Grenze International Journal of Engineering and Technology, Jan 2019

Automated Detection of Microorganisms of TINEA using Deep Learning

Aditya Manelkar[1], Varad Jadhav[2], Sanket Alurkar[3], Chitra Bhole[4] and Dr. Shital Amin Poojary[5]
[1-4] K J Somaiya Institute of Engineering and Information Technology/Department of Computer Engineering, Mumbai, India
Email: aditya.manelkar@somaiya.edu, varad.jadhav@somaiya.edu, sanket.alurkar@somaiya.edu, cbhole@somaiy.edu
[5] K J Somaiya Medical College and Research Centre/Department of Dermatology, Mumbai, India
Email: sheetal@somaiya.edu

***Abstract*—Medical Sciences have seen a tremendous advancement in terms of diagnosis, medication, as well as discovery of various new families of disorders and diseases. Diagnosis and treatment of such complicated diseases requires keen observation and highly experienced physicians and doctors. The field of Artificial Intelligence and Machine Learning can very constructively provide a reliable support to doctors for many complicated diseases. There are various scenarios, such as diagnosis of different types of Cancers, where Artificial Intelligence systems are already being used for faster and more accurate detection. Medical Science has various branches, one such being the field of Dermatology. It is a field that primarily deals with the skin, nails, hair, and their accompanying diseases, which being the most exposed parts of the human body, encounter various kinds of fungi, bacteria, and viruses. A skin disease called 'Tinea' is one such disease, that when approached with the traditional method of observing the specimen under a microscope, requires long durations of diagnosis by even experienced doctors. Our work aims at designing an efficient system, that would assist doctors by predicting the presence of the diseases in the specimen in short periods of time and reducing human intervention for every case of Tinea encountered. There are existing models using deep learning, that can be used to implement the approach we are looking at, which we aim to build on using Transfer Learning. We aim to bring a solution to facilitate deployable options for such models and muster high accuracies for them.**

***Index Terms*—AI Artificial Intelligence; KOH Potassium Hydroxide; CSB Chicago Sky Blue; CNN Convolutional Neural Networks; SIANN Shift Invariant or Space Invariant Artificial Neural Networks; GPU Graphical Processing Unit; DDS Deep Diagnose System; GUI Graphical User Interface; HTML Hypertext Markup Language; CSS Cascading Style Sheets.**

I. INTRODUCTION

Tinea is a fungus that is infectious and thrives in warm environments. It affects various areas of the skin. Tinea can be detected and identified by techniques such as examining skin scrapings under direct microscope using potassium hydroxide (KOH) mounts, cultures, or molecular methods. Although, a trained eye is required to infer a KOH mount as it lacks a colour contrast. Culture cultivation is time consuming, costly,

Grenze ID: 01.GIJET.5.1.9

and difficult. Molecular methods including DNA probes are technology intensive, expensive, and not easily available in developing countries; hence used only for research purposes. There are few reports in recent literature [9] on Chicago Sky Blue 6B (CSB) stain being a new promising contrast stain that can be used as a diagnostic method for P. Versicolor.

The above method of diagnosis uses the conventional procedure of observing the specimen using the microscope, not changing the inevitably of requiring an experienced medical practitioner and significant amount of time (if the specimen count or number of cases increase). We aim to optimize pretrained models through transfer learning, ones that are implemented on the concept of Deep learning (CNN). The model would accept the microscopic images as an input and predict the presence of the disease in the specimen, which would be then presented to the end user through an intuitive graphical interface.

A. Convolutional Neural Network

Convolutional Neural Network (CNN) is the methodology implemented here to categorize images. They are specifically designed to deal with the variability of 2D shapes, and they outperform all other techniques. Real-life document recognition systems [1] use a new learning paradigm, called graph transformer networks (GTN), allow such multi module systems to be trained globally using gradient-based methods to minimize an overall performance measure. The CNN image classification takes an input image, processes it and classify it under certain categories, in this case, Positive (presence) and Negative (absence). CNN models pass each input image through a series of convolution layers with filters (or kernels), then pool the features most desired, and continue the process to generate fully connected layers (FC). Activation functions are then applied to classify the image (or an object in the image) with probabilistic values between 0 and 1, which decide the label it is given.

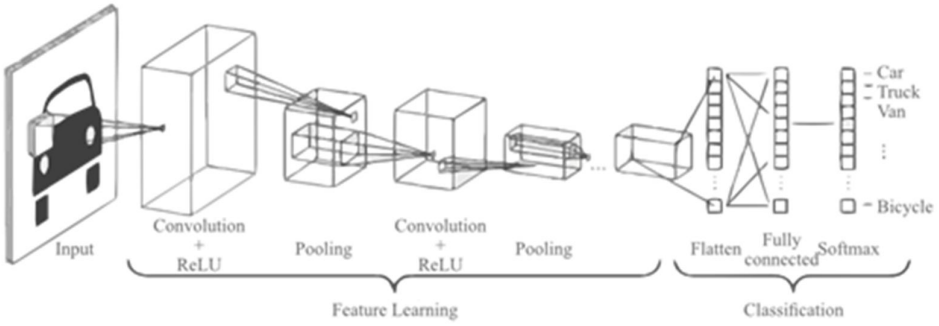

Figure 1. Convolutional Neural Network

Convolution is the first layer to extract features from an input image. It preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel. The convolution of the image matrix when multiplied with the filter matrix results in the 'Feature Map' matrix. Convolution of an image with different filters can perform operations such as edge detection, blur and sharpen by applying filters.

ReLU stands for Rectified Linear Unit for a non-linear operation. The output is $f(x) = \max(0, x)$. Its purpose is to introduce linearity, as the real-world data requires our CNN to learn non-negative linear values.

Pooling layers section reduce the number of parameters or dimensionality size, typically when the images are too large. They do so by retaining the important information. Max pooling (what we have used) takes the largest element from the rectified feature map.

In this layer, we flatten the matrix into vector and feed it into a fully connected layer like neural network. With the fully connected layers, we combine the features extracted to create a model. We use an activation function to classify the outputs as Car, Bicycle, etc., based on probability values.

B. Transfer Learning

Transfer learning is a machine learning technique, where knowledge gain during training in one type of problem is used to train in other similar type of problem. For Deep learning, first few of layers are trained to identify features of the problem. So, during transfer learning, you can remove last few layers of the trained network and retrain with fresh layers for target job. The network can be built from scratch but would require more computations that would take more time and require costlier GPUs for computations. Transfer Learning

uses a neural network trained on a data and the knowledge gained from this data to compile as weights of the network. These weights can be extracted and then transferred to any other neural network, instead of training the other neural network from scratch.
In practice, very few people train an entire Convolutional Network from scratch (with random initialization), because it is relatively rare to have a dataset of considerable size. Instead, it is common to pretrain a CNN on a very large dataset (e.g. ImageNet, which contains 1.2 million images with 1000 categories), and then use the CNN either as an initialization or a fixed feature extractor for the task of interest. It is possible to fine-tune all the layers of the CNN, or it's possible to keep some of the earlier layers fixed (due to overfitting concerns) and only fine-tune some higher-level portion of the network. This is motivated by the observation that the earlier features of a CNN generally contain more generic features (e.g. edge detectors or color blob detectors) that are useful to many tasks. But the later layers of the CNN become progressively more specific to the details of the classes contained in the original dataset.
We aim to discuss through this paper the takeaways of Transfer Learning model implementations that we have tweaked with our own image sets of microorganisms of Tinea (taken under a microscope). We have given a measure of the accuracies as seen in the methodology section and have proposed certain changes that could be incorporated to overcome the shortcomings of the existing system using the pretrained models.

II. Methodology

A. Existing System - Significance

The process began with two choices to decide which model was to be used for the prediction process. The options were: Building a CNN from scratch or using an existing Transfer Learning Model. The first choice required a large number of resources as well as time to train the network but would result in good accuracy due to the architecture specifically designed for the problem. The second choice required a fraction of resources and gave good results. The Transfer Learning model had been already trained on Millions of classes and was generalized. So, going for the Transfer Learning model was the best choice as it required less time and would fit in the project timeframe.
We had to take a CNN pretrained on ImageNet, remove the last fully-connected layer (this layer's outputs are the 1000 class scores for a different task like ImageNet), then treat the rest of the CNN as a fixed feature extractor for the new dataset. This computes a vector for every image that contains the activations of the hidden layer immediately before the classifier. These features are called CNN codes. These codes were ReLUd (i.e. thresholded at zero) during the training of the CNN on ImageNet. Once the CNN codes for all the images were extracted, the classifier was trained for the new dataset.
The models that were shortlisted to be used for this study were:

TABLE I. SHORTLISTED MODELS

Model	Size	Top-1 Accuracy	Top-2 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
InceptionV3	92 MB	0.788	0.944	23,851,784	159
ResNet50	99 MB	0.759	0.929	25,636,712	168

The code was implemented in Python. The development of the software to support this model took place in three stages: Alpha I, Alpha II, and Beta.
Alpha I phase was conducted to validate the working of the InceptionV3 model (Transfer Learning model). The software was totally based on the terminal to predict the output. The InceptionV3 model was successfully able to predict the correct outcome of the input. With an accuracy of 92% on a dataset of Cats and Dogs, the validation of using Transfer Learning as the method of prediction was decided.
In Alpha II, the GUI app was developed with all the properties mentioned in Alpha I. This GUI application was developed to be a Web application. It consisted of the following main actions for the end user: Instructions for scraping the skin, Instructions for Staining the sample, Instructions for Uploading the slide image, and View the Report. Python-Flask along with TensorFlow, Keras, Pandas and NumPy were used to develop the back end of the application. HTML5, CSS and Bootstrap were used to develop the front-end of the app.
The time for prediction was reduced from 30-45 minutes to 10-15 seconds. This was a huge upgrade in reducing the diagnosis time. This phase had a huge upgrade in diagnosis, but the time span required for predicting the diagnosis were about 10-12 minutes. The results were good but not as expected.

Even though the average time for diagnosis reduced from 30-45 minutes to 10-12 minutes, it just wasn't enough. So, getting faster diagnosis was necessary. The shortcomings in the Alpha II version were solved in the Beta phase. The problem of high average diagnosis time is now reduced to 2-3 minutes.
The image set was populated by acquiring the microscopic images from KJSMC. The comparison data was chalked up by performing studies on 750 images, based on which the respective models were trained. The table below shows a study comparing time required for diagnosis and the accuracy each model provided for the 750 image training data:

TABLE II. COMPARATIVE STUDY OF SELECTED MODELS

Phase/Parameters for Comparison	Alpha I	Alpha II	Beta
Average Inference Time (seconds)	10-15	10-15	1-3
Average Diagnosis Time (minutes)	—	10-12	2-3
UI/UX	CLI	Web	Web + Stats
Ease of Use	Hard	Easy	Easy
InceptionV3 Accuracy (%) with 750 Images	68	73	87
Xception Accuracy (%) with 750 Images	61	67	83

B. Problems Encountered

The major problem would be the inaccuracy that the system still possesses (as seen in the comparison). The accuracy must be increased to over 95%, which is currently not possible given two major handicaps: The fist being the limited data set (as of now an additional 700 images to the current 750) which leads to a problem of overfitting, that is even slight noise is added as a learning character. The next problem is the inflexibility of the model; changes to make the model more specific cannot be employed. Hence, certain additions are proposed to work around these problems and increase the accuracy of prediction.

C. Proposed Changes/Additions to the System

In order to fit our CNN to the image set, we are going to pre-process the images to prevent overfitting [11]. Overfitting in our specific model pertains to our great training accuracies but relatively poor test accuracies, owing to overfitting of nodes from one layer to another.
Since we cannot radically improve the size of our training set for the neural network, we plan on performing additional augmentations on them, which basically equates to synthesising the training data. We are going to do this using the *keras.preprocessing* library for the synthesising part and preparing the training set as well as the test set of images that are present in our directories, where the directory's name is take as the label of all the images present in it. For instance, all the images inside the folder named 'Positive' will be considered as positive cases of Tinea by keras.
Another method that can be deployed is using visual information of landmark locations in the prior stages of the model to be provided as a heatmap [10]. Using entire slide image rather than divided block allows the algorithm to handle detections with much more differences and difficult initializations. A landmark heatmap detects the features in the image and matches accordingly with the model. In our case, these features include the hyphae and spores around it. A cluster of hyphae of Tinea can be found at multiple layers in the same test slide. Thus, these groups can be used to create high intensity valued landmark heatmap around landmark locations where decreasing intensity from the closest landmark.
Heatmap is used by the CNN to calculate the landmark location possibilities in the image. Cascaded Shape Regression (CSR) model iterations are characterized by (1), where St is the estimate of landmark locations at iteration t, rt is a regression function returning the update to St given a feature φ extracted from image I at the landmark locations:

$$St+1 = St + rt\ (\varphi\ (I, St)) \qquad (1)$$

Landmark heatmap is represented by an image in which the intensity is highest at the landmark locations and gradually decreasing w.r.t distance from it. Equation (2) is used to generate it where H is the heatmap image and si is the i^{th} landmark of Tt (St − 1):

$$H(x, y) = \frac{1}{1 + \min s_i \in T_t (S_t - 1) \,||\, (x, y) - s_i \,||} \quad (2)$$

A feed-forward neural network layer connects the Transform Estimation layer, Image Transform layer, Landmark Transform layer, Heatmap Generation layer and Feature Generation layer. Transform generated in the Transform estimation layer is used to wrap the input image and the current landmark estimate close to the canonical shape. The transformed landmarks are passed on to the heatmap generation layer. The feature image layer is an image created from a dense layer with connections that allow any information learned by the previous stage to be transferred to the consecutive stages. This transfer of knowledge about landmark locations is also carries out in heatmap generation.

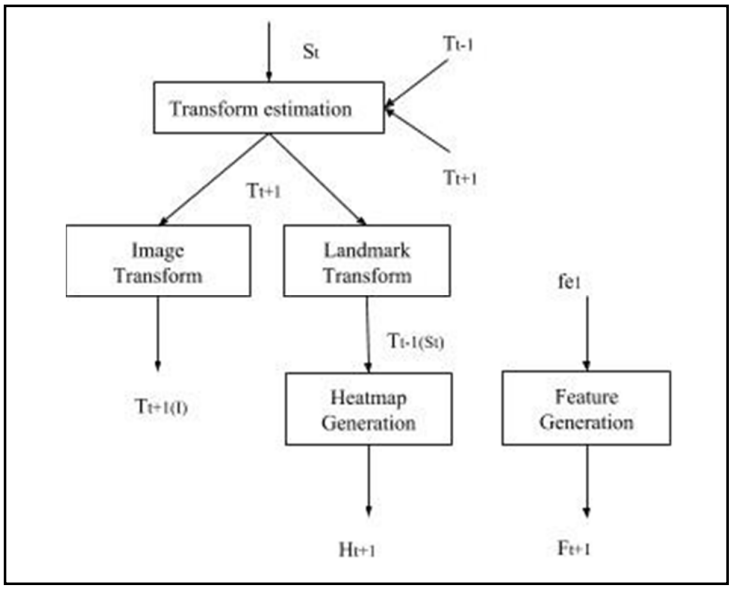

Figure 2. Heatmap Generation Process

III. Conclusions

The system proposed for detection of Tinea predicts whether the image input belongs to one of the two defined classes: Positive or Negative. A comparative study on using transfer learning models through Inception and Xception has provided the result that the Inception model gives a greater accuracy as compared to the Xception model. The models both lack the accuracy required due to rigidity over large pretrained weights. We have proposed methods to deal with the problem, and possibly increase efficiencies.

Acknowledgment

We would like to thank Prof. Chitra Bhole (Department of Computer Engineering) and Dr. Shital Amin Poojary (Department of Dermatology, K J Somaiya Medical College and Research Centre) for their patience, encouragement, guidance and cooperation provided in the form of unparalleled facilities.

References

[1] Y. Lecun, P. Haffner, L. Bottou, and Y. Bengio, “Object Recognition with Gradient-Based Learning,”, 1999.
[2] Goodfellow, Y. Bengio, and A. Courville, Deep learning. Cambridge, MA: The MIT Press, 2017.
[3] G. James, D. Witten, T. Hastie, and R. Tibshirani, An introduction to statistical learning with applications in R. New York: Springer, 2017.
[4] S. Raschka and V. Mirajalili, Python machine learning: machine learning and deep learning with Python, scikit-learn, and TensorFlow. Birmingham, UK: Packt Publishing, 2017.
[5] Kirill. Eremenko, “Machine Learning A-Z.”, SuperDataScience Team, Udemy 2017.
[6] Kirill. Eremenko, Hadelin de Ponteves “Deep Learning A-Z - Udemy.”, SuperDataScience Team, Udemy 2017.
[7] Jose Portilla, “Python for Data Science and Machine Learning Bootcamp” Udemy 2017.

[8] Nikita Lodha, Shital Amin Poojary, “Novel Contrast Stain for the Rapid Diagnosis of Pityriasis Versicolor: A Comparison of Chicago Sky Blue 6B Stain, Potassium Hydroxide Mount and Culture”, Indian Journal of Dermatology, 2015.
[9] Marek Kowalski, Jacek Narunicc, and Tomasz Trzcinski, “Deep Alignment Network: A convolutional neural network for robust face alignment”.
[10] Image Preprocessing - Keras Documentation (https://keras.io/preprocessing/image)
[11] https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/
[12] https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/
[13] Simple Image Classification using Convolutional Neural Networks - Deep Learning in python: beinghuman.ai