Cloud-based Skin Lesion Diagnosis System Using Convolutional Neural Networks

E. Akar, O. Marques, W.A. Andrews, and B. Furht
Florida Atlantic University, 777 Glades Road, Boca Raton, Florida 33431
bfurht@fau.edu

Abstract. In this paper we developed cloud-based skin lesion diagnosis system using convolutional neural networks, which consists of the following:

- Deep learning based classifier that processes user submitted lesion images which runs on a server connected to the cloud based database
- Deep learning based classifier performs quality checks and filters user requests before the request is sent off to the diagnosis classifier
- A mobile application that runs on Android and iOS platforms to showcase the system

We designed and implemented the system's architecture.

Keywords: skin lesion diagnosis, early melanoma detection, convolutional neural networks, cloud-based system, mobile application

1. Introduction

Development of AI may have significant impact in the future in important fields like medicine which may help to provide more efficient and affordable services to patients around the world. Intelligent diagnosis systems based on AI can aid doctors in detecting diseases. Recently, deep learning methods have been explored to provide solutions to image-based diagnosis tasks such as skin lesion classification. Effective diagnosis systems for lesion detection can improve diagnosis accuracy, which inherently reduces the number of biopsies made by dermatologists to determine the nature of the lesion, and can also aide with early detection of skin cancer like melanoma which can turn fatal if left undiagnosed and untreated. The diagnosis system may run on different types of personal devices like mobile phones and tablets to maximize its access to patients around the world.

On the research end of deep learning based diagnosis systems, there is progress being made, but a full, production-level diagnosis system is still lacking in the field of dermatology. As such, the focus of this thesis is the design and implementation of a skin lesion diagnosis system powered by convolutional neural networks and based on cloud architecture. Front-ends for this system can be made to run on multitude of devices including mobile phones, tablets, and personal computers. A mobile application has also been developed as a front-end to showcase the system. In this paper we developed cloud-based skin lesion diagnosis system using convolutional neural networks, which consists of:

Background on skin cancer, the magnitude of the problem, and current deep learning based diagnosis solutions are described in section 2. In section 3, the design and implementation of the diagnosis system and its components are explained. In sections 4 and 5, the architecture, training, and results of the deep learning based diagnosis and preliminary classifiers are discussed. The design and implementation of the mobile application demo is presented in section 6. Future work is discussed in section 7.

2. Background

Skin cancer is a major medical problem. Every year in United States alone 5.4 million skin cancer cases are reported [1, 2, 3]. In United States $8.1 billion is spent annually for skin cancer treatment, $3.3 billion of which is the total cost of treatment for melanoma alone [4]. Melanoma is responsible for 75% of annual deaths due to skin cancer, claiming 10,000 lives annually although it only makes up 5% of skin cancer cases [1, 5]. Early detection for melanoma is vital for survival. If detected in its earliest stages, survival rate for melanoma is extremely high at 99% which fall down to only 14% in its latest stage.

As shown in Figure 1, visual differences between different skin lesions, especially melanoma and benign lesions, can be minute. There are several visual methods used by dermatologist to diagnose melanoma without biopsy, but the accuracy of these methods are poor, only around 60% [6]. Since these methods are not reliable, biopsies are generally required for diagnosis. As a result, computer aided visual diagnosis systems made available to dermatologists and patients may provide more accurate and convenient methods of diagnosis to improve the rate of early detection of skin cancer and reduce the number of diagnosis tests.

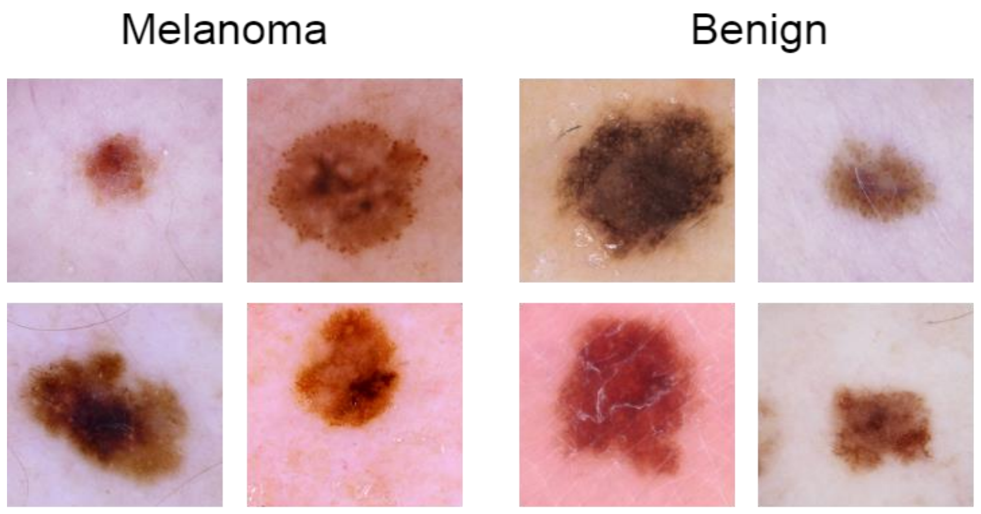


Figure 1. Visual similarities between melanoma and benign lesions make it hard for dermatologists and machine learning algorithm to visually identify the lesions. Sample images taken from the "ISIC 2018: Skin Lesion Analysis Towards Melanoma Detection" grand challenge datasets [14, 15].

2.1 Deep Learning in Lesion Detection

Deep learning, specifically convolutional neural networks (CNNs), have rapidly become the technique of choice in computer vision. This is due to the contribution of Krizhevsky et al. winning the ImageNet competition in 2012 and other more recent successes of CNNs improving the state of the art in many domains. This is becoming increasingly the case for many visual problems in the field of medicine as well.

To tackle the problem of skin lesion diagnosis, several authors have used CNNs. Training the CNNs were done using only dermoscopic and photographic images as input along with image labels in a single CNN [7, 8]. In the case of Esteva et al., the trained network was tested against 21 board-certified dermatologists and achieved expert level performance; a feat, according to the authors, have not been achieved before [8].

One of the general downsides of training neural networks is the requirement for a large training set. Another challenge for this specific task is that images of lesions may vary in factors like lighting and zoom or even feature the same skin disease in different stages which may appear different. Esteva et al. were able to overcome these challenges by collecting almost 130,00 clinical images and 3370 dermoscopy images. They utilized GoogleNet Inception v3 CNN architecture that was pre-trained on 2014 ImageNet Large Scale Visual Recognition Challenge and trained the network with the collected lesion images using transfer learning [9, 10, 11].

Esteva et al. also point out that with billions of smartphones in use around the world that can be equipped with deep neural networks, low-cost access to diagnostic services can be provided universally.

3. The Diagnosis System Architecture

The system consists of the client device, cloud database, and server that is connected to both cloud database and the script that loads the CNN, as shown in Figure 2. Any device that has internet connection could connect to the cloud database and upload lesion images to be processed. At the end of the diagnosis process, a skin lesion classification and the confidence score computed by the CNN is reported back to the client.

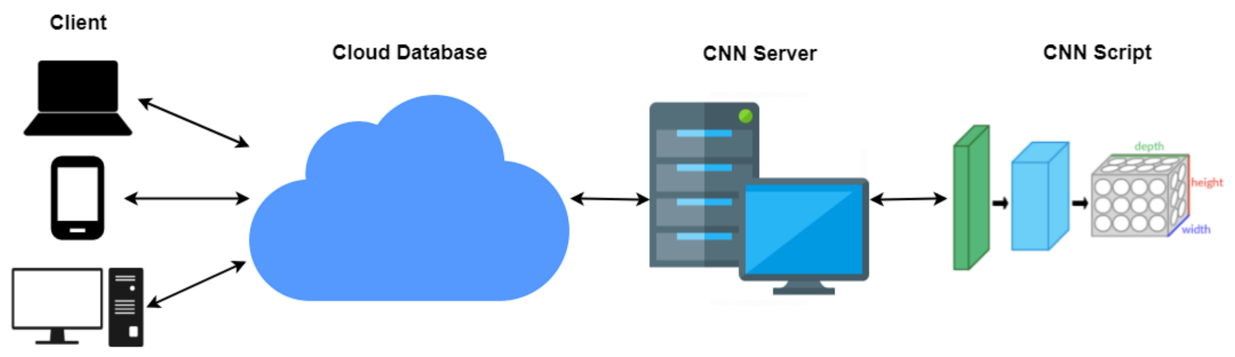


Figure 2. Cloud-based diagnosis system architecture.

Once an image is uploaded from a device to the cloud database, the CNN server is notified which then downloads the image and informs the CNN script to process the image with the already trained CNN. Once a result is acquired from the CNN, such as melanoma or benign lesion, the result is sent back to the CNN server which updates the cloud database which then updates the client with the lesion results.

3.1 Client

Applications running on client devices is relatively simple as the only requirements are connection to the cloud database and client-side image resizing. The architecture of the system was designed to keep client memory and processor requirements minimal and be accessible to a large range of devices with varying specifications. The issue of slow internet upload and download speed is also mitigated by utilizing client-image image resizing which resizes lesion images to the exact size as required by the CNN. As a result, patients and physicians who may have access to different types of devices and internet connection can employ the system with ease.
The application running on devices with cameras can also utilize the camera and handle the task of taking images of lesions. This has been implemented in the mobile phone application written for this thesis. The details of the mobile app is discussed in the next paragraph.

3.2 Cloud Database

The task of the cloud database is to establish connection between client devices and the CNN server and store and serve data coming in from both directions. It must be able to connect to a large range of devices, store lesion images uploaded from clients into a filesystem and capture results for these images sent from the CNN server into a database. Google’s Firebase has many services that offers these features so we elected to build our system with it, specifically with Cloud Firestore and Cloud Storage Firebase services [14]. Firebase provides client-side libraries for applications that run on Android, iOS, and JavaScript, allowing mobile, web, and server applications to integrate with this system. Firestore is a realtime database service that synchronizes data events across applications, and Cloud Storage handles the storage and serving of files which are lesions images in our case.
When a diagnosis for an image is requested, a new document with a unique id is created in Firestore, and the image is uploaded with this id as the filename. The document initially stores the id, timestamp, and the location of the uploaded image in Cloud Storage. In the next step, the CNN server gets notified of the document and handles the rest of the process.

3.3 CNN Server

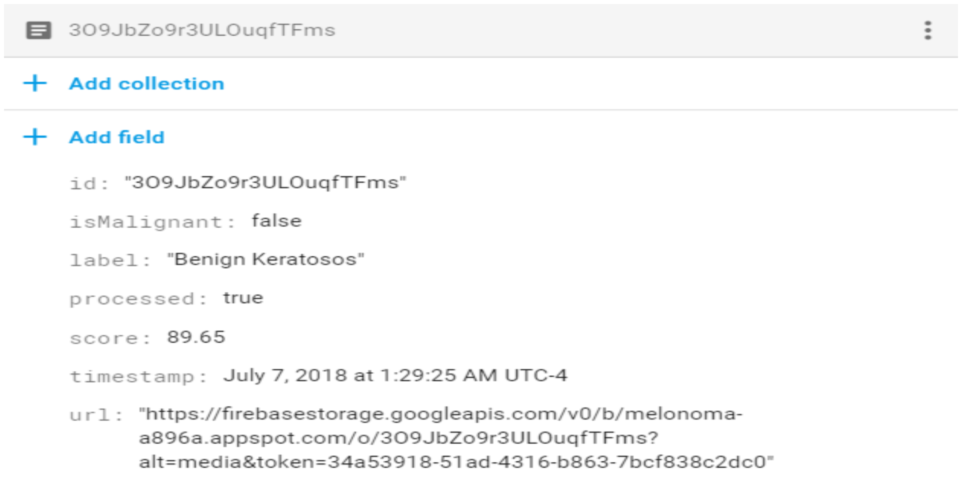
The CNN server is written in JavaScript for Node.js, and acts as a task manager for the CNN script. Since Firestore synchronizes all connections, the CNN server which is already listening for changes in Firestore is notified immediately of requests in the form of newly created documents. The server then retrieves documents that have not yet been processed, and using the id of the documents, downloads the corresponding lesion images as shown in part 2 in the source code of the Node.js server shown in Figure 3.

```
const admin = require('firebase-admin');
const keys = require('./key.json');
const spawn = require('child_process').spawn;
const fs = require('fs');
admin.initializeApp({
    credential: admin.credential.cert(keys),
    databaseURL: "https://melonoma-a896a.firebaseio.com",
    storageBucket: "gs://melonoma-a896a.appspot.com"
});
const db = admin.firestore();
function processUpload(id) {
    jobs[id] = true;
    py.stdin.write(JSON.stringify({id}) + '\n'); //send off to script to process
}

let jobs = {};

// 1) spawn the CNN script
const py = spawn("python", ["-u", "app.py"]);
py.stdout.setEncoding('utf-8');
py.on('exit', () => {
    console.log('PYTHON SCRIPT ERROR');
});

// 2) listen for new documents in Firestore and download images
db.collection('uploads').where('processed', '==', false).onSnapshot(docs => {
    docs.forEach(doc => {
        if (jobs[doc.id] !== undefined) return;
           admin.storage().bucket().file(doc.id).createReadStream()
           .pipe(fs.createWriteStream(`tmp/${doc.id}`).on('finish', () => {
             processUpload(doc.id);
        }));
    });
});

// 3) returned results from the CNN script
py.stdout.on('data', (data) => {
    let result = JSON.parse(data); // json result
    result.processed = true;
    console.log(result);
    fs.unlink(`tmp/${result.id}`, (error) => { /* handle error */ }); // delete file
    db.doc(`uploads/${result.id}`).update(result).then(() => { // update change
        delete jobs[result.id];
    });
});

process.stdin.resume();
```

Figure 3. Source code of the Node.js.

Using the document IDs, the uploaded images stored in Cloud Storage are downloaded into a temporary location in the filesystem. The CNN script is a child process of the CNN server and is spawned at the initialization of the server in part 1 of the source code. The communication between the script and server is done using standard IO between parent and child processes. Notifying the CNN script to processes a request, handled by the processUpload function, is done by writing the id of the document to the standard input of the CNN script. In part 3, results from the script in the form of stringified JSON are written to the standard input of the server. Next, the JSON response is parsed, and with the extracted id, the corresponding document is updated with the results from the script. For clean-up, the downloaded image in the temporary location get deleted.

As shown in Figure 4, the updated document includes the label of the lesion, confidence of the CNN in the diagnosis, and a Boolean for recording the malignancy of the lesion. After this update, the client device is synchronized with the changes in Firestore, and receives the results of the diagnosis.

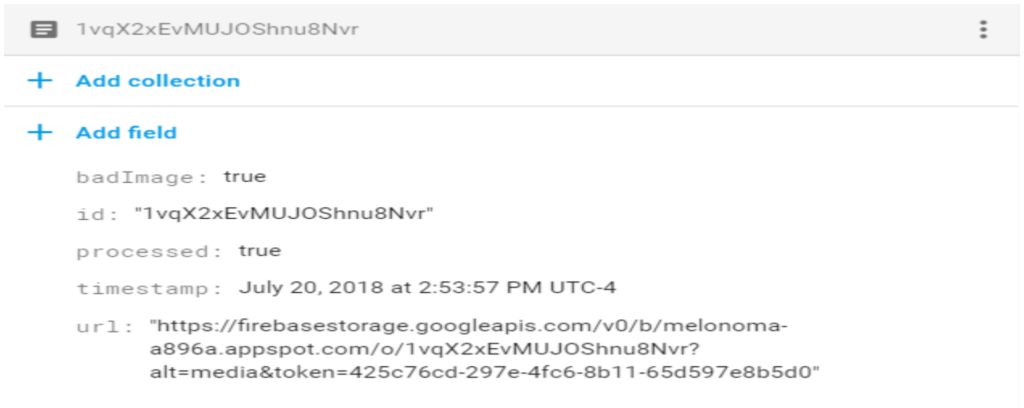


```
id: "3O9JbZo9r3ULOuqfTFms"
isMalignant: false
label: "Benign Keratosos"
processed: true
score: 89.65
timestamp: July 7, 2018 at 1:29:25 AM UTC-4
url: "https://firebasestorage.googleapis.com/v0/b/melonoma-
      a896a.appspot.com/o/3O9JbZo9r3ULOuqfTFms?
      alt=media&token=34a53918-51ad-4316-b863-7bcf838c2dc0"
```

Figure 4. A Firestore document with the updated by the CNN server with results from the CNN script.

3.4 CNN Script

The CNN script is a child process spawned by the CNN server that loads the trained CNN model to process the uploaded lesion images, and returns the output from the CNN. The script also features a preliminary CNN that was trained to detect images that do not contain lesions. Images that do not pass the preliminary CNN are rejected, and do not get processed by the diagnosis CNN. In that case, the CNN script returns an error message to the CNN server which then updates the Firestore document in the Cloud Database with the appropriate fields as shown in Figure 5.



```
badImage: true
id: "1vqX2xEvMUJOShnu8Nvr"
processed: true
timestamp: July 20, 2018 at 2:53:57 PM UTC-4
url: "https://firebasestorage.googleapis.com/v0/b/melonoma-
      a896a.appspot.com/o/1vqX2xEvMUJOShnu8Nvr?
      alt=media&token=425c76cd-297e-4fc6-8b11-65d597e8b5d0"
```

Figure 5. A Firestore document that reports that an image request was declined by the preliminary CNN.

The preliminary CNN is tiny in number of parameters compared to the diagnosis CNN, and a quick check by this CNN increases the overall quality of the system and may even increase the overall efficiency of the system by preventing some images making their way to the much larger, slower diagnosis CNN. The flow of the script is given in Figure 6.

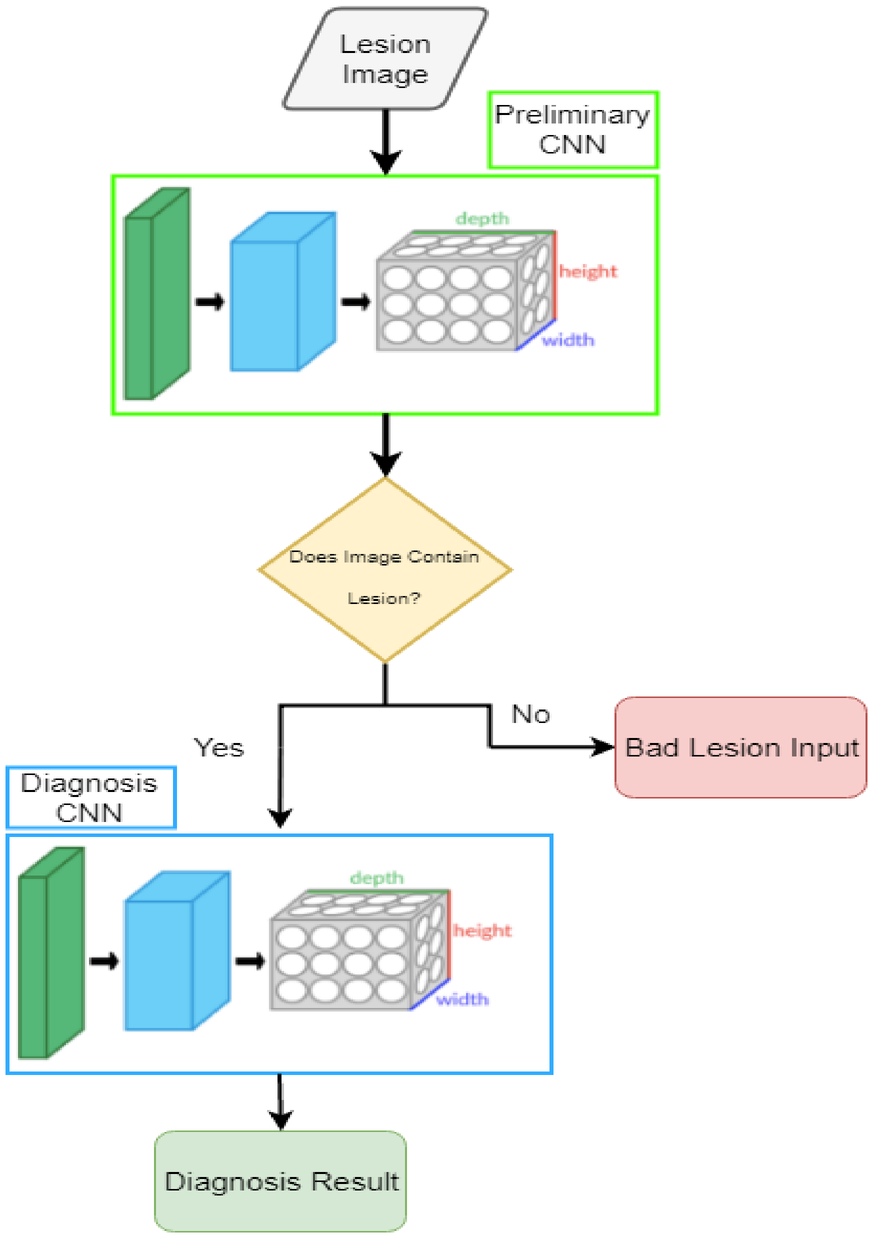


Figure 6. The preliminary CNN prevents images that do not contain lesions from getting further processed by the diagnosis model.

The script was written in Python, and for training and running the CNNs, the neural network library Keras was used [15]. The source code for the CNN script is given in Figure 7.

```python
import sys
import json

from keras.applications.resnet50 import preprocess_input
from keras.models import load_model
from keras.preprocessing import image
import numpy as np

# 1) Two trained models
model = load_model('models/model_resnet_best.hdf5')
isLesionModel = load_model('models/model_islesion.hdf5')

# 2) Labels CNN outputs
LABEL_STRS = ["AKIEC", "BCC", "BKL", "DF", "MEL", "NV", "VASC"]
LABEL_FULL = ["Actinic Keratosis", "Basal Cell Carcinoma",  "Benign Keratosos",
"Dermatofibroma", "Melanoma", "Nevus", "Vascular Lesion"]
ISMALIGNANT = [True, True, False, False, True, False, False]

# 3) Await IDs input from CNN server
for line in sys.stdin:

    # 3.1) Extract id from JSON string from server and load image with id
    id = json.loads(line)['id']
    PATH = "tmp/" + id
    img = np.array([preprocess_input(image.img_to_array(image.load_img(PATH)))])

    # 3.2) Run the image through the preliminary CNN to check if image contains lesion.
    pred = isLesionModel.predict([img])
    score = round(max(pred[0]) * 100.0,2) #score
    idx = pred.argmax(axis=-1)[0] #label index

    # 3.3) if image is good, run the diagnosis CNN, else report back with badImage label
    if idx == 0:
        result = {'badImage': True, 'id': id}
        print(json.dumps(result))
    else:
        pred = model.predict([img])
        score = round(max(pred[0]) * 100.0,2) #score
        idx = pred.argmax(axis=-1)[0] #label index
        result = {'score': score, 'isMalignant': ISMALIGNANT[idx], 'label': LABEL_FULL[idx],
                  'id': id}
        print(json.dumps(result))
```

Figure 7. Source code for CNN script.

In part 1 of the source code, the two models are loaded. The diagnosis model was trained on the ISIC 2018 dataset which contains images for several skin diseases including melanoma as can be observed in the array of labels in part 2 [12, 13]. The preliminary CNN model has two outputs for lesion and non-lesion images. The datasets used to the train the network were Caltech 101 for non-lesion and the ISIC dataset for lesion images [16].

The image IDs are passed to the standard input of the script in JSON form, and in part 3.1, the id is loaded from the JSON input, and the image corresponding to the id is decoded and preprocessed for both CNNs. Next, in part 3.2, the input image is processed by the preliminary CNN, and an output, either lesion or non-lesion, is acquired. In the following subsection, images labeled as lesion images by the preliminary CNN are processed by the diagnosis CNN, and the diagnosis results are printed out in JSON form back to the CNN server. Non-lesion images on the hand are reported back with an error field. The discussion on the architecture of the models and the training methods take place in the following chapter.

3.5 Alternative Architectures

There are several alternative architectures that were considered during the design stage of the system. Generally, there are two ways of incorporating deep neural networks in a mobile app: running the network locally on the native device or offloading the processing to servers. Running locally only became possible in recent years as training and running neural networks require huge amounts of computation power. In fact, one of the main reasons for the booming of deep learning is due to the recent growth in processing capabilities of processors, mainly graphics processing units (GPU) [17]. In addition to computation demands, the size of the model can put restraints on the device's memory. ResNet50 is the model that was used for the diagnosis CNN, and it's binary size is roughly 99 Megabytes (MBs) with 25,636,712 parameters, which makes it difficult to run on older mobile devices with constrained memory and computation capabilities [18, 19]. Another downside of running locally is that some application that are powered by neural networks may need frequent updates. Any updates to the model as a result of training as more and more medical data is gathered would need be downloaded by the client device in the form 99 MBs patches in the least, putting users with weak and slow Internet connection out of reach.

For server architectures, a seemingly more simple, monolith architecture was also considered where client requests, database, and the CNN are handled by a single server instance. The cloud architecture, on the contrary, establishes connection between specialized programs written in different programming environments that solve different tasks and does not force the entire system to operate in one environment, as it is the case in a monolith server. It follows a software philosophy where a large system is composed of a number of components or programs that performs a simple task and pass on data to another component to complete another task. As a result, although monolith architecture has fewer independent components than the cloud architecture, the implementation may be more complex and inefficient. An example of specialization is that many neural network libraries have been written in Python which makes Python a popular, specialized language of choice for implementing and training neural networks. Node.js is also another popular programming environment used for creating server which is featured in the cloud architecture. The cloud architecture takes advantage of segregating tasks to specialized components whereas a monolith system offers very little flexibility.

The cloud architecture can also grow more naturally and efficiently. To respond to high user demand, multiple CNN server nodes can be spawned in different geographical locations where each CNN server is also capable of spawning multiple CNN scripts to maximize throughput with little change to the source code. Also, since data storage and processing are separated in the form of the cloud database and CNN server, users attempting to access results from the cloud database are not affected by any amount of request traffic the CNN server may be under. The cloud architecture allows the client application to be light and simple which opens up the range of devices and environments the system can operate in, so this architecture was opted to be implemented.

4. Diagnosis Convolutional Neural Network

4.1 Dataset

The dataset containing lesion image samples used to train the preliminary and diagnosis CNNs come from the "ISIC 2018: Skin Lesion Analysis Towards Melanoma Detection" grand challenge datasets [12, 13], which contains a total of 10,015 samples of skin diseases shown in Table 1. The dataset is not balanced as there are significantly more nevus (benign, non-harmful) samples than any other samples, combined. Also, dermatofibroma and vascular lesion are severely underrepresented. An unbalanced dataset can make it difficult to train and assess the accuracy of a network so the dataset was rebalanced before training.

Table 1. Skin diseases and sample counts in the ISIC 2018 dataset (10,015 samples).

Disease	Actinic Keratosis	Basal Cell Carcinoma	Benign Keratosis	Dermato-fibroma	Melanoma	Nevus	Vascular Lesion
Count	327	514	1099	115	1113	6705	142
Percentage of Total	3.3%	5.1%	11.1%	1.1%	11%	67%	1.4%

4.2 Training and Results

The dataset needs to be rebalanced by under-sampling overrepresented categories and split to create training and testing sets. For the split, 80% of the dataset was used for training and 20% for testing. Each split set was rebalanced by under-sampling so that only 20% of the set contained nevus samples. The balance of the training and training sets can be observed in Table 2.

Because training from scratch isn't plausible, transfer learning on the ResNet50 architecture pre-trained for the 2014 ImageNet Large Scale Visual Recognition Challenge was used for training [11, 19]. The final layer of a ResNet network is a simple softmax dense layer which connects to a flattened average-pooled 2048 dimension layer connected to the final convolutional layer. The final softmax layer was replaced by two densely connected layers with 500 neurons in each and a new, 7 category softmax layer. During training, the entire network besides the newly added layers were kept frozen. 25% dropout between every connection and batch normalization before every activation was added to the new layers for regularization [20, 21]. Stochastic gradient descent (SGD) with a learning rate of 0.01 was used to train the CNN. The code sample, shown in Figure 8, is the implementation of the new dense layers in Keras.

Table2. Nevus samples were under-sampled to balance the training and testing sets.

Train Set

Disease	Actinic Keratosis	Basal Cell Carcinoma	Benign Keratosis	Dermatofibroma	Melanoma	Nevus	Vascular Lesion
Count	265	408	893	93	874	662	112
Percentage of Total	8%	12.3%	27%	2.8%	26.4%	20%	3.4%

Test Set

Disease	Actinic Keratosis	Basal Cell Carcinoma	Benign Keratosis	Dermatofibroma	Melanoma	Nevus	Vascular Lesion
Count	62	106	206	22	239	166	30
Percentage of Total	7.5%	12.8%	24.8%	2.7%	28.8%	20%	3.6%

The accuracy of the neural network after training on the test set is 77.4%. The rebalance of the dataset resulted in high accuracies for non-nevus categories, as shown in Figures 9 and 10, and having more samples of melanoma than nevus also helped improved sensitivity of the network (lowered false negatives) when diagnosing melanoma vs nevus lesions which is especially vital for diagnosis systems. Although human level accuracy on this dataset has not been determined, it is safe to assume that as per [4] the accuracy of the diagnosis network, at least on melanoma detection, is on or very close to human level performance.

Further experiments like unfreezing final few layers to continue backpropagation over ResNet resulted in overfitting and lowered test accuracy. Despite a tiny amount of training data, several CNNs ranging from few layers to up to 18 layers were experimented with, and the best accuracy achieved was 66%. In the end, the network trained with transfer learning was chosen as the diagnosis CNN.

```
epochs = 350
drop = 0.25
dns = 500
batch_size = 16
rg = 1e-3

mc_top = ModelCheckpoint('weights/model_bottleneck_test.hdf5', monitor='val_acc', verbose=0,
save_best_only=True, save_weights_only=False, mode='auto', period=1)

model = Sequential()

model.add(Flatten(input_shape=train_data.shape[1:]))
model.add(Dropout(drop))

model.add(Dense(dns, kernel_initializer='he_normal', kernel_regularizer=l2(rg)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(drop))

model.add(Dense(dns, kernel_initializer='he_normal', kernel_regularizer=l2(rg)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(drop))

model.add(Dense(num_classes, kernel_initializer="he_normal", activation='softmax'))

opt = SGD(lr=0.01)
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(train_data, train_labels,
          epochs=epochs,
          verbose=2,
          batch_size=batch_size,
          shuffle=True,
          validation_data=(test_data, test_labels),
          callbacks=[mc_top])
```

Figure 8. The implementation of the new dense layers in Keras.

5. Preliminary Convolutional Neural Network

The purpose of the preliminary CNN is to filter bad requests from the user. This way, the diagnosis system won't attempt to classify images that are not lesion images or do not meet the quality of the images in the training set and send back garbage results as a result. To train the CNN, any dataset containing categories of common office or household objects that a user might take a picture of to test the system, or any category that is exclusively not medical and skin lesion related to train against the lesion images could work. As counter to the ISIC 2018 dataset, the Caltech 101 dataset along with ISIC 2018 was used to train the CNN. Samples in the two categories in the training and testing sets were randomly chosen and contain roughly 8000 and 2000 sample images, respectively.

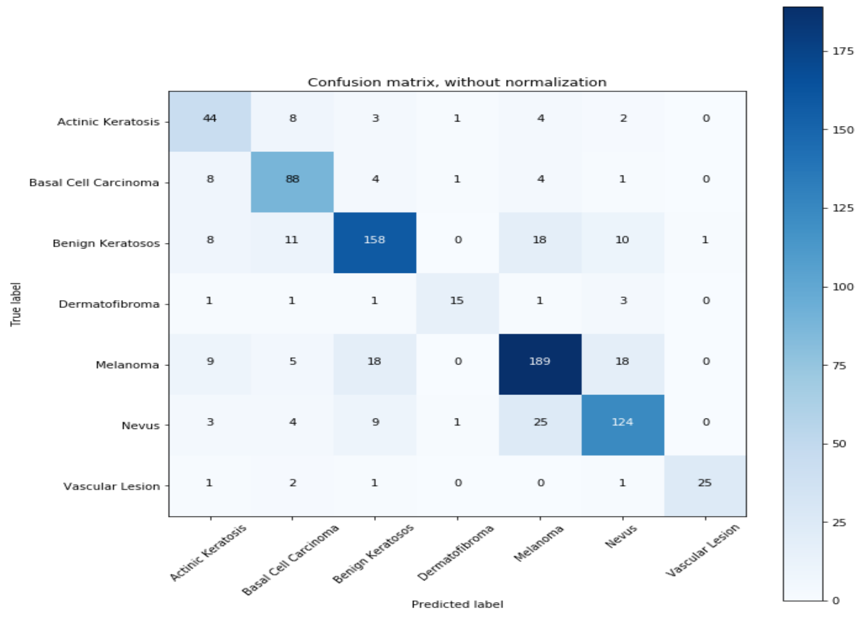


Figure 9. Confusion matrix of the test set.

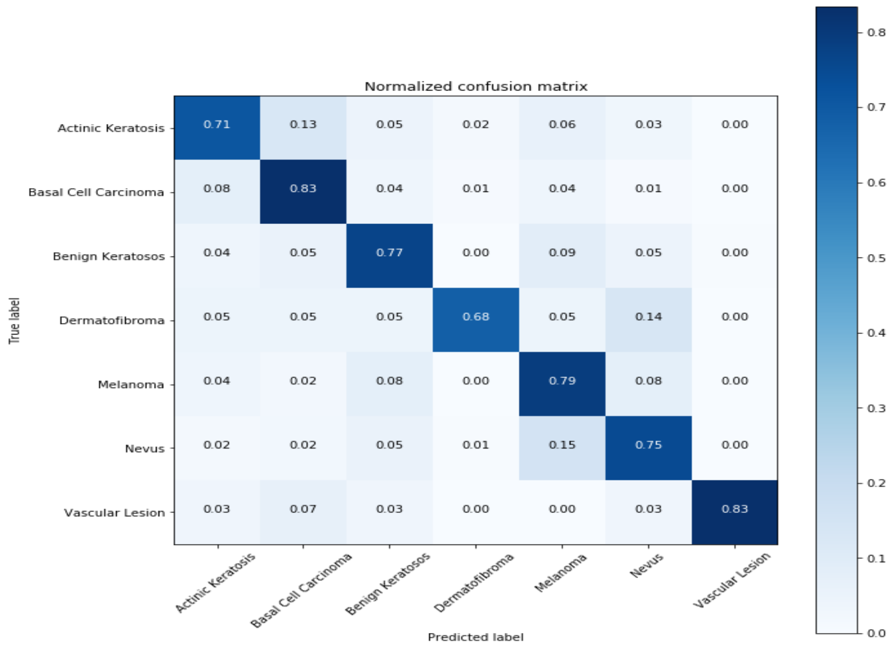


Figure 10. Normalized confusion matrix of the test set.

The architecture of the CNN is relatively small, containing roughly 73,000 parameters. After the input layer, there are 6 convolutional layers with kernel size of 3x3 and strides of 2x2. After the final convolution layer, an average pool is computed and the layer is flattened which is connected to the final double output softmax layer. Training was done with SGD and learning rate of 0.01. The architecture can be observed in detail in the Keras code sample shown in Figure 11.

```
ROW_AXIS = 1
COL_AXIS = 2
CHANNEL_AXIS = 3

def bn_rel (conv):
    norm = BatchNormalization(axis=CHANNEL_AXIS)(conv)
    return Activation("relu")(norm)

def conv2(prev, filters=64, kernel_size=(3,3), strides=(2,2), padding='same'):
        return Conv2D(filters=filters,
                      kernel_size=kernel_size,
                      strides=strides,
                      padding=padding,
                      kernel_initializer='he_normal',
                      kernel_regularizer=l2(1.e-4))(prev)
dim = 224
num_classes = 7

# input block
input = Input(shape=(dim,dim, 3))

filters = 16

conv = conv2(input,filters=filters)
actv = bn_rel(conv)
conv = conv2(actv, filters=filters)
actv = bn_rel(conv)

filters = 32

conv = conv2(actv, filters=filters)
actv = bn_rel(conv)
conv = conv2(actv, filters=filters)
actv = bn_rel(conv)

filters = 64

conv = conv2(actv, filters=filters)
actv = bn_rel(conv)
conv = conv2(actv, filters=filters)
actv = bn_rel(conv)

block_shape = K.int_shape(actv)
pool = AveragePooling2D(pool_size=(block_shape[ROW_AXIS], block_shape[COL_AXIS]),
strides=(1, 1))(actv)
```

```
flat = Flatten()(pool)

dnse = Dense(2, kernel_initializer="he_normal", activation="softmax")(flat)

model = Model(inputs=input, outputs=dnse)

opt = SGD(lr=0.01)
model.compile(optimizer=opt, loss='binary_crossentropy', metrics=['accuracy'])
model.summary()
```

Figure 11. Keras code sample of the CN architecture.

Since the images in both categories are very different from each other, the accuracy of the network is very high. After only three epochs, the accuracy on the test set was 99.4% so training was stopped early, and the preliminary CNN was fitted with this network.

6. Mobile Phone Application

We developed an application for both Android and iOS platforms in order to complete the entire cloud-based skin lesion diagnosis system. The application connects to Firebase at startup to upload images and receive results from the diagnosis pipeline. The application is a hybrid mobile app built using Ionic SDK [24]. With Ionic, developers can build mobile and web applications using Web technologies like HTML, CSS, and Javascript with built-in native features that the device offers like camera and geolocation. Using the same source code, the application can be compiled to run on multiple platforms devices which includes Android, iOS, and web browsers.

The application connects to the device's camera and displays the camera stream. The user is guided with a red circle to help place the skin lesion centered horizontally and vertically in the visual field of the camera (Figure 12). This is an important feature because the lesion images in the dataset are centered on the lesion and replicating this property in the user requests may increase the accuracy of the diagnosis. Once an image is taken by the user, the users have the option to either retake the image or upload the image to Firebase. Before uploading, the image is cropped and resized to match the exact dimensions of the diagnosis and preliminary CNNs. Once uploaded, the user is taken to the History page, which displays results from the diagnosis CNN, as illustrated in Figure 13.

During the diagnosis process, the uploaded request is marked as 'processing'. Once a result is acquired, the app is updated with a diagnosis and confidence score of the diagnosis CNN. If an image does not pass the preliminary CNN, the user is notified with a warning text.

The diagnosis system being cloud-based, complexity of the application is minimal. The main requirements of the application are connection to the camera and Firebase and client-side image resizing which can be done with few lines of Javascript. The minimalist nature of the front-end application also allowed for a very quick design and implementation.

7. Conclusion and Future Work

Skin cancer is a serious medical problem. Early diagnosis for type of cancer like melanoma is vital for survival. For this thesis, a skin lesion diagnosis system was built to help with diagnosis. The system is cloud-based and uses Firebase. The diagnosis process is offloaded to servers which allows developers to build front-ends for a large range of mobile devices that are lightweight and minimalistic in code complexity. The diagnosis is handled by a two-stage CNN pipeline where a preliminary CNN does quality check on user requests, and a diagnosis CNN, whose accuracy is near dermatologist level, outputs probabilities over seven different lesion categories corresponding to the categories in the ISIC 2018 dataset used for training. For training, transfer learning was applied to a ResNet50 network trained on the ImageNet competition dataset.

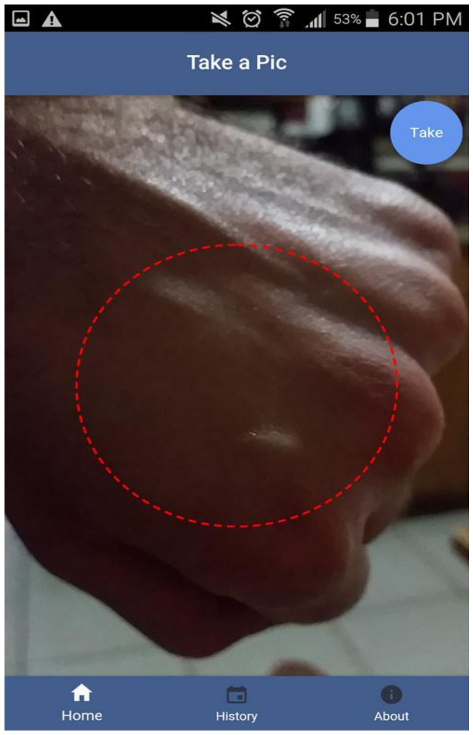
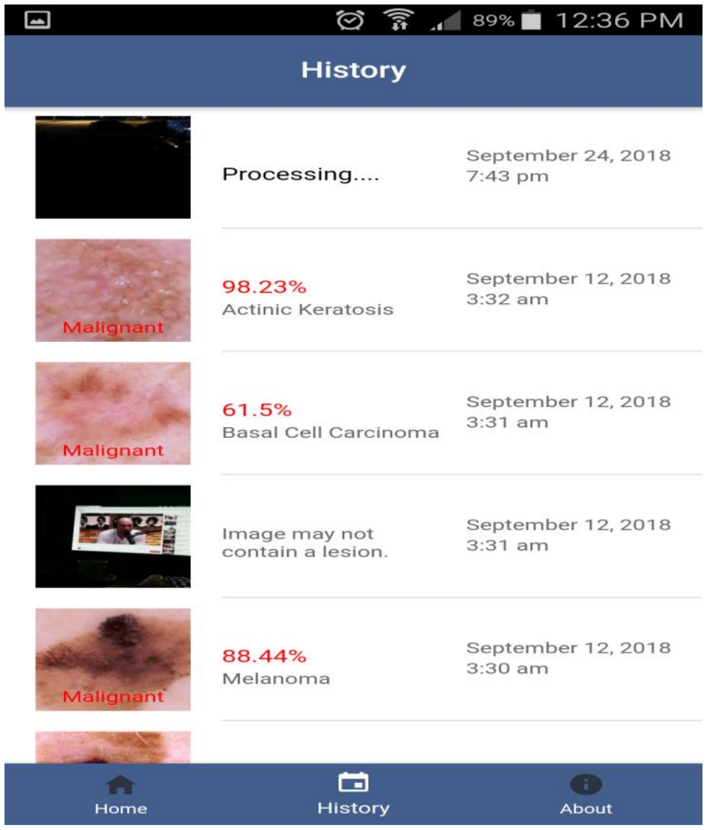


Figure 12. The application utilizes the native camera on the mobile phone to takes pictures of lesions.



Processing.... September 24, 2018 7:43 pm

98.23% Actinic Keratosis September 12, 2018 3:32 am

61.5% Basal Cell Carcinoma September 12, 2018 3:31 am

Image may not contain a lesion. September 12, 2018 3:31 am

88.44% Melanoma September 12, 2018 3:30 am

Figure 13. History of results page displays diagnosis results.

For future work, to improve accuracy of the diagnosis network, more data would need to be gathered. The diagnosis CNN was trained using transfer learning with roughly 3,000 images and that is not nearly enough to allow experimentation with different CNN architectures and training paradigms. One experiment, put under future work, is combining the diagnosis ResNet with a smaller network that is trained from scratch in multi-stream fashion where the outputs of both networks are connected to a combined final layer. This way, a unique version of ensemble of networks may result in higher accuracy and better generalization.

Acknowledgments

The authors gratefully acknowledge funding from NSF award No. 1464537, Industry/University Cooperative Research Center, Phase II under NSF 13-542. We are also thankful to Farris Foundation who also provided funds for this project.

References

1. Rogers HW, Weinstock MA, Feldman SR, Coldiron BM. Incidence estimate of nonmelanoma skin cancer (keratinocyte carcinomas) in the US population, 2012. JAMA Dermatol 2015; 151(10):1081-1086.
2. Cancer Facts and Figures 2018. American Cancer Society. https://www.cancer.org/content/dam/cancer-org/research/cancer-facts-and-statistics/annual-cancer-facts-and-figures/2018/cancer-facts-and-figures-2018.pdf. Accessed May 3, 2018.
3. Stern, RS. Prevalence of a history of skin cancer in 2007: results of an incidence-based model. Arch Dermatol 2010; 146(3):279-282.
4. Guy GP, Machlin SR, Ekwueme DU, Yabroff KR. Prevalence and costs of skin cancer treatment in the U.S., 2002-2006 and 2007-2011. Am J Prev Med 2014. 104(4):e69-e74. doi:dx.doi.org/10.1016/j.amepre.2014.08.036.
5. R. Siegel and A. Miller, K.D.and Jemal, "Cancer statistics, 2016.," CA: A Cancer Journal for Clinicians, vol. 66, pp. 7–30, 2016.
6. K. H, H. Pehamberger, K. Wolf, and M. Binder, "Diagnostic of dermoscopy," The Lancet Oncology, vol. 3, . 159–165, 2002.
7. Yu, L., Chen, H., Dou, Q., Qin, J., Heng, P.A., 2017a. Automated melanoma recognition in dermoscopy images via very deep residual networks. IEEE Trans. Med.Imaging 36 (4), 994–1004. doi:10.1109/TMI.2016.2642839.
8. Esteva, A., Kuprel, B., Novoa, R.A., Ko, J., Swetter, S.M., Blau, H.M., Thrun, S.: Dermatologist-level classification of skin cancer with deep neural networks. Nature 542(7639), 115–118 (2017)
9. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. & Wojna, Z. Rethinking the inception architecture for computer vision. Preprint at https://arxiv.org/abs/1512.00567 (2015).
10. Russakovsky, O. et al. Imagenet large scale visual recognition challenge. Int. J. Comput. Vis. 115, 211–252 (2015).
11. Pan, S. J. & Yang, Q. A survey on transfer learning. IEEE Trans. Knowl. Data Eng. 22, 1345–1359 (2010).
12. Tschandl P., Rosendahl C. & Kittler H. The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. Sci. Data 5, 180161 doi.10.1038/sdata.2018.161 (2018).
13. Noel C. F. Codella, David Gutman, M. Emre Celebi, Brian Helba, Michael A. Marchetti, Stephen W. Dusza, Aadi Kalloo, Konstantinos Liopyris, Nabin Mishra, Harald Kittler, Allan Halpern: "Skin Lesion Analysis Toward Melanoma Detection: A Challenge at the 2017 International Symposium on Biomedical Imaging (ISBI), Hosted by the International Skin Imaging Collaboration (ISIC)", 2017; arXiv:1710.05006.
14. Cloud Firestore. (n.d.). Retrieved August 29, 2018, from https://firebase.google.com/docs/firestore/
15. Chollet, Fran and others, Keras (2018), GitHub repository, https://github.com/keras-team/keras
16. L. Fei-Fei, R. Fergus and P. Perona. Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. IEEE. CVPR 2004, Workshop on Generative-Model Based Vision. 2004.
17. L. Deng, "A tutorial survey of architectures, algorithms, and applications for deep learning," APSIPA Transactions on Signal and Information Processing, vol. 3, p. e2, 2014.
18. [8] "Documentation for individual models." https://keras.io/applications, 2018. Accessed: 2018-08-01.

19. He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
20. Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." The Journal of Machine Learning Research 15.1 (2014): 1929-1958.
21. Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv preprint arXiv:1502.03167 (2015).
22. Drifty, Inc (2016). Ionic. https://ionicframework.com.