of the entire image. Once the defect type is identified, we predict the root cause of this defect by running a trained sub model for each defect class over the entire image and then predicting the defect cause in a way similar to the defect type prediction. Not all defect classes are caused by a specific way. Only three out of nine classes are caused by more than one way and hence a sub model is trained only for these specific defect classes.
<table>
<thead>
<tr>
<th>Defect Type</th>
<th>No. of Images</th>
<th>True positive</th>
<th>True negative</th>
<th>False positive</th>
<th>False negative</th>
<th>Precision</th>
<th>Sensitivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Banding</td>
<td>50</td>
<td>42</td>
<td>1</td>
<td>0</td>
<td>7</td>
<td>42/42=1.0</td>
<td>42/49=0.85</td>
</tr>
<tr>
<td>CPR</td>
<td>48</td>
<td>40</td>
<td>1</td>
<td>0</td>
<td>7</td>
<td>40/40=1.0</td>
<td>40/47=0.85</td>
</tr>
<tr>
<td>Fade</td>
<td>40</td>
<td>35</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>35/35=1.0</td>
<td>35/38=0.92</td>
</tr>
<tr>
<td>Ghosting</td>
<td>45</td>
<td>48</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>48/48=1.0</td>
<td>48/50=0.96</td>
</tr>
<tr>
<td>Streak</td>
<td>50</td>
<td>41</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>41/42=0.97</td>
<td>41/45=0.91</td>
</tr>
<tr>
<td>Shotgun</td>
<td>40</td>
<td>36</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>36/39=0.92</td>
<td>36/37=0.97</td>
</tr>
<tr>
<td>Spots</td>
<td>20</td>
<td>17</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>17/17=1.0</td>
<td>17/18=0.94</td>
</tr>
<tr>
<td>LowSAD</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>20/20=1.0</td>
<td>20/20=1.0</td>
</tr>
<tr>
<td>NA_PASS</td>
<td>50</td>
<td>44</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>44/44=1.0</td>
<td>44/45=0.97</td>
</tr>
</tbody>
</table>

**Algorithm 2** Identifying the Defect type and Cause

**Require:** A set of scanned PQ test page images to be tested

1. Read in the PQ test page image
2. Create a Dictionary D1 with key=defect_names and value=model predictions
3. Create a Dictionary D2 with key=defect_cause_names and value=model predictions
4. while rows < imageheight do
5.     while columns < imagewidth do
6.         Obtain a 128x128 sized patch of the image
7.         Feed this image as input to the trained model
8.         Save the model predictions in D1
9.     end while
10. end while
11. Find max(D1) to identify the defect type
12. Load corresponding sub model based on the result of step 11
13. Repeat steps 4-7
14. Save the model predictions in D2
15. Find max(D2) to identify the defect cause

**V. EXPERIMENTAL RESULTS**

The Neural network model was trained on a NVIDIA GTX 660 with two Gigabytes of GPU memory utilizing 69,000 training sample images of size 128x128 and 26,000 testing and validation samples of size 128x128 respectively with approximately equal distribution of data across nine classes which are Banding, CPR, Fade, Ghosting, LowSAD, Shotgun, Spots, Streak and NA_PASS. Out of the nine classes, eight of them are classes indicating a type of defect in a PQ test page. NA_PASS is a class indicating defect-free images. This is included in order to help the model to learn to differentiate between a defective and a non-defective PQ test page.

Table 1 shows the results of the performance of our model in predicting the type of defect in a full sized PQ test page. 363 PQ test page images of size 5103x7018 were used to evaluate the detection performance of the defect classifier and we can see that the model has a high precision accuracy of 1.0 for seven out of nine classes. Precision here refers to the predictive value positive or the proportion of positives that correspond to the presence of a defect. Precision can be calculated by $P = TP/(TP+FP)$ [15]. Sensitivity shows the ability of the model to detect a defect when the defect is present. It can be calculated by $S = TP/(TP+FN)$. A true positive test result is when the model is able to detect a defect which is present in the image. A true negative test result is when the model is not able to detect a specific defect that is not present in the image. A false positive test result is when the model detects a defect even though that specific defect is not present in the image. Finally a false negative test result is when the model is unable to detect a defect when that specific defect is present in the image [15]. The overall accuracy of the model in detecting the defect type is 92.5% which was more than expected and can be further improved with more images to train as well as a deeper convolutional neural network.

Figures 3, 4 and 5 show the performance of the defect cause classifier trained to identify the root cause of the defect in a PQ test page. Out of nine defect classes, only three types of defects are caused by more than one component failure. Banding and Fade defects are caused by five different component failures and Streak is caused by six various component failures. Hence three separate sub models were trained to detect the cause of Fade, Streak and Banding defects. The overall accuracy of the fade, streak and banding defect cause classifiers are 88%, 87% and 89% respectively. Due to a limited set of testing images, the model’s performance is evaluated only on a small number of images.

**VI. CONCLUSION AND FUTURE WORK**

Using deep learning algorithms, a new SDS approach can be made possible which will revolutionize the service industry, making repairs seamless from a customer’s perspective and also increasing the company’s profits. Using a small CNN architecture and limited hardware, a neural network model with a high precision accuracy is established and can also differentiate between defective and non-defective images.
In the future, instead of using separate neural networks to detect the type and cause of a detect, it can be more efficient to use a single but larger deep convolutional neural network. This can further improve the detection accuracy as well as reduce the time taken to make a prediction on a PQ test page. Also, due to a limited set of testing images as seen in the results section, further testing is required to observe the performance of the model in handling diverse real world PQ test pages before it can be used by manufacturers to provide a novel Remote Device Services solution.

ACKNOWLEDGMENT

I thank Hewlett Packard Inc. for their valuable support and expertise which greatly assisted me in this project. I would also like to thank my advisors - Dr. Reynold Bailey and Dr. Eli Saber for their timely advice and motivation in helping me complete this project.

REFERENCES


