2021 IEEE International Conference on Computer Science, Electronic Information Engineering and Intelligent Control Technology (CEI) | 978-1-6654-3881-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/CEI52496.2021.9574583

Full-Stack Application of Skin Cancer Diagnosis Based on CNN Model

Yiyang Huo*
College of Letter and Science, University of California Davis, Davis, CA, USA
*Corresponding author email: guanghua.ren@gecacademy.cn

***Abstract*—Convolutional neural network (CNN) is a subset of deep neural networks, and it has commonly applied to analyze images. Skin cancer is a disease that can be observed without the help of expensive and professional instruments. At the same time, consulting a doctor in a hospital is not a cheap thing for many people. To tackle this issue, this paper introduced a skin cancer detection application based on CNN. In this application, the graphical user interface is implemented by Swift UI, and the backend is ExpressJs. It loads a keras CNN model through TensorFlow JS to detect and classify skin cancer. The CNN model used in the application is created through TensorFlow and trained with the HAM10000 skin cancer data set. It also integrated Natural Language Process (NLP) function, so that users can ask some questions and consult doctors online. The current version of this application can successfully operate in the IOS environment, and fully achieve those functions above. Also, the experimental result demonstrated that the accuracy of CNN model is around 75% for HAM10000 test set.**

Keywords—CNN model, Skin cancer, IOS app

I. INTRODUCTION

Deep learning is currently one of the hottest topics, which can be applied to a number of scenarios [1]. Meanwhile, as medical technology develops, people care more about their own health than before. For instance, people sometimes would worry about nodules or black spots on the skin, which looks similar to skin cancers. To assist the public to check conditions by themselves, healthcare apps with deep learning algorithms started to emerge in the market [2]. For example, various health apps like cancer detection can be found in app galleries. For most current applications, deep learning models are well constructed and modified with professional datasets, which means that the accuracy of prediction is expected to be high enough for self-diagnose [3]. Moreover, when talking about skin cancer detection app specifically, it can be seen that people can simply check their own conditions by just taking pictures of the skin, and then receive the result in the app or via email immediately. Image identifications in such apps are achieved with the help of convolutional neural network (CNN), which is a well-developed and widely used algorithm [4]. However, some drawbacks appear in the existing apps. The user interface of several apps is not good-looking enough and some of them lack interaction with users, which would assist users to understand situations better. Therefore, some modifications such as adding live communication function utilizing Natural Language Process (NLP) and setting feedback as well as history search would be beneficial. In this context, it aims to design a full-stack application of skin cancer diagnosis based on CNN model [5].

The main aspects of this work can be summarized as follows:

1. Design and implement a full-stack application to diagnose skin cancer with modern user experience.

2. Trained CNN Model with small dataset to effectively classify skin cancer.

3. Based on existing HAM10000 dataset [6], improve it with more content and build a system that constantly enrich the dataset.

4. Implement a chatting system that supports interactive chatting powered by Google Dialogflow service [7] and real-person customer service powered by NodeJS WebSocket [8-10].

The rest of this paper is organized as follows. In Section 2, the comprehensive service framework is introduced and explained in detail. In Section 3, the corresponding results of detection and classification are compared and analyzed. Finally in section 4, this paper is concluded.

II. MODEL FORMULATION

The overall service provides contains two parts, the training part and service part. The training part trains a CNN model to do object detection to identify and classify skin cancers, and by collecting the adjusted feedback provided by user, the model will update itself to make better prediction.

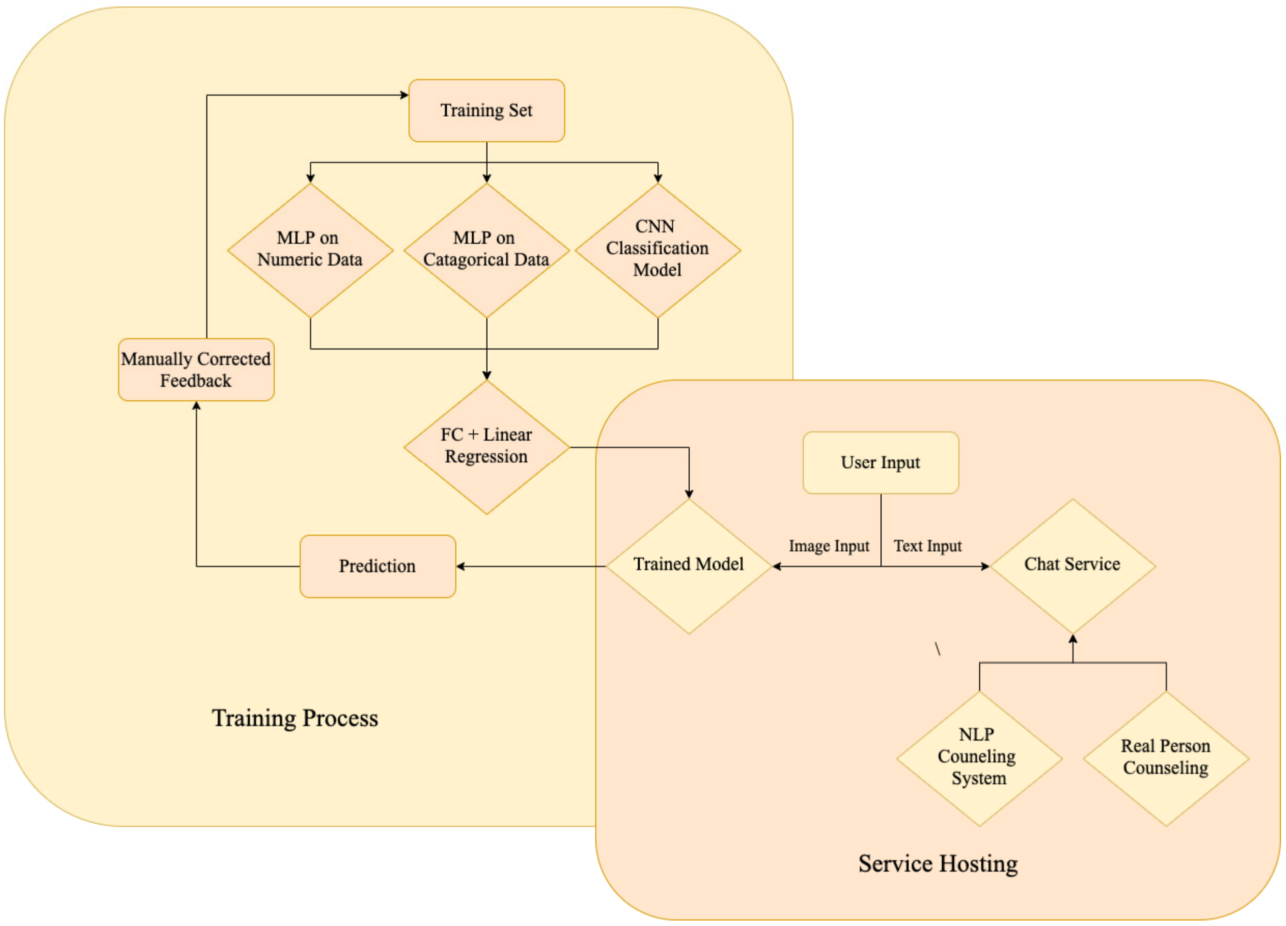


Figure 1. The overall framework of service structure

A. Training Process

This process creates a model before offering the service using CNN model. Then, it constantly updates itself to create more efficient model.

1) Training

Using HAM10000 and more image data of a healthy person's arm as default data set, this part first create a CNN classification model. This model can identify Melanocytic Nevus, Melanoma, Seborrheic Keratosis, Basal Cell Carcinoma, Pyogenic Granuloma and Actinic Keratosis. Then, consider the huge difference between HAM10000 and the actual input of user's image input. HAM10000 contains images with front captures of diseased areas that occupies a large amount of area. However, the diseased area of actual user input may not occupy a major part of the input image. As a result, this paper will transfer CNN classification model into an image detection model to deal with huge number of unexpected noises created by user inputs. However, comparison is still needed to test whether turning into an object detection model optimize the classification. In the experiment part, this paper will provide the experiment data in detail.

2) Feedback

Considering a huge number of user input, it is beneficial to make full use of this input to improve this model. Since the pre-diagnose service this paper offers plans to integrate national medical service system. As a result, it can easily retrieve the real skin cancer users have if they go to the cancer specialists. Meanwhile, it can obtain new dataset by creating a feedback system that let doctor submit the actual result of the skin cancer. In this way, this model can be improved as the service continues.

B. Service Hosting

This paper suggests and implement a service system that not only provide API for users to quickly get access to the model and retrieve predicting results described in part 2.1, but also host a chatting system that not only include NLP but also real person chatting system to offer a more friendly pre-diagnose support for users. This part hosts a service that has been implemented by a complete front and back end. For front end, this research firstly chooses implement in iOS platform because an iOS app has a huge market and complete service that can let users take picture easily. For backend, this paper chooses the NodeJS to host the service. The hosts also use google Firebase to store users' text and image input. Firebase also offers an account system for users to track their history diagnose results. The graph of the architecture of the hosting service is listed below:

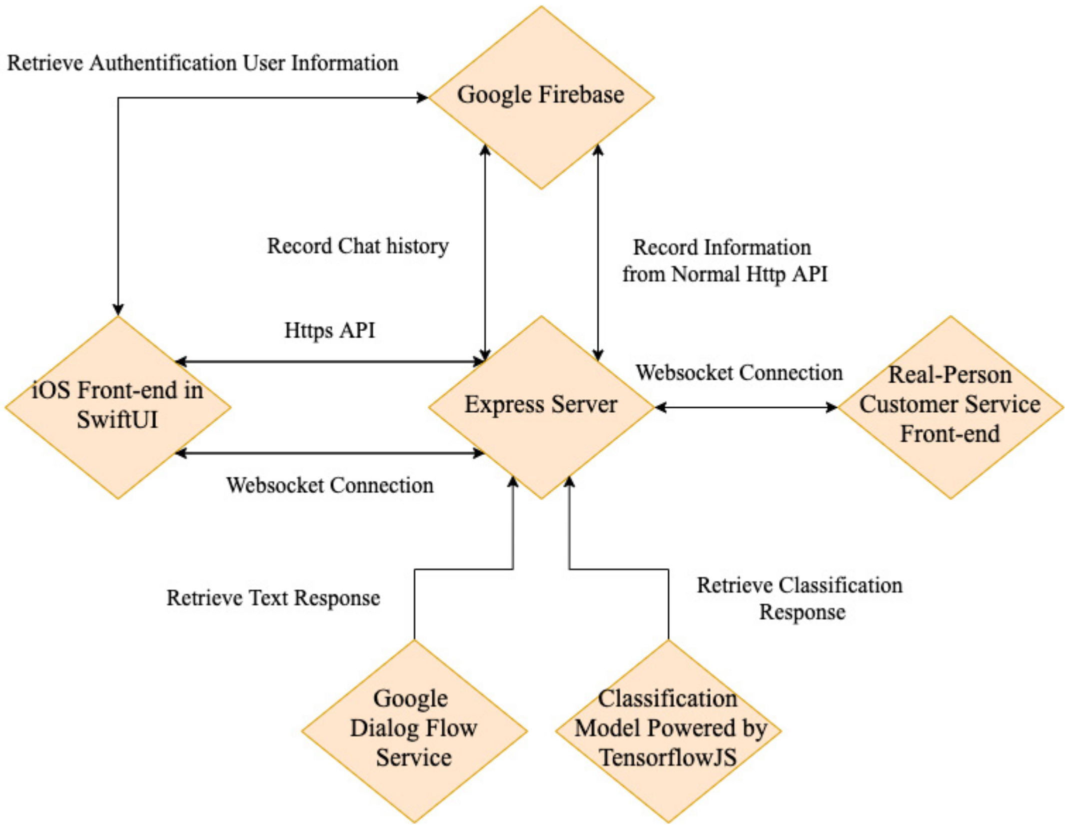

Retrieve Authentification User Information
Google Firebase
Record Chat history
Record Information from Normal Http API
Https API
iOS Front-end in SwiftUI
Express Server
Websocket Connection
Real-Person Customer Service Front-end
Websocket Connection
Retrieve Text Response
Retrieve Classification Response
Google Dialog Flow Service
Classification Model Powered by TensorflowJS

Figure 2. The architecture of hosting service

1) *Model Hosting Service*

Although Keras is used to train the model, it can host this model using TensorFlowJS library to host the trained model. Users firstly upload their image to the server. The server processes the image, and offers prediction using the model it hosts. Then, the server communicates with Firebase and store the prediction and the image into the system. Finally, the server sends the result back to the frontend and the logged in user can browse their history diagnose result.

Besides, like what has been discussed in 2.1.2, logged in user can submit the feedback for any history results. The server will then communicate with Firebase and store the actual result in a separate database. Then, the developer can get new dataset by extracting data from this new database.

2) *Chatting Service*

This service system also integrates a chatting service to answer questions of users that is related to skin cancer. The chatting service contains two parts: the NLP chatting system and a real person chatting system.

For NLP chatting system, this service use the Dialog flow API provided by Google. The dialog flow can not only extract meaning from input text and generate the text feedback, but also send signal back to the server to do API call. By the time this paper is written, the offered signal service contains 1: deleting chatting or history diagnose records of the user. 2: asks the host to make a chatroom with a real person.

The real person chatting system can answer questions that is not covered well in NLP system. Here, this study uses web socket in order to build separate chat room for a single user and a single customer service staff.

III. EXPERIMENT

To train the model, this research uses Google Colab platform. Besides, Keras is used to implement CNN model, a MLP model to deal with categorized value, a MLP model to deal numeric value and a concatenate layer to combine three models together to form a one-hot encoding output.

A. *Datasets*

For dataset, this study uses HAM10000 skin cancer dataset. For the graphic part that is supposed to input in CNN, this research transfers all 600*450 image to 300*225 and normalize its three-color channels. For categorical MLP, it one-hot encoded prepared gender, place that took pictures, and dx type. For continuous MLP model, the age is normalized treated as input. To better see the training result, it separates the dataset into training set and validation set. There are 10016 entries in total, this study randomizes the order of the entries and separate the first 8000 entries as training set and the rest as the validation set.

B. *Evaluation Metrics*

These experiments employed two evaluation metrics (Loss and ACC) to measure the performance of the proposed model, which are represented in Eq. (1) and Eq. (2).

$$Loss = -\frac{1}{batch_{size}} \sum_{i=1}^{batch_{size}} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log(1 - \hat{y}_i) \qquad (1)$$

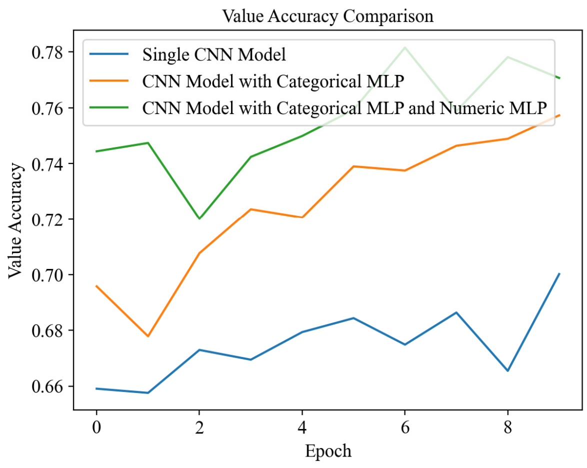
where $\hat{y}$ means the estimated label of the model given input, y means the actual label, and batch size means the batch size in each epoch.

$$ACC = \frac{1}{batch_{size}} \sum_{i=1}^{batch_{size}} [indexofmax(\hat{y}_i) == indexofmax(y_i)] \qquad (2)$$

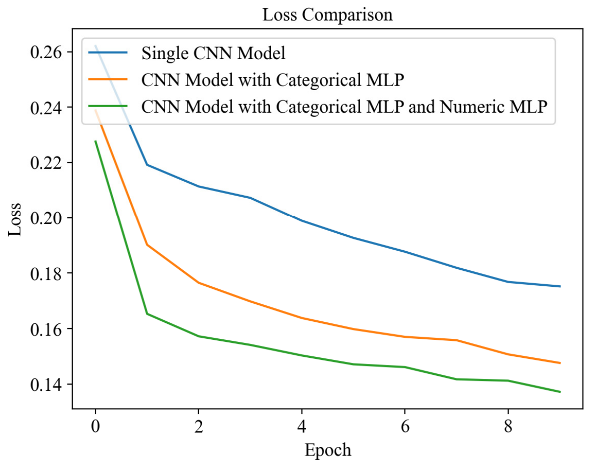
where *indexofmax* indicates the max index of the one-hot label, $\hat{y}$ indicates the estimated label, y represents the actual label, and batch size is the batch size in each epoch.

C. *Experimental Results and Analysis*

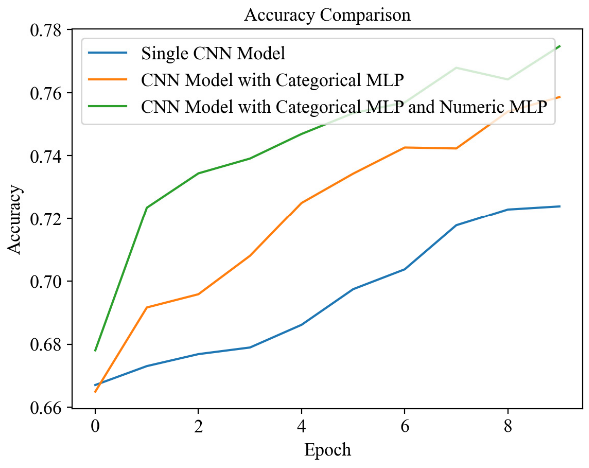
In order to test how effective the multi-input model is. This paper trained three models, one is only CNN classification model, one is CNN and categorical MLP and one is CNN plus categorical and numerical MLP. To control the variables, this paper uses the same seed to randomize the order of the HAM10000 dataset to ensure that all three models receive the same training set and validation set. And as what the paper has demonstrated above, the evaluation metrics are Loss and ACC that follows Eq. (1) and Eq. (2). After training, the graphic representation of the experiment result is shown below.



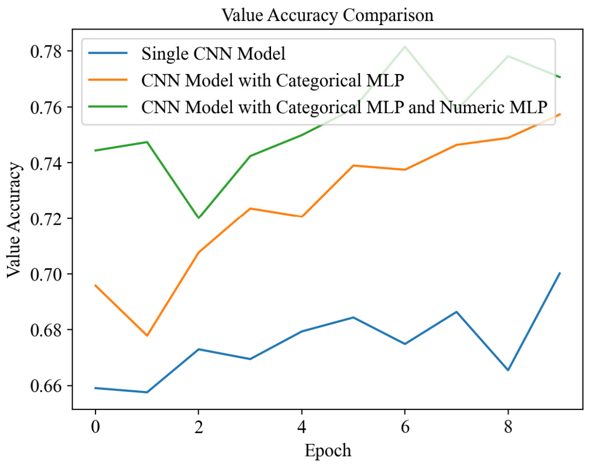
(a)



(b)



(c)



(d)

Figure 3. Comparison of the loss and accuracy of different models (CNN model, CNN model with caegorical MLP, CNN model with categorical and numeric MLP)

According to the experiment result, there is a magnificent improvement not only by adding categorical MLP, but also in numeric MLP. By adding validate data, it is also clear that overfitting do not happen in the result. The most significant improvement is the numeric MLP since the improvement is the same even by adding just one variable age in the model instead of three in categorical MLP. Besides, it could be seen that the value accuracy goes to 0.78 in validation. A model with this accuracy is fit to put into the service framework mentioned in Figure 3.

IV. Conclusion

An IOS skin cancer detection app based on CNN was developed to help people check vitals particularly in skin cancer on their own with low expenses. With the help of Swift UI and ExpressJs, CNN model was loaded using TensorFlow and trained with HAM10000 dataset. To improve the user experience, a chatbot including feedback and history search was integrated to the app using NLP.

In the future, it is possible to upgrade this skin cancer detection app based on the current work and issue. For instance, the feedback function could be enhanced by cooperating with hospitals and clinics directly. To be more specifically, after seeing the doctor, the report of the patient who uses the app would be automatically synchronized to the database of ours, which would help to improve the performance in terms of prediction. Moreover, patients would easily pay the bill, if the insurance system is connected to the app, and the app would also give an estimation of the cost of treating disease. Furthermore, the app is only available on IOS at the moment. It is necessary to develop more versions so that it could be used on different platforms such as Android and Harmony as well as detect more diseases in only one app to make it a powerful and useful tool before receiving instructions from doctors.

References

[1] N. Tziolas, N. Tsakiridis, E. Ben-Dor, J. Theocharis and G. Zalidis, “Employing a multi-input deep convolutional neural network to derive soil clay content from a synergy of multi-temporal optical and radar imagery data,” Soil Properties Using Imaging Spectroscopy, vol. 12, no. 9, pp. 1389, 2020.

[2] D.-G. Shen, G.-R. Wu, and H. Suk, “Deep learning in medical image analysis,” Annual Review of Biomedical Engineering, vol. 19, pp. 221-248, 2017.

[3] T. Saba, M.A. Khan, A. Rehman, and S.L. Marie-Sainte, “Region extraction and classification of skin cancer: a heterogeneous framework of deep CNN features fusion and reduction,” Journal of Medical Systems, vol. 43, no. 289, 2019.

[4] G. Papandreou, I. Kokkinos, and P.-A. Savalle, “Untangling local and global deformations in deep convolutional networks for image classification and sliding window detection,” Computer Vision and Pattern Recognition, https://arxiv.org/abs/1412.0296, 2014.

[5] A.A. Nugroho, I. Slamet, and Sugiyanto, “Skins cancer identification system of HAM10000 skin cancer dataset using convolutional neural network,” AIP Conference Proceedings, vol. 2202, no. 02003.

[6] P. Tschandl, C. Rosendahl, and H. Kittler, “The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions,” Scientific Data, vol. 5, no. 180161, 2018.

[7] N. Sabharwal and A. Agrawal, “Introduction to Google Dialogflow,” Cognitive Virtual Assistants Using Google Dialogflow, https://doi.org/10.1007/978-1-4842-5741-8_2, 2020.

[8] I. Fette and A. Melnikov, “The WebSocket Protocol,” https://www.hjp.at/doc/rfc/rfc6455.html, 2011.

[9] V. Wang, F. Salim, and P. Moskovits, “The definitive guide to HTML5 WebSocket,” 2013.

[10] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen. “Compressing neural networks with the hashing trick”, International conference on machine learning, pp.2285-2294, 2015