Capstone Project Presentation

VGG19 Skin Cancer Detection

Group Members :

1. Douglas Koo Chen Soon
2. Zakiyyah binti Mohd Zahid
3. Shamsuri Azmi
4. Shafiq Aiman

Problem Statement

- What is actually considered that skin cancer cells is cancerous (danger zone) malignant or benign?
- How to achieve better and cheaper prediction than a dermatologist?

Objective

- To adapt VGG19 to suit the cancer datasets
- To evaluate the accuracy of the prediction

Solution

- Skin Cancer detection using VGG19 Image Classification

Image Classification

- VGG19
- Dataset: Kaggle Skin Cancer (Malignant vs Benign)

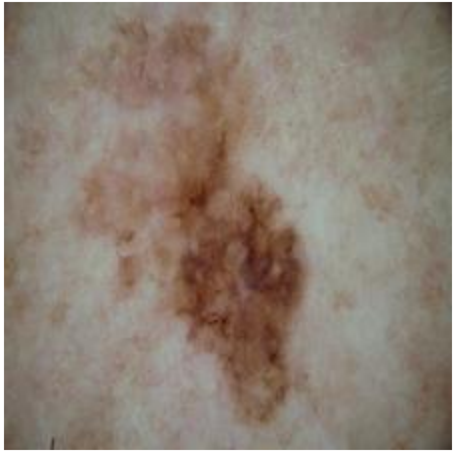
Image Data

VGG 19
Network

Malignant
OR
Benign

How skin cancer image appear visually

Benign

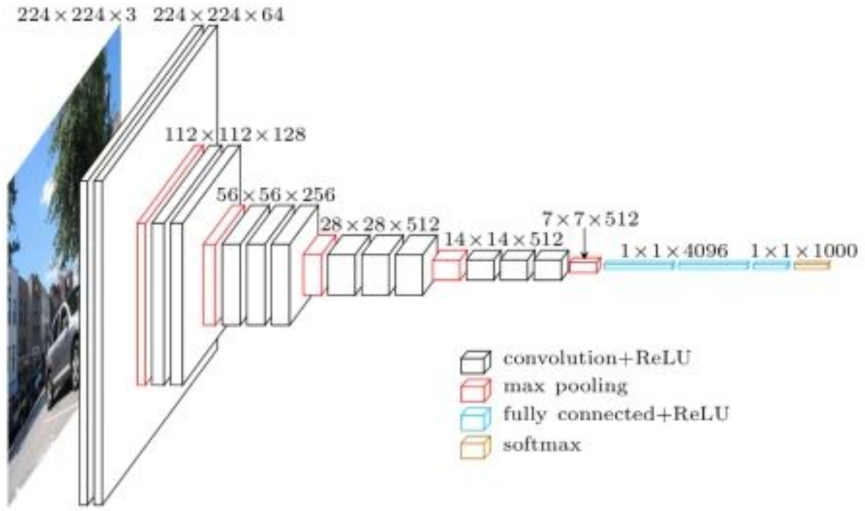


Malignant

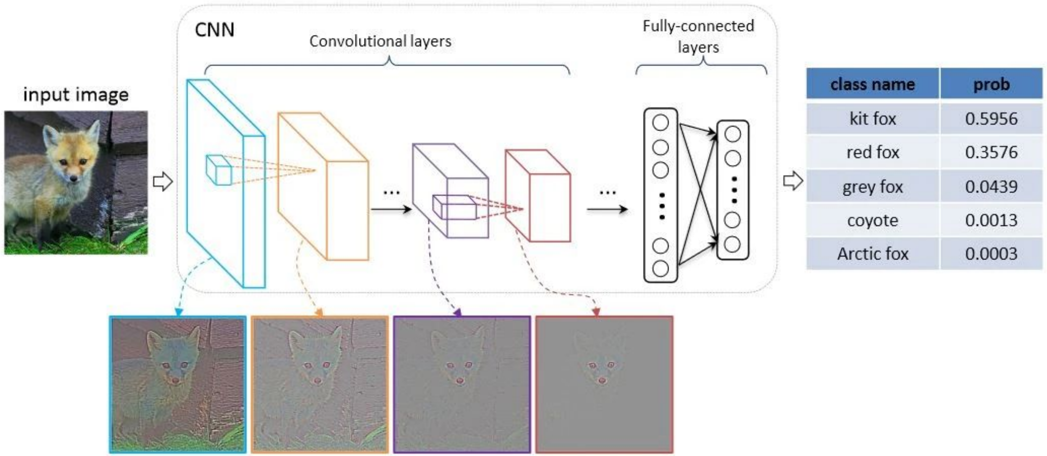
Hopefully, Machine vision with intelligence can do it better than human doctor

VGG19 Architecture

- VGG19 is a deep CNN used to classify images
- variant of VGG model which consists of 19 layers (16 convolution layers, 3 Fully connected layer, 5 MaxPool layers and 1 SoftMax layer)
- weights are easily available with other frameworks like keras
- VGG19 has learned rich feature representations for a wide range of images



How VGG 19 works?



Input: The VGGNet takes in an image input size of 224×224

Convolutional Layers: VGG’s convolutional layers leverage a minimal receptive field, i.e., 3×3. Moreover, there are also 1×1 convolution filters acting as a linear transformation of the input. This is followed by a ReLU unit, which is a huge innovation from AlexNet that reduces training time. The convolution stride is fixed at 1 pixel to keep the spatial resolution preserved after convolution

Hidden Layers: All the hidden layers in the VGG network use ReLU

Fully-Connected Layers: The VGGNet has three fully connected layers.

Reasons to use VGG19

Performance of VGG Models [1]

VGG16 result is competing for the classification task winner (GoogLeNet with 6.7% error) and considerably outperforms the ILSVRC-2013 winning submission Clarifai.

VGGNet with more layers, such as VGG20, or VGG50, or VGG100? This is where the problem arises.

The weights of a neural network are updated through the backpropagation algorithm, which makes a minor change to each weight so that the loss of the model decreases.

However, as the gradient keeps flowing backward to the initial layers, the value keeps increasing by each local gradient. This results in the gradient becoming smaller and smaller, thereby making changes to the initial layers very small. This, in turn, increases the training time significantly.

Steps for Image Classification Development

Step 1 : Collecting Dataset (from Kaggle)

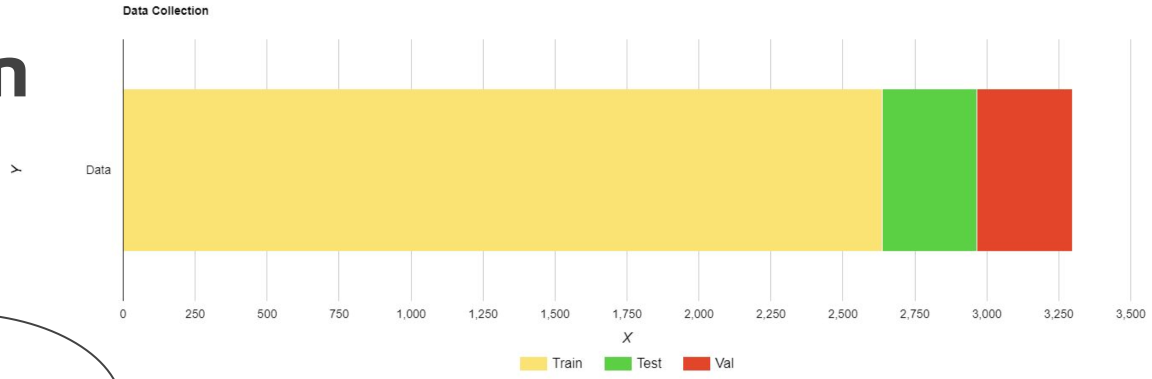
Step 2: Building Model

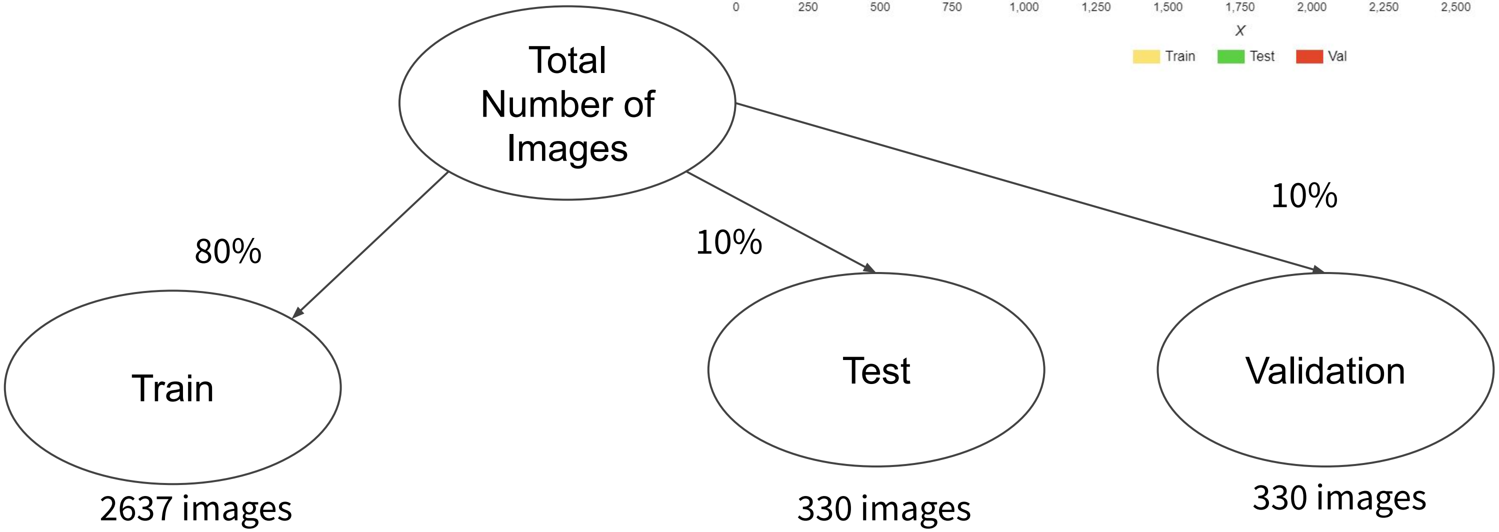
Step 3: Training

Step 4: Evaluation metrics - Model predict on TEST DATA = 330 files

Step 5: Displaying the Data on Graphical User Interface (GUI), using Streamlit

Data Collection





Model Architecture - VGG19

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv4 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv4 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv4 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten_1 (Flatten)	(None, 25088)	0
dense_1 (Dense)	(None, 3)	75267

Total params: 20,099,651
Trainable params: 75,267
Non-trainable params: 20,024,384

Results

- Evaluation Metrics

	precision	recall	f1-score	support
0	0.77	0.85	0.81	164
1	0.83	0.75	0.79	166
accuracy			0.80	330
macro avg	0.80	0.80	0.80	330
weighted avg	0.80	0.80	0.80	330

Label 0 = Benign
Label 1 = Malignant

Results

- Evaluation Metrics

		Precision	Recall	F1-score	Support
Benign	0	0.77	0.85	0.81	164
Malignant	1	0.83	0.75	0.79	166
Accuracy				0.80	330
Macro avg		0.80	0.80	0.80	330
Weighted avg		0.80	0.80	0.80	330

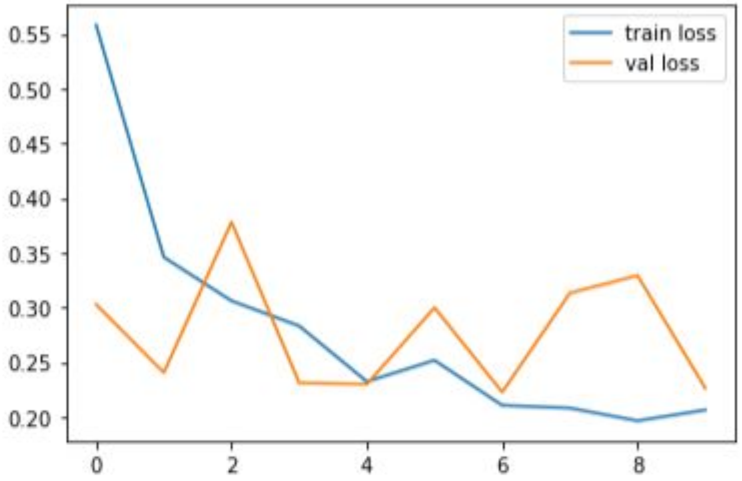
Results

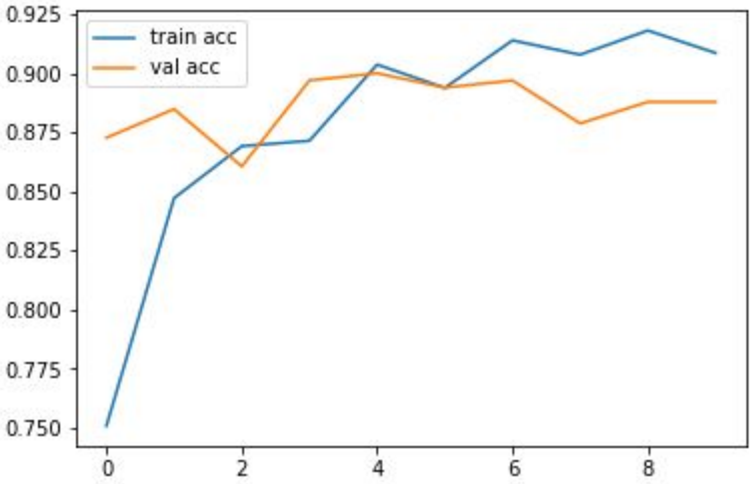
- Confusion Matrix

		Benign	Malignant
Benign	0	139	25
Malignant	1	41	125

Training graph on Losses and accuracy

Label 0 = Benign
Label 1 = Malignant





Coding

```
#%% Load Images Files

x_train=[]

for folder in os.listdir(train_path):
    sub_path=train_path+"/"+folder
    for img in os.listdir(sub_path):
        image_path=sub_path+"/"+img
        img_arr=cv2.imread(image_path)
        img_arr=cv2.resize(img_arr,(224,224))
        x_train.append(img_arr)

x_test=[]

for folder in os.listdir(test_path):
    sub_path=test_path+"/"+folder
    for img in os.listdir(sub_path):
        image_path=sub_path+"/"+img
        img_arr=cv2.imread(image_path)
        img_arr=cv2.resize(img_arr,(224,224))
        x_test.append(img_arr)

x_val=[]

for folder in os.listdir(val_path):
    sub_path=val_path+"/"+folder
    for img in os.listdir(sub_path):
        image_path=sub_path+"/"+img
        img_arr=cv2.imread(image_path)
        img_arr=cv2.resize(img_arr,(224,224))
        x_val.append(img_arr)

train_x=np.array(x_train)
test_x=np.array(x_test)
val_x=np.array(x_val)

train_x.shape,test_x.shape,val_x.shape

#%% Data normalization

train_x=train_x/255.0
test_x=test_x/255.0
val_x=val_x/255.0

from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

Coding

```
#%% Model training

# add preprocessing layer to the front of VGG
vgg = VGG19(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)

# don't train existing weights
for layer in vgg.layers:
    layer.trainable = False

# our layers - you can add more if you want
x = Flatten()(vgg.output)

prediction = Dense(3, activation='softmax')(x)

# create a model object
model = Model(inputs=vgg.input, outputs=prediction)

#%% Model Architechture

# view the structure of the model
model.summary()

#%% Cost and optimization use

model.compile(
  loss='sparse_categorical_crossentropy',
  optimizer="adam",
  metrics=['accuracy']
)
```

Epochs

```
Epoch 1/10
83/83 [==============================] - 227s 3s/step - loss: 0.6068 - accuracy: 0.7281 - val_loss: 0.2693 - val_accuracy: 0.8939
Epoch 2/10
83/83 [==============================] - 213s 3s/step - loss: 0.4001 - accuracy: 0.8278 - val_loss: 0.2466 - val_accuracy: 0.8939
Epoch 3/10
83/83 [==============================] - 214s 3s/step - loss: 0.3000 - accuracy: 0.8635 - val_loss: 0.2341 - val_accuracy: 0.8970
Epoch 4/10
83/83 [==============================] - 215s 3s/step - loss: 0.2823 - accuracy: 0.8760 - val_loss: 0.3339 - val_accuracy: 0.8636
Epoch 5/10
83/83 [==============================] - 224s 3s/step - loss: 0.2626 - accuracy: 0.8866 - val_loss: 0.3046 - val_accuracy: 0.8758
Epoch 6/10
83/83 [==============================] - 219s 3s/step - loss: 0.2282 - accuracy: 0.9037 - val_loss: 0.2268 - val_accuracy: 0.8788
Epoch 7/10
83/83 [==============================] - 220s 3s/step - loss: 0.2018 - accuracy: 0.9147 - val_loss: 0.2992 - val_accuracy: 0.8758
Epoch 8/10
83/83 [==============================] - 220s 3s/step - loss: 0.2357 - accuracy: 0.9006 - val_loss: 0.3293 - val_accuracy: 0.8667
Epoch 9/10
83/83 [==============================] - 221s 3s/step - loss: 0.2094 - accuracy: 0.9075 - val_loss: 0.2913 - val_accuracy: 0.8879
Epoch 10/10
83/83 [==============================] - 223s 3s/step - loss: 0.1979 - accuracy: 0.9207 - val_loss: 0.4160 - val_accuracy: 0.8485
```

Challenges of Skin Cancer project

- Accuracy of detection is roughly 0.8 (slightly good enough)
- Only one Detection algorithm is used : VGG-19
- More classes on skin cancer type such as :

1. Melanocytic nevi
2. Melanoma
3. Benign keratosis-like lesions
4. Basal cell carcinoma
5. Actinic keratoses
6. Vascular lesions
7. Dermatofibromaes

Future Development

- Include more image of skin cancer in the database
- Develop a mobile application for users
- Add sound to recognise whether it malignant or benign

Conclusion

- This project could help Malaysians to detect skin cancer without consulting dermatologist. Assisting doctors in diagnosis.
- Can save money and have better prediction.

References

1. VGG19 citation1