Skin Lesions Classification with Deep Convolutional Neural Network

1. Introduction

The system can classify 5 diseases with accuracy rate 80%, which still needs to be enhanced. Skin diseases are more common than other diseases. Skin diseases may be caused by fungal infection, bacteria, allergy, or viruses, etc. The advancement of lasers and Photonics based medical technology has made it possible to diagnose the skin diseases much more quickly and accurately. But the cost of such diagnosis is still limited and very expensive. So, image processing techniques help to build automated screening system for dermatology at an initial stage. The extraction of features plays a key role in helping to classify skin diseases. Computer vision has a role in the detection of skin diseases in a variety of techniques. We proposed an image processing-based method to detect skin diseases. This method takes the digital image of disease effect skin area, then use image analysis to identify the type of disease. Our proposed approach is simple, fast and does not require expensive equipment other than a camera and a computer. The approach works on the inputs of a color image. Then resize the of the image to extract features using pre-trained convolutional neural network. After that classifying the images.

2. Description of The Dataset

We collected our dataset from Kaggle, the dataset has 11K different images of 5 skin diseases (Eczema, Melanoma, Atopic Dermatitis, Basal Cell Carcinoma, Psoriasis pictures Lichen Planus and related diseases)

3. Methodology

3.1. Preprocessing

1- Unifying the image size will get the same number of features from all images. Moreover, resizing the image reduces processing time and thus increases system performance the images was resized into 224x224 pixels.

2- We used ImageDataGenerator to carry out data augmentation, we rescaled our images with rescaling factor 1/255, horizontal and vertical flip, shear-range and zoom-range.

3- We splitted our dataset into 80% training and 20% testing.

3.2. Models

1- Convolutional Neural Network (CNN)

We constructed a CNN model of 20 layers which are 7 convolutional layers where a nonlinear ReLU layer is stacked after each convolutional layer, 4 pooling Layers, 5 Dropout layers and one dense layer as output layer with activation function softmax which is suitable for multiclass classification.

Categorical crossentropy loss function is our choice for all the models as it is used for multi-class classification tasks.

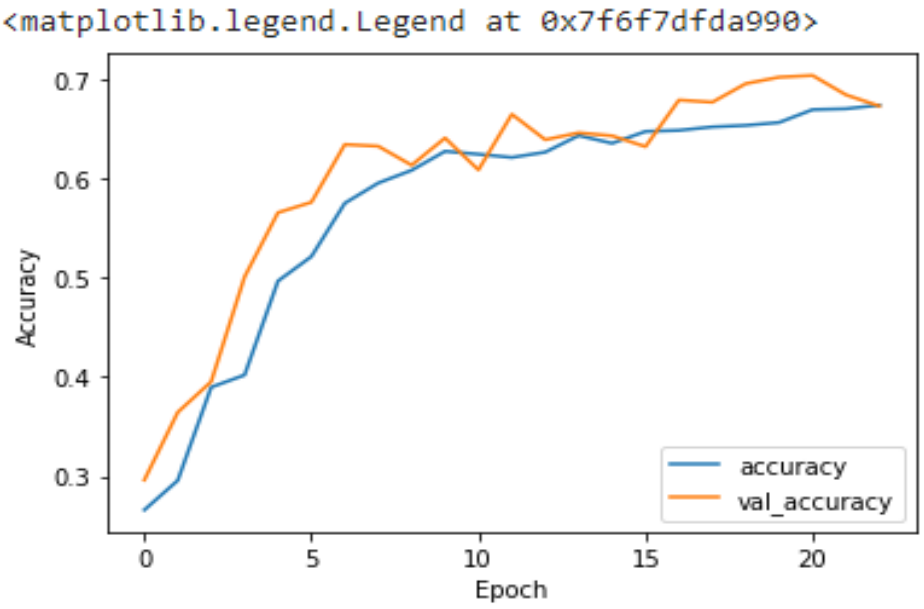
We choose adamax activation function for this model and the accuracy reached up to 72%

```
model = keras.Sequential()
    # The first two layers with 32 filters of window size 3x3
model.add(Conv2D(32, (3, 3), padding='same', activation='relu', input_shape=(224,244,3)))
model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(AveragePooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), padding='valid', activation='relu'))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(AveragePooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), padding='valid', activation='relu'))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(AveragePooling2D(pool_size=(2, 2)))
model.add(Dropout(0.3))

model.add(Conv2D(64, (3, 3), padding='same', activation='tanh'))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(AveragePooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))


model.add(Flatten())
model.add(Dense(512, activation='relu'))

model.add(Dropout(0.4))
model.add(Dense(5, activation='softmax'))
```

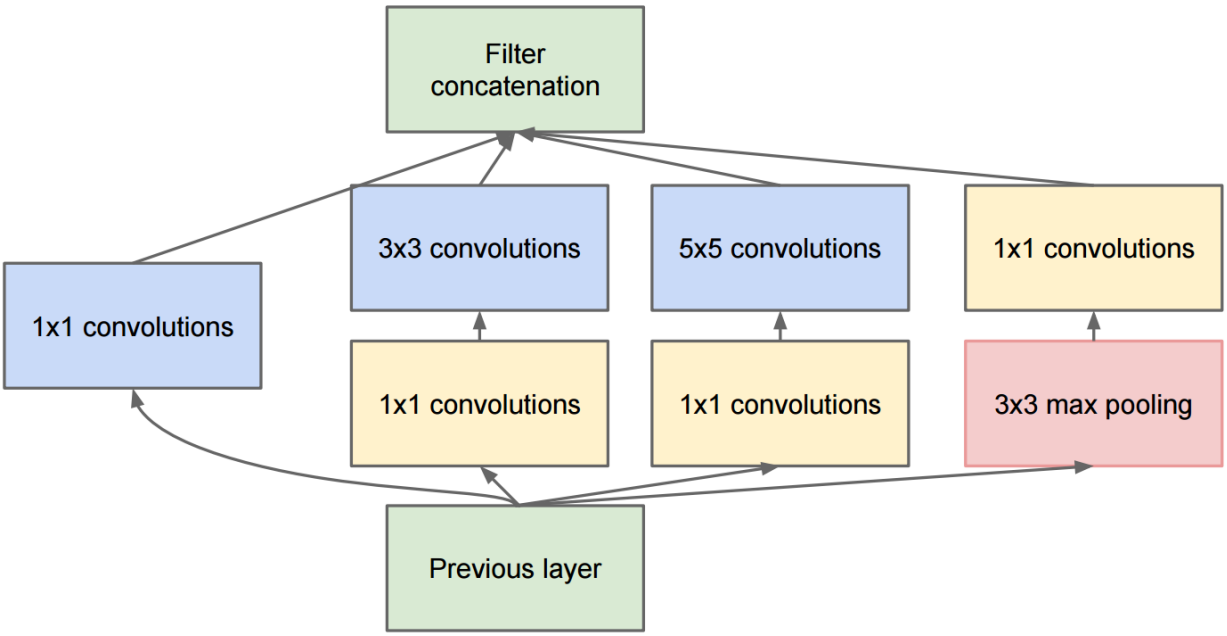
```
model.compile(loss=keras.losses.categorical_crossentropy, optimizer="adamax", metrics=['accuracy'])
```

```
esults=model.fit(
    train_data,
    epochs=10,
    validation_data=test_data,steps_per_epoch=len(train_data),
    validation_steps=len(test_data),
    callbacks=tf.keras.callbacks.EarlyStopping(patience=3,restore_best_weights=True)
  )
```

```
Epoch 1/10
252/252 [==============================] - 217s 858ms/step - loss: 0.6594 - accuracy: 0.7135 - val_loss: 0.6447 - val_accuracy: 0.7299
Epoch 2/10
252/252 [==============================] - 215s 852ms/step - loss: 0.6378 - accuracy: 0.7158 - val_loss: 0.6108 - val_accuracy: 0.7385
Epoch 3/10
252/252 [==============================] - 224s 891ms/step - loss: 0.6575 - accuracy: 0.7028 - val_loss: 0.6084 - val_accuracy: 0.7290
Epoch 4/10
252/252 [==============================] - 220s 874ms/step - loss: 0.6394 - accuracy: 0.7176 - val_loss: 0.6767 - val_accuracy: 0.7100
Epoch 5/10
252/252 [==============================] - 221s 878ms/step - loss: 0.6409 - accuracy: 0.7151 - val_loss: 0.6098 - val_accuracy: 0.7264
Epoch 6/10
252/252 [==============================] - 223s 884ms/step - loss: 0.6461 - accuracy: 0.7152 - val_loss: 0.6148 - val_accuracy: 0.7212
```



2- Inception

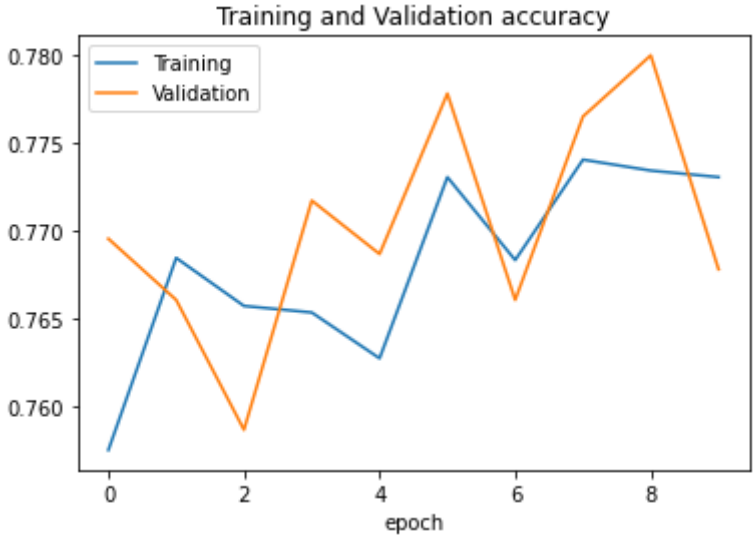


1- We Flatten the output of our base model to 1 dimension
2- Add a fully connected layer with 1,024 hidden units and ReLU activation
3- Add a final Fully Connected Softmax Layer
4- Adagrad Loss Function with 0.001 learning rate

We reached accuracy up to 77% and Loss 52%

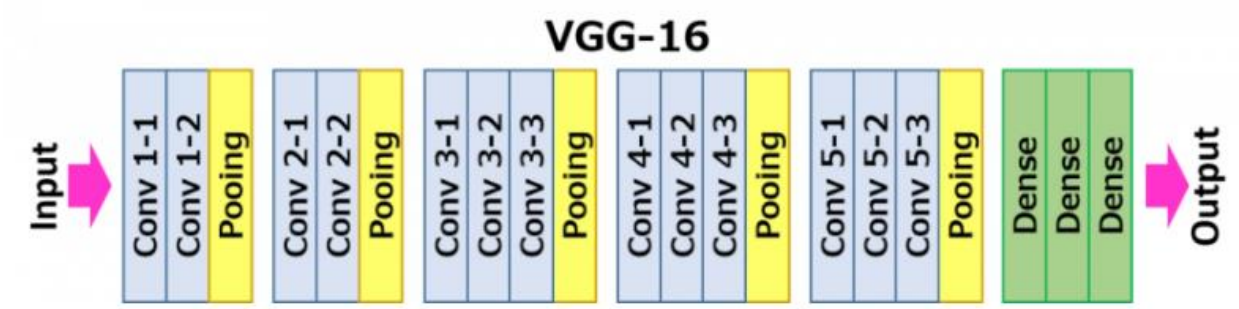
```
# Taking the output of the last convolution block
x = base_model.output

# Adding a Global Average Pooling layer
x = GlobalAveragePooling2D()(x)

# Adding a fully connected layer having 1024 neurons
x = Dense(1024, activation='relu')(x)

# Adding a fully connected layer having 2 neurons which will
# give the probability of image having either dog or cat
predictions = Dense(5, activation='softmax')(x)
# Model to be trained
model = Model(inputs=base_model.input, outputs=predictions)

# Training only top layers i.e. the layers which we have added in the end
for layer in base_model.layers:
    layer.trainable = False

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=tf.keras.optimizers.Adagrad(learning_rate=0.001), metrics=['accuracy'])
```

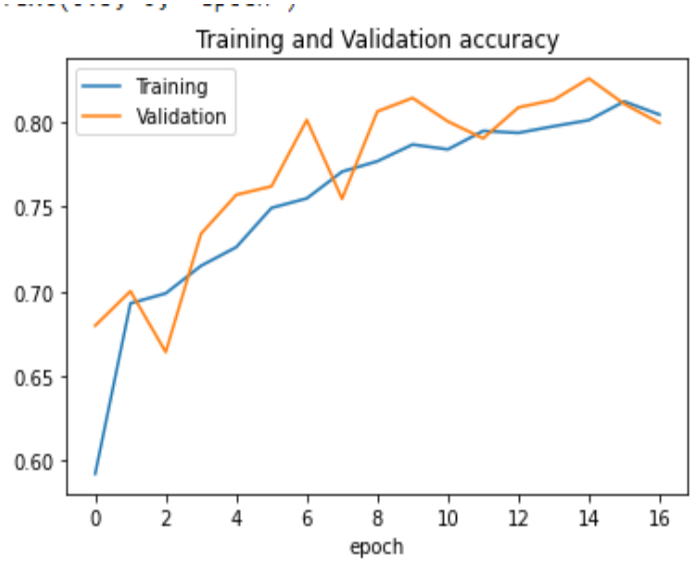
```
Epoch 7/10
252/252 [==============================] - 178s 704ms/step - loss: 0.5354 - accuracy: 0.7684 - val_loss: 0.5487 - val_accuracy: 0.7661
Epoch 8/10
252/252 [==============================] - 178s 708ms/step - loss: 0.5312 - accuracy: 0.7741 - val_loss: 0.5190 - val_accuracy: 0.7765
Epoch 9/10
252/252 [==============================] - 179s 711ms/step - loss: 0.5239 - accuracy: 0.7734 - val_loss: 0.5180 - val_accuracy: 0.7800
Epoch 10/10
252/252 [==============================] - 179s 712ms/step - loss: 0.5331 - accuracy: 0.7731 - val_loss: 0.5274 - val_accuracy: 0.7678
```

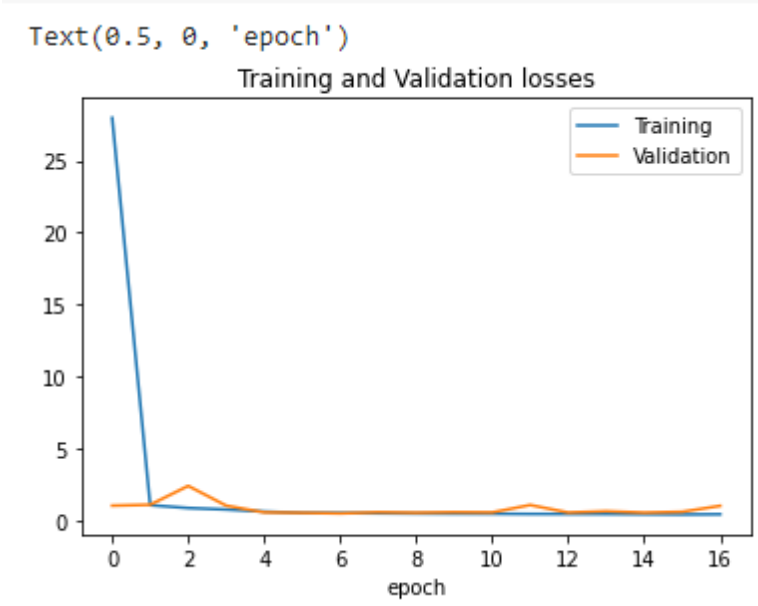


3- Very Deep Convolutional Networks for Large-Scale Image Recognition (VGG-16)

1- We used the VGG16 with 19 layers and added two dense layers with Relu activation function
2- RMSprop loss function with learning Rate 0.001, the accuracy of this model reached 80%







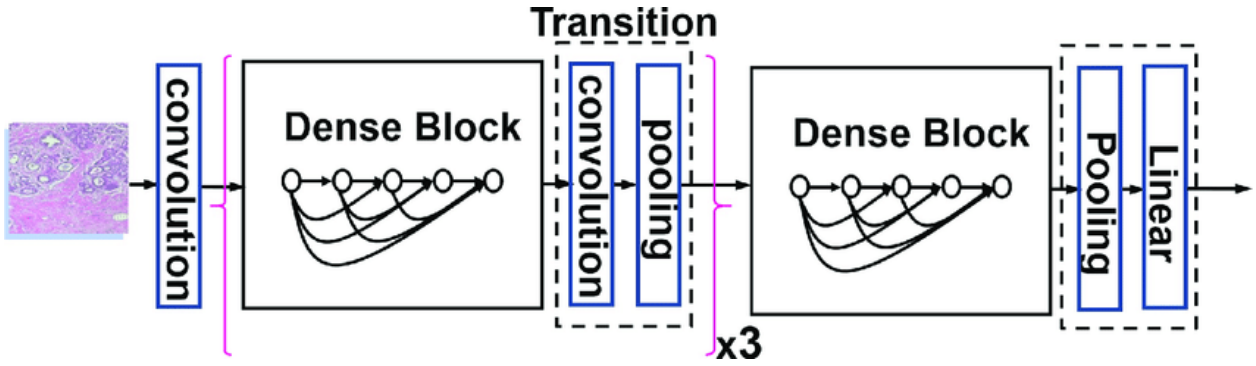

```
Epoch 14/30
125/125 [==============================] - 207s 2s/step - loss: 0.5074 - Accuracy: 0.7975 - val_loss: 0.6978 - val_Accuracy: 0.8131
Epoch 15/30
125/125 [==============================] - 207s 2s/step - loss: 0.4916 - Accuracy: 0.8012 - val_loss: 0.6032 - val_Accuracy: 0.8258
Epoch 16/30
125/125 [==============================] - 206s 2s/step - loss: 0.4763 - Accuracy: 0.8122 - val_loss: 0.6591 - val_Accuracy: 0.8107
Epoch 17/30
125/125 [==============================] - 216s 2s/step - loss: 0.4907 - Accuracy: 0.8044 - val_loss: 1.0749 - val_Accuracy: 0.7995
```

4- DenseNet-121

DenseNet-121 has the following layers:

- 1 7x7 Convolution
- 58 3x3 Convolution
- 61 1x1 Convolution
- 4 AvgPool
- 1 Fully Connected Layer

In a traditional feed-forward Convolutional Neural Network (CNN), each convolutional layer except the first one (which takes in the input), receives the output of the previous convolutional layer and produces an output feature map that is then passed on to the next convolutional layer. Therefore, for 'L' layers, there are 'L' direct connections; one between each layer and the next layer.



We added

1- **2 Dense layers with Relu Activation Function**
2- **3 Dropout layer equals 0.2**
3- **Adamax Loss Function**

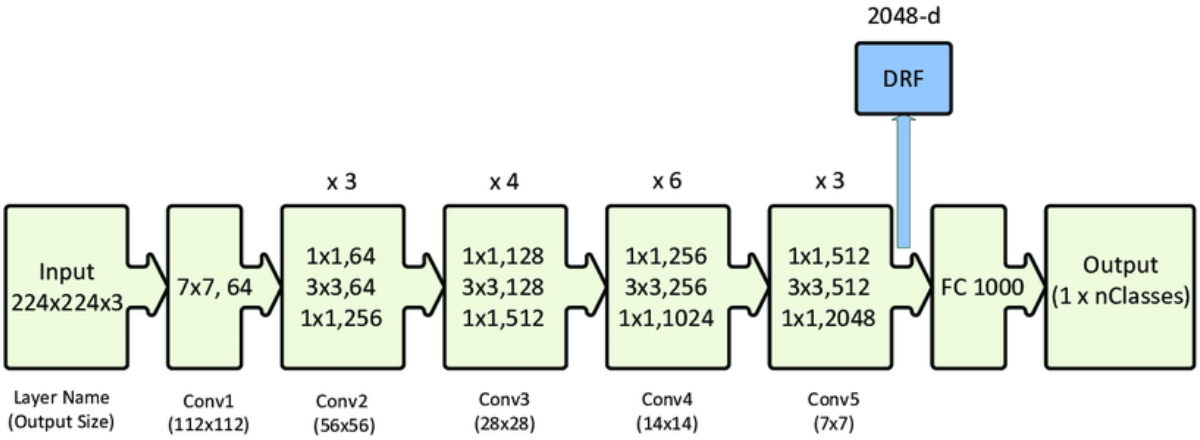
```
base_model = tf.keras.applications.DenseNet121( include_top = False,
                                                 weights = 'imagenet', input_shape=(224,224,3))
for layer in base_model.layers:
  layer.trainable = False
x = layers.Flatten()(base_model.output)
x = layers.Dropout(0.2)(x)
x = Dense(4096, activation='relu')(x)
x = layers.Dropout(0.2)(x)
x = Dense(4096, activation='relu')(x)
x = layers.Dropout(0.2)(x)
predictions = Dense(5, activation='softmax')(x)
# Model to be trained
model = Model(inputs=base_model.input, outputs=predictions)
```

Achieved up to 81% Accuracy and 41% Loss

```
Epoch 7/10
288/288 [==============================] - 211s 733ms/step - loss: 0.4171 - accuracy: 0.8160 - val_loss: 0.4324 - val_accuracy: 0.8059
Epoch 8/10
288/288 [==============================] - 211s 733ms/step - loss: 0.4408 - accuracy: 0.8070 - val_loss: 0.4123 - val_accuracy: 0.8185
Epoch 9/10
288/288 [==============================] - 211s 731ms/step - loss: 0.4328 - accuracy: 0.8031 - val_loss: 0.4007 - val_accuracy: 0.8172
Epoch 10/10
288/288 [==============================] - 211s 733ms/step - loss: 0.4270 - accuracy: 0.8086 - val_loss: 0.4191 - val_accuracy: 0.8094
```

5- ResNet-50 V2

ResNet50 is a variant of ResNet model which has 48 Convolution layers along with 1 MaxPool and 1 Average Pool layer.



We added

1- **2 Dense layers with Relu Activation Function**
2- **3 Dropout layer equals 0.2**
3- **Adamax Loss Function with 0.001 Learning Rate**

```
base_model = tf.keras.applications.DenseNet121( include_top = False,
                                                 weights = 'imagenet', input_shape=(224,224,3))
for layer in base_model.layers:
  layer.trainable = False
x = layers.Flatten()(base_model.output)
x = layers.Dropout(0.2)(x)
x = Dense(4096, activation='relu')(x)
x = layers.Dropout(0.2)(x)
x = Dense(4096, activation='relu')(x)
x = layers.Dropout(0.2)(x)
predictions = Dense(5, activation='softmax')(x)
# Model to be trained
model = Model(inputs=base_model.input, outputs=predictions)
```

Model	Validation	Test	Layers
CNN	71%	72%	20
Inception-V3	77%	76%	48
VGG16	80%	79%	22
DenseNet-121	81%	80%	121
ResNet-50 V2	82%	82%	83

Dataset Link
https://www.kaggle.com/datasets/ismailpromus/skin-diseases-image-dataset?resource=download

Team:
Rehab Hamed Ahmed
Email: rehapahmed9@gmail.com
Mobile: 01146011247

Reem Badawy Mohamed
Email: reem25badawy@gmail.com
Mobile:01008460693