Image Classification

Convolutional Neural Network

Introduction

- Malaria is a mosquito-borne infectious disease that affects humans and other animals.
- Malaria kills one child every 30 seconds, about 3000 children every day.
- An estimated 300-600 million people suffer from malaria each year.
- More than 40 percent of the world's population lives in malaria-risk areas.

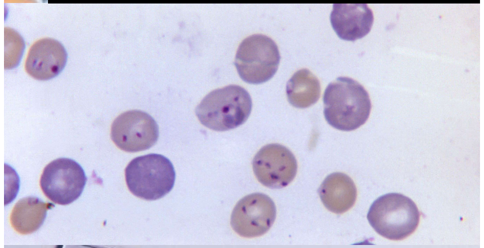
Factors

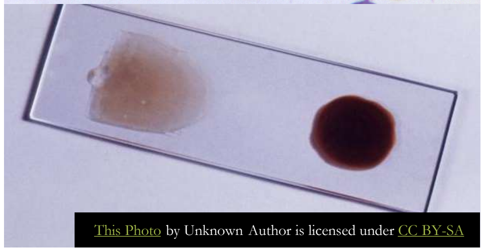
This Photo by Unknown Author is licensed under CC BY-NC

This Photo by Unknown Author is licensed under CC BY

- High poverty levels
- Political instability
- Presence of disease transmission vectors (ex. mosquitos)
- Densely populated areas

This Photo by Unknown Author is licensed under CC BY-SA-NC





This Photo by Unknown Author is licensed under CC BY-SA

Diagnosis

Most widely used method :

- Examining patients thin blood smears under a microscope
- The patients’ blood is smeared on a glass slide and stained with contrasting agents
- Clinician manually counts the number of parasitic red blood cells—sometimes up to 5,000 cells

Objective

- To reduce the burden for microscopists in resource-constrained regions and improve diagnostic accuracy
- Build an Image Classification model to obtain highest level of accuracy
- we must also consider making the model as small and computationally efficient as possible

Factors to consider

- Some of the factors to consider while building our model
  - Reliable power source
  - Battery-powered devices
  - Internet connection

Convolutional Neural Network

- CNNs use a variation of multilayer perceptron's designed to require minimal preprocessing.
- CNNs can be thought of automatic feature extractors from the image.
- While if I use a algorithm with pixel vector I lose a lot of spatial interaction between pixels, a CNN effectively uses adjacent pixel information to effectively downsample the image first by convolution and then uses a prediction layer at the end.

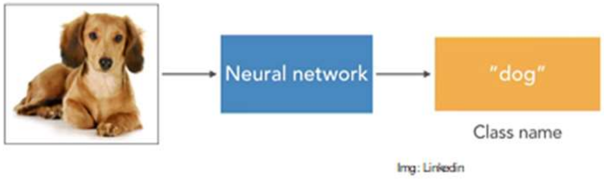
Overview

- The dataset contains 2 folders - Infected - Uninfected and it has total of 27,558 images.
- An instance of how the patient-ID is encoded into the cell name is shown herewith: "P1" denotes the patient-ID for the cell labeled "C33P1thinF_IMG_20150619_114756a_cell_179.png"
- **Source**:
  - This Dataset is taken from the official NIH Website: https://ceb.nlm.nih.gov/repositories/malaria-datasets/

This Photo by Unknown Author is licensed under CC BY-NC

Tools

- Python -3
- Jupyter Notebook
- TensorFlow
- Keras



Img: Linkedin

How it works?

- Image recognition is the ability for computers to look at a photograph and understand what's in the photograph.
- Here we're passing in a picture to the neural network, and the neural network is generating a label, dog, because that's the main object that appears in the picture.
- A neural network is made up of separate nodes called neurons. These neurons are arranged into a series of groups called layers. Nodes in each layer are connected to the nodes in the following layer. Data flows from the input to the output along these connections.

This Photo by Unknown Author is licensed under CC BY-NC

Technical Approach

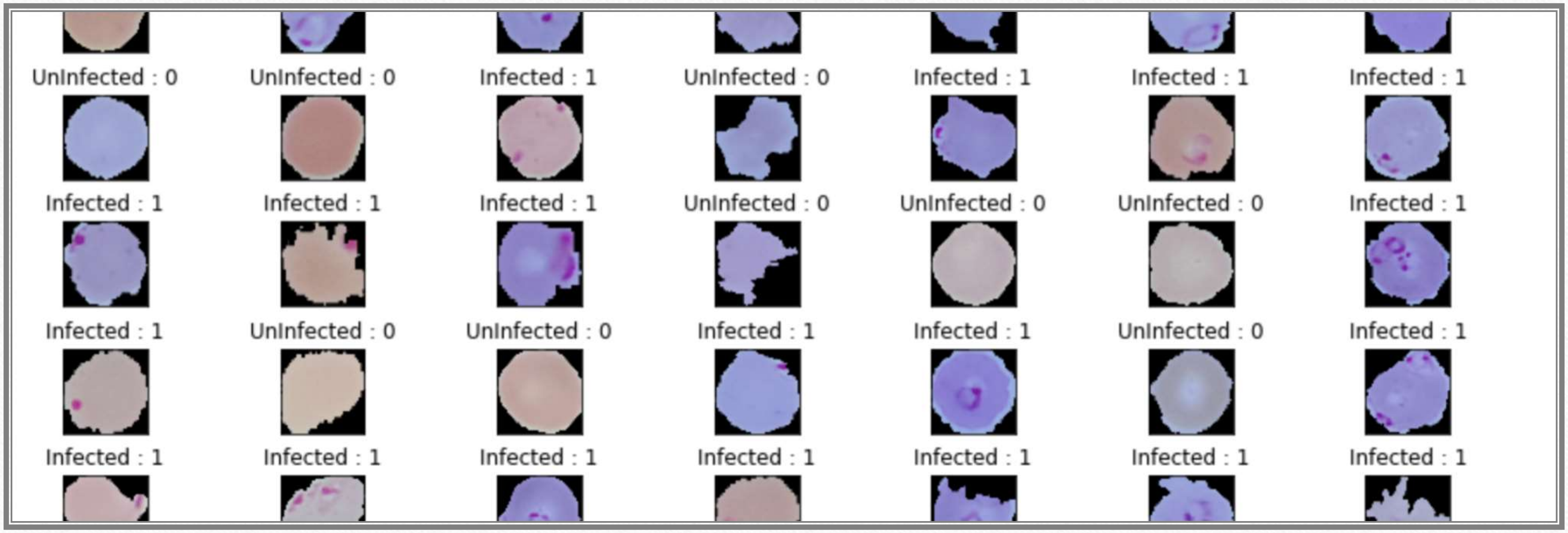
- Import required libraries
- Pre-processing image data
- Train and test data
- Building Model
- Compile & Training Model
- Transfer Learning

Import required libraries

- Keras
  - Conv2D, Max pooling, Dense, Flatten, Dropout, Load Img, Activation
- Scikit-Learn
- Matplotlib
- NumPy
- Pandas

Pre-processing image data

- Read and Image files
- Load image file
- Resize image file (50 x 50)
- Apply the same process for Parasitized & Uninfected
- Store and shuffle the data in array



Data Visualization

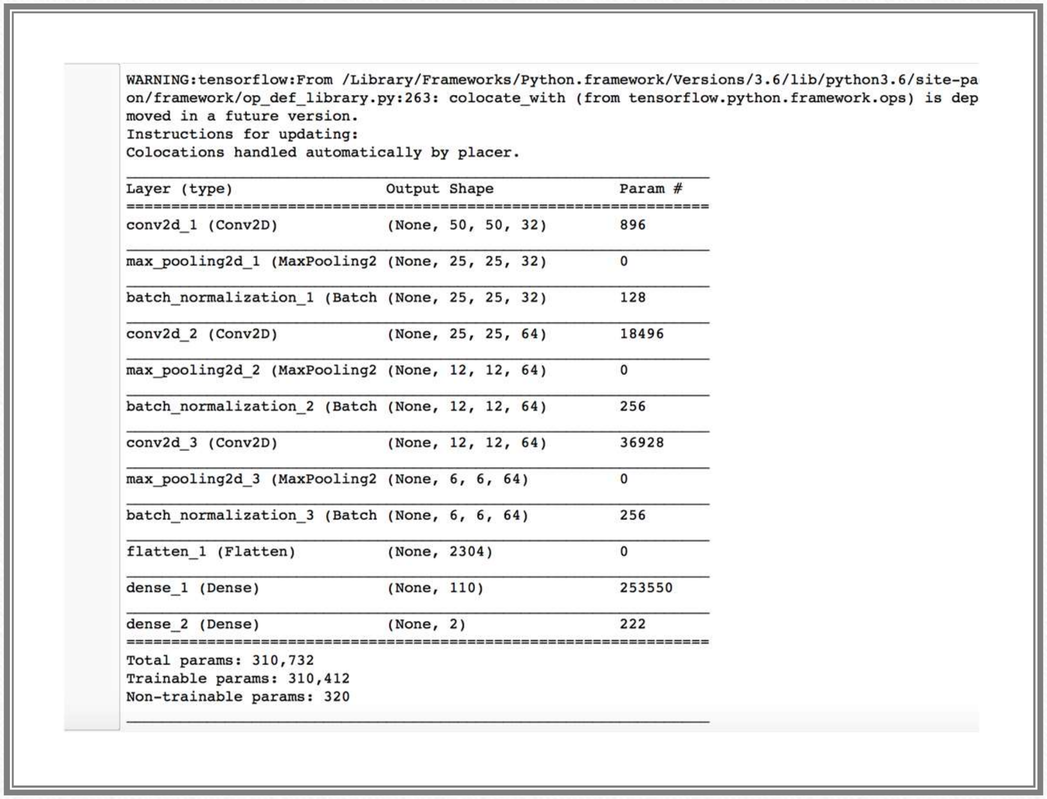
Normalize data

- Neural network best when the values are floating point values in between 0 and one.
- Normally images are stored as integer values for each pixel, this values is between 0 and 255, so convert the data to float and divide by 255.

Building Model

- Sequential
- Con2D :
- Maxpooling
- Batch Normalization
- Dropout

Model Summary



```
WARNING:tensorflow:From /Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-pa
on/framework/op_def_library.py:263: colocate_with (from tensorflow.python.framework.ops) is dep
moved in a future version.
Instructions for updating:
Colocations handled automatically by placer.
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_1 (Conv2D)            (None, 50, 50, 32)        896       
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 25, 25, 32)        0         
_________________________________________________________________
batch_normalization_1 (Batch (None, 25, 25, 32)        128       
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 25, 25, 64)        18496     
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 12, 12, 64)        0         
_________________________________________________________________
batch_normalization_2 (Batch (None, 12, 12, 64)        256       
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 12, 12, 64)        36928     
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 6, 6, 64)          0         
_________________________________________________________________
batch_normalization_3 (Batch (None, 6, 6, 64)          256       
_________________________________________________________________
flatten_1 (Flatten)          (None, 2304)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 110)               253550    
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 222       
=================================================================
Total params: 310,732
Trainable params: 310,412
Non-trainable params: 320
_________________________________________________________________
```

- Total params: 310,732
- Trainable params: 310,412
- Non-trainable params: 320

Model Compile

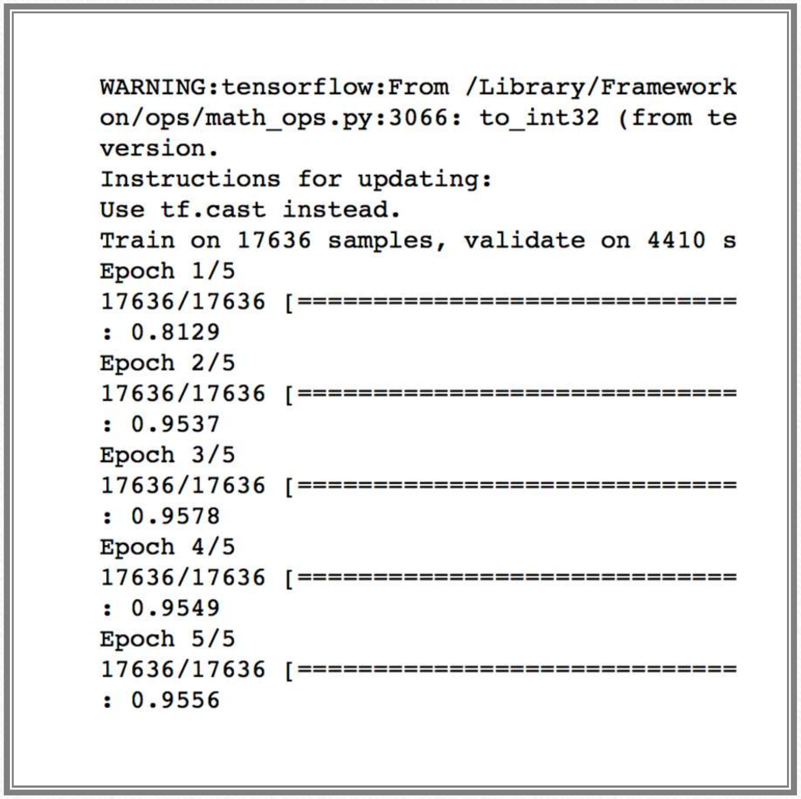
- Compile the model
- Params:
  - Loss
  - Optimizer
  - Metrics

Model Training

- Fit the data to Model
- Specify
  - Batch Size
  - Epochs
  - validation data

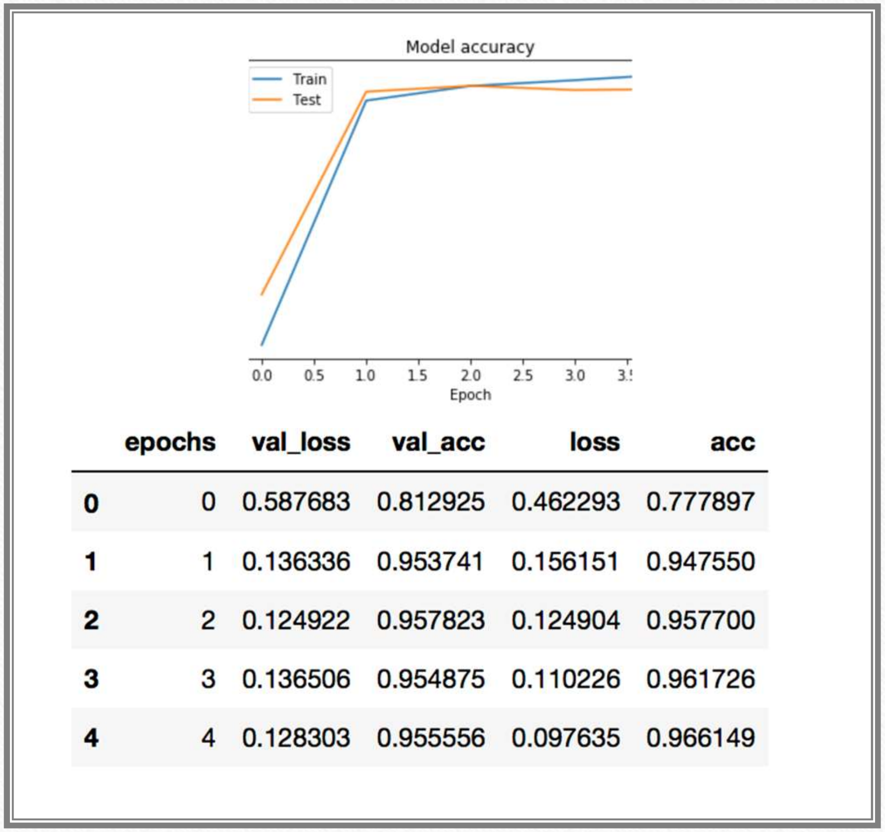
Training Results

- Epoch : 5



```
WARNING:tensorflow:From /Library/Framework
on/ops/math_ops.py:3066: to_int32 (from te
version.
Instructions for updating:
Use tf.cast instead.
Train on 17636 samples, validate on 4410 s
Epoch 1/5
17636/17636 [==============================
: 0.8129
Epoch 2/5
17636/17636 [==============================
: 0.9537
Epoch 3/5
17636/17636 [==============================
: 0.9578
Epoch 4/5
17636/17636 [==============================
: 0.9549
Epoch 5/5
17636/17636 [==============================
: 0.9556
```

Model Accuracy



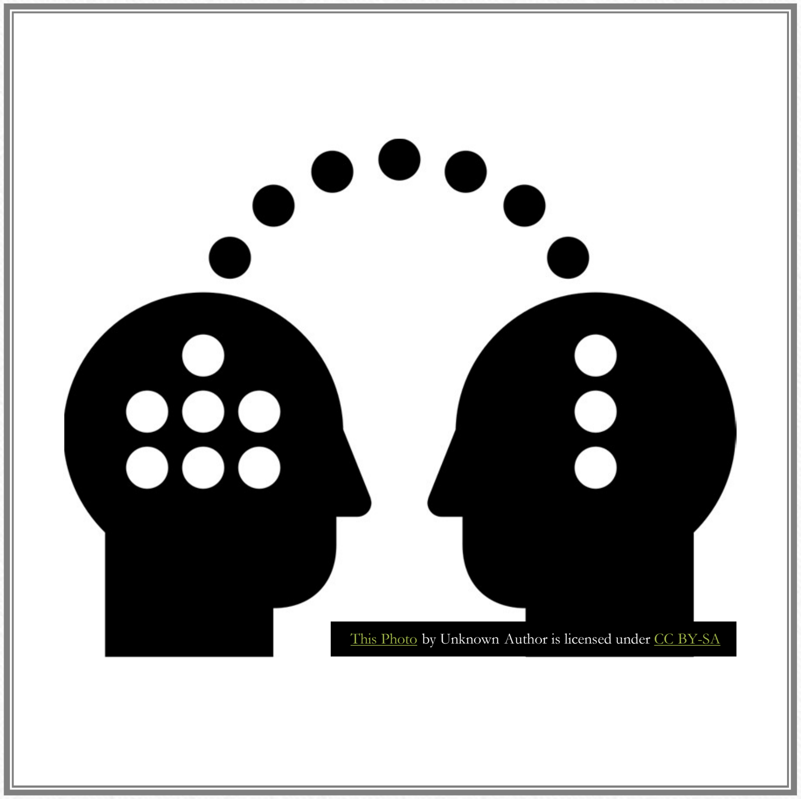
	epochs	val_loss	val_acc	loss	acc
0	0	0.587683	0.812925	0.462293	0.777897
1	1	0.136336	0.953741	0.156151	0.947550
2	2	0.124922	0.957823	0.124904	0.957700
3	3	0.136506	0.954875	0.110226	0.961726
4	4	0.128303	0.955556	0.097635	0.966149

Saving model weights

- Model structure
- Model Weights

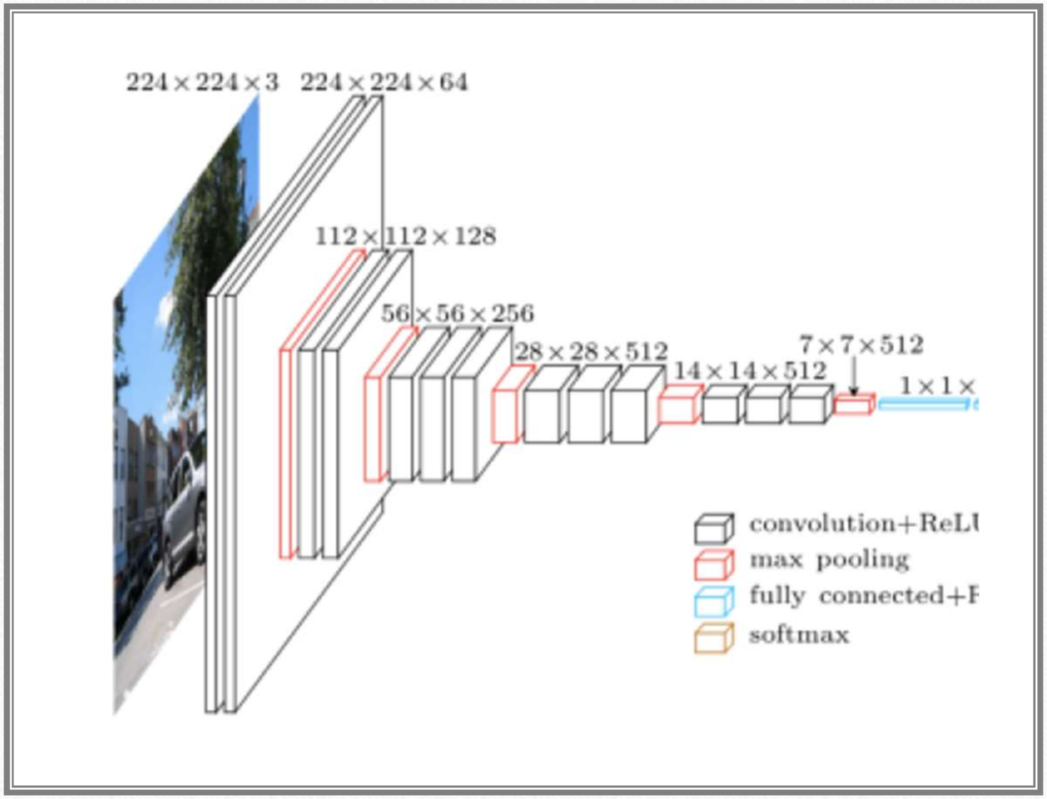
Model Testing with weights

- Import required libraries
- Load Model structure
- Load Model weights
- Load Image with target size
- Predict the image



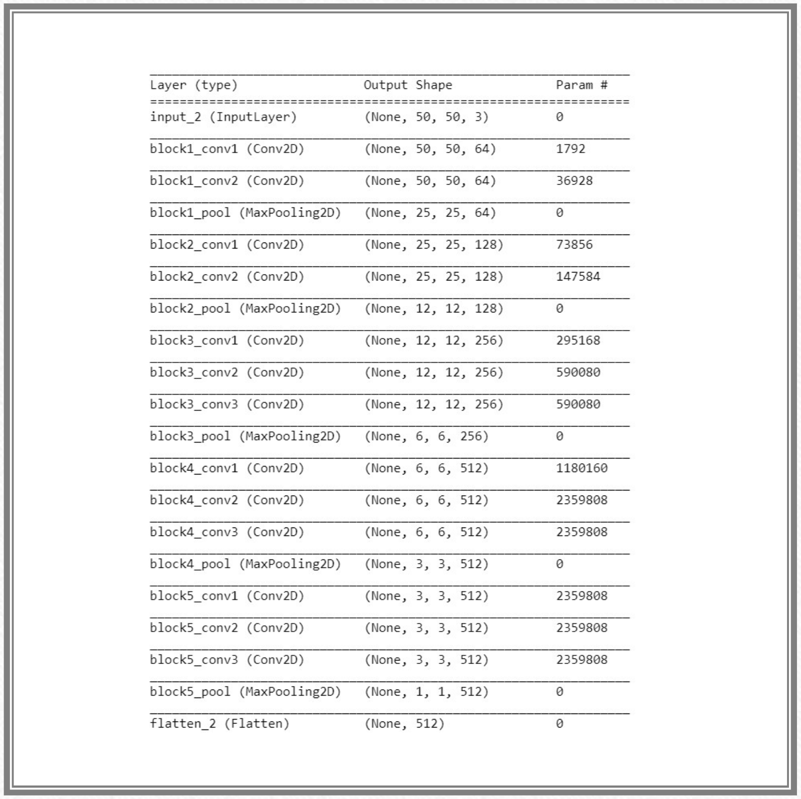
Transfer Learning (VGG16)

- ConvNet as fixed feature extractor.
- Fine-tuning the ConvNet
- Pretrained models



Load VGG16 model

- Total params: 14,714,688
- Trainable params: 14,714,688
- Non-trainable params: 0



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         (None, 50, 50, 3)         0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 50, 50, 64)        1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 50, 50, 64)        36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 25, 25, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 25, 25, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 25, 25, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 12, 12, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 12, 12, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 12, 12, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 12, 12, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 6, 6, 256)         0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 6, 6, 512)         1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 6, 6, 512)         2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 6, 6, 512)         2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 3, 3, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 3, 3, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 3, 3, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 3, 3, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 1, 1, 512)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 512)               0
```

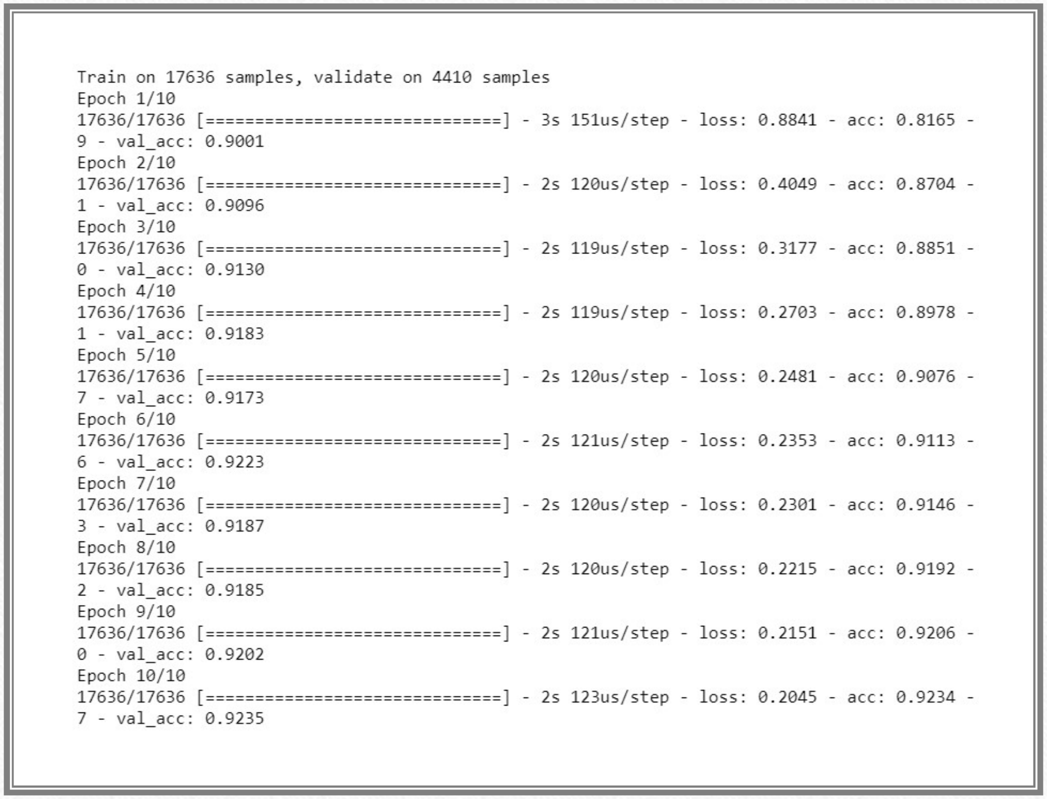
Flatten & Set the layer training to False

- Total params: 14,714,688
- Trainable params: 14,714,688
- Non-trainable params: 0

Features Extraction

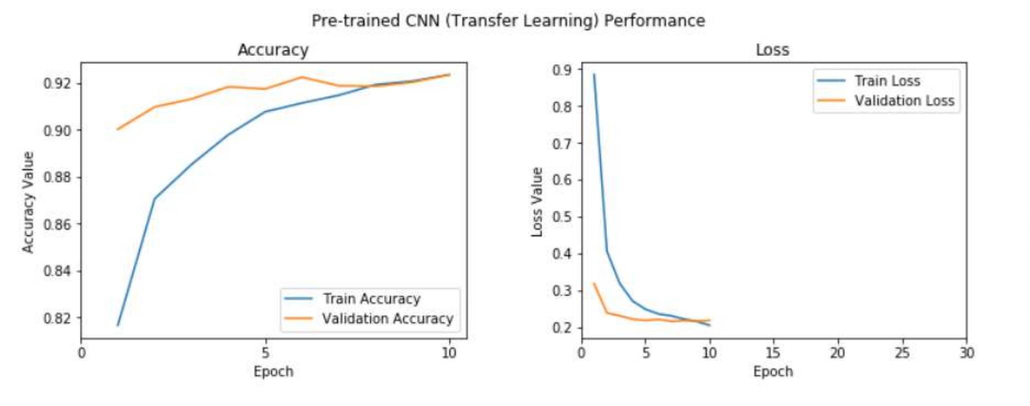
Train VGG Features: (17636, 512)
Validation VGG Features: (4410, 512)

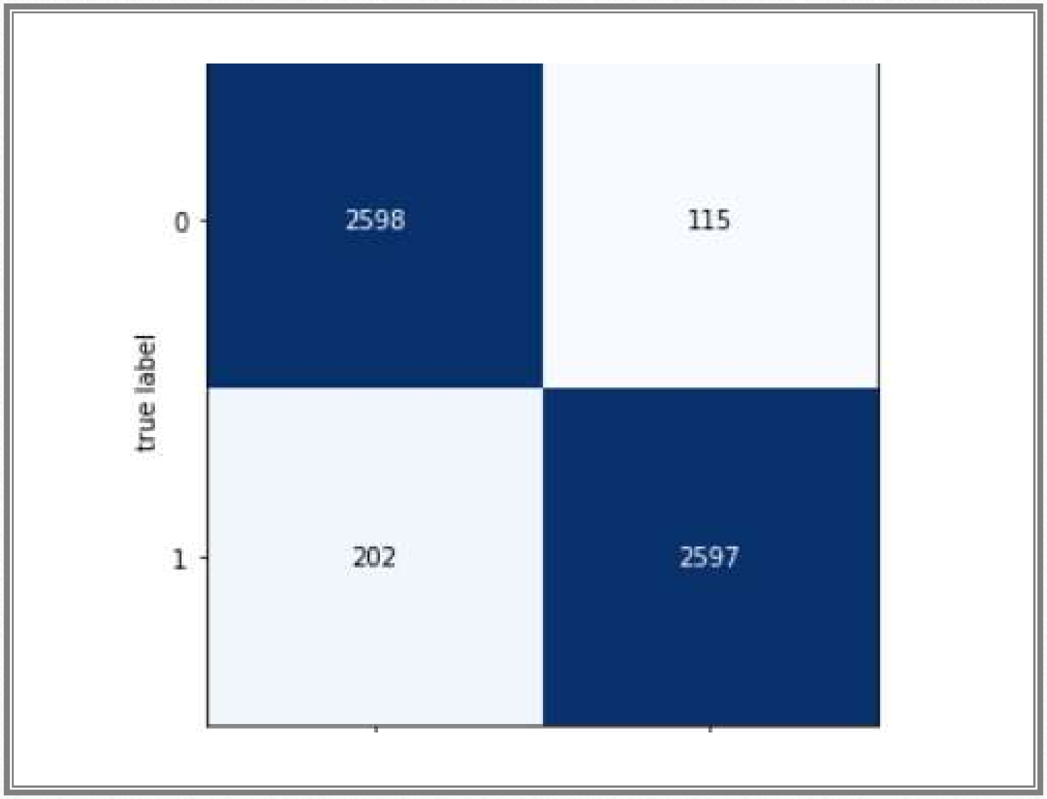
Train Model



```
Train on 17636 samples, validate on 4410 samples
Epoch 1/10
17636/17636 [==============================] - 3s 151us/step - loss: 0.8841 - acc: 0.8165 -
9 - val_acc: 0.9001
Epoch 2/10
17636/17636 [==============================] - 2s 120us/step - loss: 0.4049 - acc: 0.8704 -
1 - val_acc: 0.9096
Epoch 3/10
17636/17636 [==============================] - 2s 119us/step - loss: 0.3177 - acc: 0.8851 -
0 - val_acc: 0.9130
Epoch 4/10
17636/17636 [==============================] - 2s 119us/step - loss: 0.2703 - acc: 0.8978 -
1 - val_acc: 0.9183
Epoch 5/10
17636/17636 [==============================] - 2s 120us/step - loss: 0.2481 - acc: 0.9076 -
7 - val_acc: 0.9173
Epoch 6/10
17636/17636 [==============================] - 2s 121us/step - loss: 0.2353 - acc: 0.9113 -
6 - val_acc: 0.9223
Epoch 7/10
17636/17636 [==============================] - 2s 120us/step - loss: 0.2301 - acc: 0.9146 -
3 - val_acc: 0.9187
Epoch 8/10
17636/17636 [==============================] - 2s 120us/step - loss: 0.2215 - acc: 0.9192 -
2 - val_acc: 0.9185
Epoch 9/10
17636/17636 [==============================] - 2s 121us/step - loss: 0.2151 - acc: 0.9206 -
0 - val_acc: 0.9202
Epoch 10/10
17636/17636 [==============================] - 2s 123us/step - loss: 0.2045 - acc: 0.9234 -
7 - val_acc: 0.9235
```

Plot Accuracy and Loss





0	2598	115
1	202	2597

Model Performance

- Test Data:
  - Accuracy : 94.24 %
  - Precision: 95.76 %
  - Recall : 92.78 %
  - F-Measure: 94.2 %

Thank You