Application of Convolutional Neural Networks in Mobile Devices for Inferring Readings from Medical Apparatus

Prakruthi M K, Vismaya Kalyan V, Suhas S, Manoj G Katti, Satwik Pai N

** Department of Computer Science and Engineering, SJBIT, Bangalore, India.*

***Abstract*: - The field of medicine tends to be a stickler for perfection while handling important records and data for maintaining an efficient patient report and also for her/his further diagnosis. Manual recording of data can be tedious and prone to errors. Thus arises a need for systematic and meticulous retrieval of data. The paper proposes a concept of having an android application which processes the image captured to make it suitable for a convolutional neural network to recognize. The above is achieved using OpenCV, tensorflow and the Inception v3 model. The last layer of the inception model is retrained using a manually curated dataset of images of digits from a seven segment display of medical devices.**

***Keywords*: Convolutional neural networks, Deep learning, Image processing, Inception, Tensorflow.**

I. INTRODUCTION

Health workers everywhere, in hospitals, in medical camps etc. have to manually write down readings from medical devices. In the event these readings have to be used for any inference, statistics, patient profile or anything that involves processing on a computer, the readings have to be entered again. In addition to this, the health workers may have to be trained on the specifics of taking down certain complex readings when a new device comes in. The use of intelligent IoT- capable medical devices is a plausible solution. This would increase costs and moreover the number of devices that are not IoT capable are abundantly used and this trend is not poised to change any time soon. Approaches to several problems have changed since the coming of the smartphone especially now that they have enough processing power to run a deep neural network. We decided to deploy our solution on the android platform since gaining access to a smartphone that deploys it is very easy and relatively cheap especially because we're not proposing the use of any specialized hardware along with it; just the app. This also provides our app with the flexibility of working on various tiers of devices and even on different versions of android. When a user takes a photo of the medical device through the app, the app uses image processing and deep learning; image processing to crop the screen, perform a perspective transform to compensate for any awkward camera angles, isolate individual digits which become the input to the retrained Inception v3 model. We decided to go with deep learning rather than OCR because it reduced the preprocessing steps required and especially eliminates those steps required to deal with different non-ideal situations like uneven lighting etc. Also, with deep learning, it's easier to incorporate new devices as it's just a matter of training the convolutional neural network with images of the new device.

II. EXISTING SYSTEMS

There exists a plethora of systems which are intended to deal with the issue of reading data off of a medical device using images, each of which bring something new to the table while also having a few drawbacks.

In a paper presented by Murlidaran Mariappa[1]proposes the use of a telemedicine robot called OTOROB. It puts forth a method to procure various LCD readings of various medical instruments with the concept of Virtual medical instruments of labView and its OCR module. The results obtained are of higher accuracy and reliability but the OTOROB involves a complex architecture which requires an USB camera to be appropriately placed from the three LCD screens, each of which displays vascular doppler,blood pressure and pulse oximeter and has to be additionally lit by two LED's hence the portability of the device is compromised.The OCR has to be trained multiple times in order to obtain a consistent result which can be time consuming.Even though operating at a lower bandwidth with cost efficiency and higher accuracy, it does face certain issues with portability and placement of the camera from the LCD screens in order to obtain the image, which is a hassle.

Another paper compiled by Kuo-Yi chen[2] highlights the issues of High costs and diverse standards , where each medical device company do not adhere to any universal standard for any uniformity. Thus, it proposes the concept of using an universal reader for all medical sensors irrespective whether it has an inbuilt communication module followed by Digit recognition and its eventual testing and implementation.However it provides a complicated contraption where a camera is mounted on the patient's bed to monitor vitals constantly and the images are to be heavily preprocessed before fed to the OCR model.As inexpensive as a proposed web camera maybe,it does not compare with the ubiquitous nature of a camera equipped smartphone.

Thesis presented by Chang Liu [3] gives a detailed look into image enhancement,noise reduction,detection of region of interest and usage of an OCR model, Usage of Stroke width transformation does produce a lower accuracy with the dataset. However, filtering and chaining part to achieve higher recall rate and Linear Support Vector machine for higher precision rate.It worked well with dot matrix displays and could not function well with seven segment digits unless trained with 10 different kinds of dataset.

A paper by Soumyadip Ghosh[4] tackles a similar issues but it encompasses digital meters in general. It proposes the usage of Feedforward neural network for training of seven segment digit datasets where the images are captured using a webcam and the values are hoarded off to graphs or spreadsheets for further analysis, with the (added advantage of decimal points and signs also being recognized). It also takes into account the distance and orientation of the device when a picture is taken.By analyzing the existing systems a common through line seems to be the issues of either maintaining the portability factor which can be solved by implementing it on a smartphone owing to its omniscient nature and its lack of using deep learning concepts.

III. PROPOSED SYSTEM

3.1 Tesseract

Tesseract is one of the most accurate optical character recognition engines developed by HP and later maintained by Google since 2006. It support various languages and scripts and even detection of mathematical equations. Tesseract is completely trainable; is it possible to train it to recognize a whole new language with numerous fonts. We found that tesseractcouldn'trecognize seven segment digits and hence we had to train tesseract to do so. Tesseract uses .tiff files, which is an image of characters to be trained and .box files whose values represent coordinates of boxes around each character inorder to demarcate them and also contain the textual value of each character. As we could not come across the exact style of font of the seven segment digits we were using, we had to create our own .tiff file using by binarizing digits taken from our device. We achieved this using 3rd party tools like jTessbox editor, seraktesseract trainer. We got fairly good results but not without hassle. Initially, the results were far from perfect and after referring to similar works, we found that tesseract wasn't very efficient in recognizing disconnected characters. We had to resort to eroding the image to connect the segments before sending it off to tesseract for recognition. However the number of passes of erosion varied from image to image depending on the conditions in which the image was taken. Erode too much and the character would turn out too thick to recognize and erode too less, it would stay disconnected. This variation also applies across other functions like thresholding. These conditions for preprocessing pile on with the addition of more devices. Hence we decided to look to machine learning for a solution.

3.2 Tensorflow

Tensorflow is a machine learning library that was open sourced by google in 2015. Since then, it has grown to become the most popular machine learning github project with more than 17,500 commits, around 500 external contributors and appeared in more than 6000 open source projects. It supports various platforms including android. Performing inference on an android device is especially important when it's not always possible for the app to be connected to the internet in order to send data to a server for processing. This was a perfect fit for our app as it could leverage the performance and flexibility of deep learning for better results just on local device resources.

3.3 Inception

The model we use for classification is Google's Inceptionv3. By default it comes trained on millions of images from ImageNet. In certain domains, this model had a lower error rate than a human. We leverage the power of this model through transfer learning; we retrain just the last layer with our dataset of images and preserve the remaining layers. We can do this and still obtain good results because these layers would have learned to recognize features when they were last trained and this can be carried over to other sets of images too. Machine learning makes our task simpler by not needing as much preprocessing as it would otherwise; we now just need to isolate each digit, no thresholding based on numerous scenarios or balancing erosion and dilation. These scenarios can be covered by including images of digits that occur in these scenarios in the dataset that will be used to train the model. The object should be photographed to include multiple camera angles in varied lighting conditions:ideal, low light, uneven lighting, indoor, outdoor and anything else that might occur in real world usage.

3.4 Dataset

Figure 1 shows the devices which we'll be training on; one with backlight and the other one without. We photographed them showing various readings so as to cover all the digits and even the blank when no digit was displayed in that place. For each reading, we photographed the device in several camera angles and numerous lighting conditions as mentioned before and ended up with around 700 images. We ran a python script that read each one of these images, recognized the screen using edge detection and applying contour functions, performed a perspective transform to compensate for any awkward camera angles, cropped just the screen and then isolate each digit along with blanks and stored them for further sorting. This script ignored any images in which it couldn't successfully find the screen like in the case of a heavy tilt of the camera, very bad lighting etc. Also, certain regions were falsely recognized as screens and these segments were weeded out. In certain cases where the camera angle was too off, the resultant digits were cut off in the edges. After cleaning out the anomalies we were left with around 5000 images of digits from our devices to train on.

Figure 1. With and without backlight respectively.

3.5 Training

We trained the inception v3 model using a GPU version of the tensorflow python package on windows. The tensorflow package interfaces with Nvidia CUDA libraries(cuDNN) to perform compute operations on the GPU. The laptop the model was trained on had an NvidiaGeforceGTX 960M with 384 stream processors(CUDA cores), a core i7 6700HQ and 16 GB of RAM. It took less than 15 minutes to train the model and would have taken around an hour on a CPU depending upon the spec.

Figure 2. Cropped digits.

3.6 Sequence of operations performed

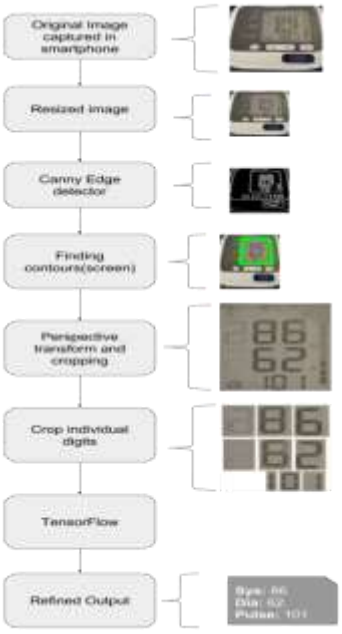

Figure 3. Steps in processing.

Using the app, a picture of the device is taken which is then resized(reduced) while maintaining the aspect ratio. Resizing is done in order to reduce unnecessary details and make processing faster. For edge detection in the image, we found that canny showed better results when compared to sobel. Once all the edges are detected, to identify the find the screen of the medical device, we find all contours present using the findcontour method in OpenCV. As it is evident that screen takes up most of the area in the image we can sort the contours in descending order ie largest to smallest and storing only a small number of contours. The largest contour which is a rectangle is flagged as the screen. The coordinates of this rectangle is used to perform a perspective transform to obtain a flat image and only the screen is cropped and retained. The resulting image is then cropped in respective ratios to extract individual digits from the device each of which is an input to the retrained inception model.

3.7Android challenges

As we tried to port our program to android we came across many challenges that are inherent in mobile devices. Constraints on processing power meant that we had to optimize the graph(the model) for mobile use. The tensorflow library is bundled with several tools that facilitate this. Tensorflow keeps its footprint light on the mobile by only including those operations that required for inference, not training and also doesn't supports certain operations that have large external dependencies. One of the operations that is not supported is the decodejpeg operation which relies on libjpeg which is a hassle to support on mobile devices and increases the binary footprint. Moreover, most mobile applications don't need to decode JPEGs as they deal with camera buffers directly. To deal with this problem, the tool optimize_for_inference removes unnecessary nodes and performs other optimizations that speed up the model. The size of the apk will be quite large when we include the model which is typically around 80MB. The quantize_graph tool quantizes the constants in place which gives more repetition for a compression algorithm to take advantage of with around 1% drop in precision. In our case, the model compressed down to 24MB when packaged in an apk. The app takes around 10 seconds to classify 6 digits on a single thread.

IV. CONCLUSION

With every technology possible, trickling down to the palm of one's hand, deep learning also can be implemented on a mobile device reducing plenty of preprocessing techniques.

Tensorflow is quick and easy to learn and thus is helpful for both developers and users and its open source nature allows for its eventually improvement by the customer as well as developer community. With tensorflow for mobile, we leveraged the power of deep neural networks for robust recognition in various conditions, addition of new devices just by retraining the model and also, a specialised hardware need not be constructed for the purpose of the proposed

implementation. Implementing it on android provides the potential to deliver the technology to anybody who posses a smartphone. The implementation can be taken a step further wherein complicated diagnosis can be handled efficiently with the smaller inferences being handled at the mobile devices and the complex part can be relegated to a specialised server. With machine learning libraries starting to be optimized for mobile and specialized hardware coming to smartphones just to run neural nets, we are just at the shores of an ocean of possibilities.

REFERENCES

[1]. Muralindran Mariappan, Vigneshwaran Ramu, Thayabaren Ganesan, Brendan Khoo, and Kumarheshan Vellian, (2011). "Virtual Medical Instrument for OTOROB Based on LabVIEW for Acquiring Multiple Medical Instrument LCD Reading Using Optical Character Recognition" in *Biomedical 2011 International Conference on Biomedical Engineering and Technology, IPCBEE vol.11*.

[2]. Kuo-Yi Chen, Fuh-Gwo Chen, Ting-Wei Hou(2012). "A Low-cost Reader for Automatically Collecting Vital Signs in Hospitals" in *J Med Syst (2012) 36:2599–2607*.

[3]. Chang Liu(2016). "Digits Recognition on Medical Device". *Electronic Thesis and Dissertation Repository*.

[4]. Sownyadip Ghosh and Suprosanna Shit (2014) "A Low cost Data Acquisition System From Digital Display Instruments Employing Image Processing Technique," IEEE.

[5]. Pranit Poland Ashwini M. Deshpande (2016), "Telemedicine Mobile System". *Conference on Advances in Signal Processing (CASP)*.

[6]. HuiyingShen and James Coughlan (2006)."Reading LCD/LED Displays with a Camera Cell Phone" from Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06).