CS 412 Intro to Machine Learning

Project Report

on

Pneumonia classification from chest X-Ray images

By Giri Madhav(UIN:671915100)

1.1 Personal views about the fields and its applications:

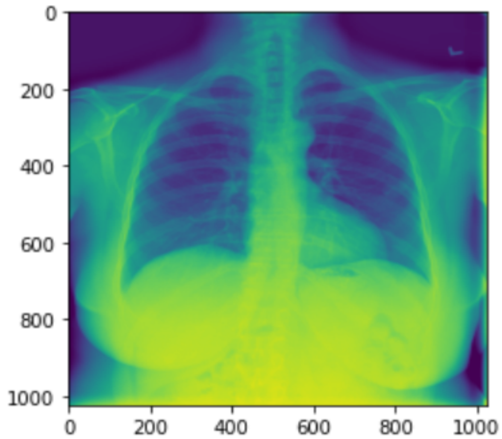
Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. The process of learning begins with data, in order to look for patterns in data and make better decisions in the future unseen data. Machine learning is used in many applications to solve real world problems. Health care is one of the important fields where machine learning can be applied vastly. ML in healthcare helps to analyze huge available data and suggest outcomes, provide timely risk scores, precise resource allocation, and has many other applications. A few major applications in health care based on Machine learning are Identifying diseases and diagnosis, Drug discovery, Medical imaging, Outbreak prediction. Machine Learning usage in Health care can probably help patients with early diagnosis of diseases. Treatment cost can be reduced, efficiency and speed of diagnosis can be improved. Pneumonia identification from chest X-Ray images is one of the important applications that can predict disease early. Deep Learning which is a part of Machine Learning has many advanced models which can efficiently deal with health care data. Due to huge availability of data, deep learning could be perfect to recognize patterns in the data and predict with high performance.

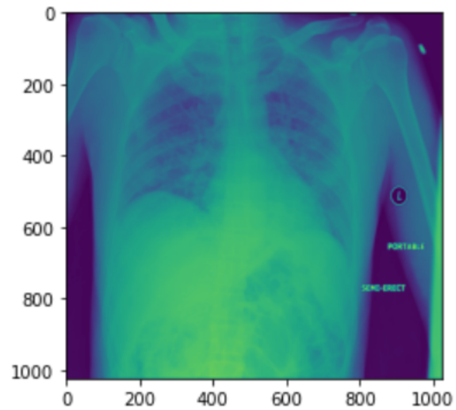
1.2. Brief description of learning in this course:

During this course, I have learned basic math required for understanding machine learning algorithms. Math include Linear algebra, Probability theory. Basics of python programming including most important machine learning packages and libraries. They include numpy, pandas, and sklearn. Different machine learning models which further categorized into supervised learning models and unsupervised learning models. I learnt different algorithms under these two categories and implemented code from scratch and also using sklearn library. This course helped me in implementing machine learning algorithms and deep learning models to the data, perform cross validation to tune hyper parameters and to evaluate model metrics. With these learnings of course i would like to deep delve into more deep learning methods that train on images, audio, video data.

2.Project Description:

Pneumonia Classification using chest X-Ray images is a binary classification task to classify images to pneumonia or normal. Dataset has 2425 chest X-Ray images of size 1024 x 1024 pixels with labels pneumonia and normal. Input is the image and output is 0 or 1 representing normal and pneumonia respectively. Below image to the left is labeled as normal image, image to right is labeled with pneumonia.





3.Approach and implementation:

Libraries used:

- Numpy
- Pandas
- Matplotlib
- OpenCV
- sklearn
- Tensorflow
- Keras

Workflow of the project is as below

- Data Preprocessing
    a. Read images
    b. Resize images
    c. Augmentation
- Data modeling
- Performance Evolution

3.1.Data Preprocessing:

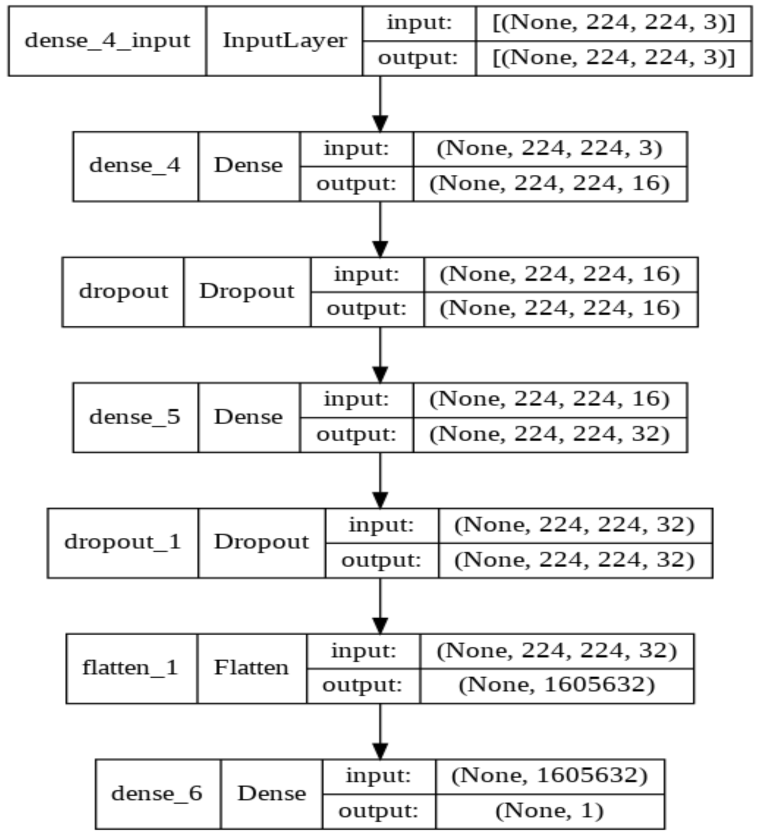
Images are read using opencv and converted to a numpy array of images. Each image is converted to the size of 224 x 224 pixels. Images are normalized so that pixel values of every image will be in the range of 0-1. Normalization helps machine learning and deep learning models to train faster and converge at global minimum quickly. Labels are encoded as numeric values(0,1) and then one hot encoded. Dataset size is increased by applying Data augmentation techniques like Rotating images, converting image to grayscale, Horizontal flip, Vertical flip, Blurring etc. Dataset is split to train data and validation /test data to calculate the performance metrics of the model, which is trained using train data, on test data. Both train and test data are converted as a generator object, which is passed to the model while fitting it.

3.2.Modeling:

Models used are:

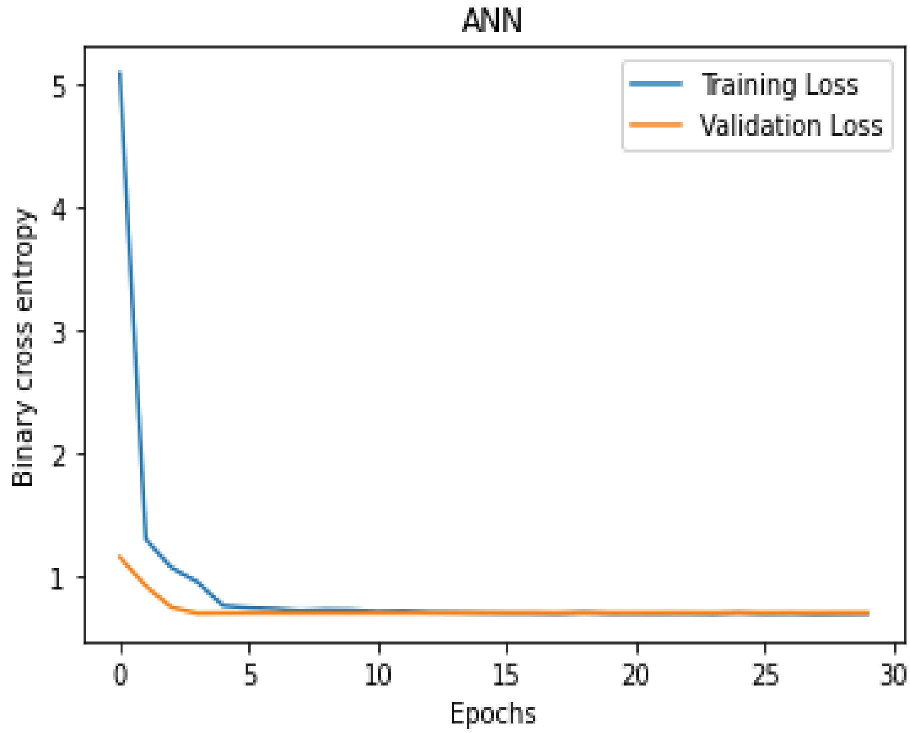
1. Artificial Neural Network
2. Convolutional Neural Network with dropout
3. Convolutional Neural Network with dropout and batch-normalization
4. Convolutional filters and Random Forest
5. Imagenet architecture

3.2.1.Artificial Neural Network: Artificial Neural Network is a simple network of neurons. Below image represents the architecture of ANN implemented. This is implemented using Keras and Tensorflow.



The model consists of 3 layers:

- Input layer feeds images with shape of 224 x 224 x 3
- Two hidden layers with dropout and activation function as RELU.
- Dropout is a regularization technique used to avoid overfitting of the model.The Dropout occurs only during training and not during testing.
- Output/Logit layer with 1 neuron with sigmoid activation function.
- Trained the model using AdamOptimizer with learning rate set to 1e-3, number of epochs= 30 and binary cross-entropy as error function.
- Callbacks are used to checkpoint and save the best model and also to stop early if no improvement in validation loss.



Above plot depicts the loss during training the model and validation. Model has not improved as loss has been constant after 4 epochs.

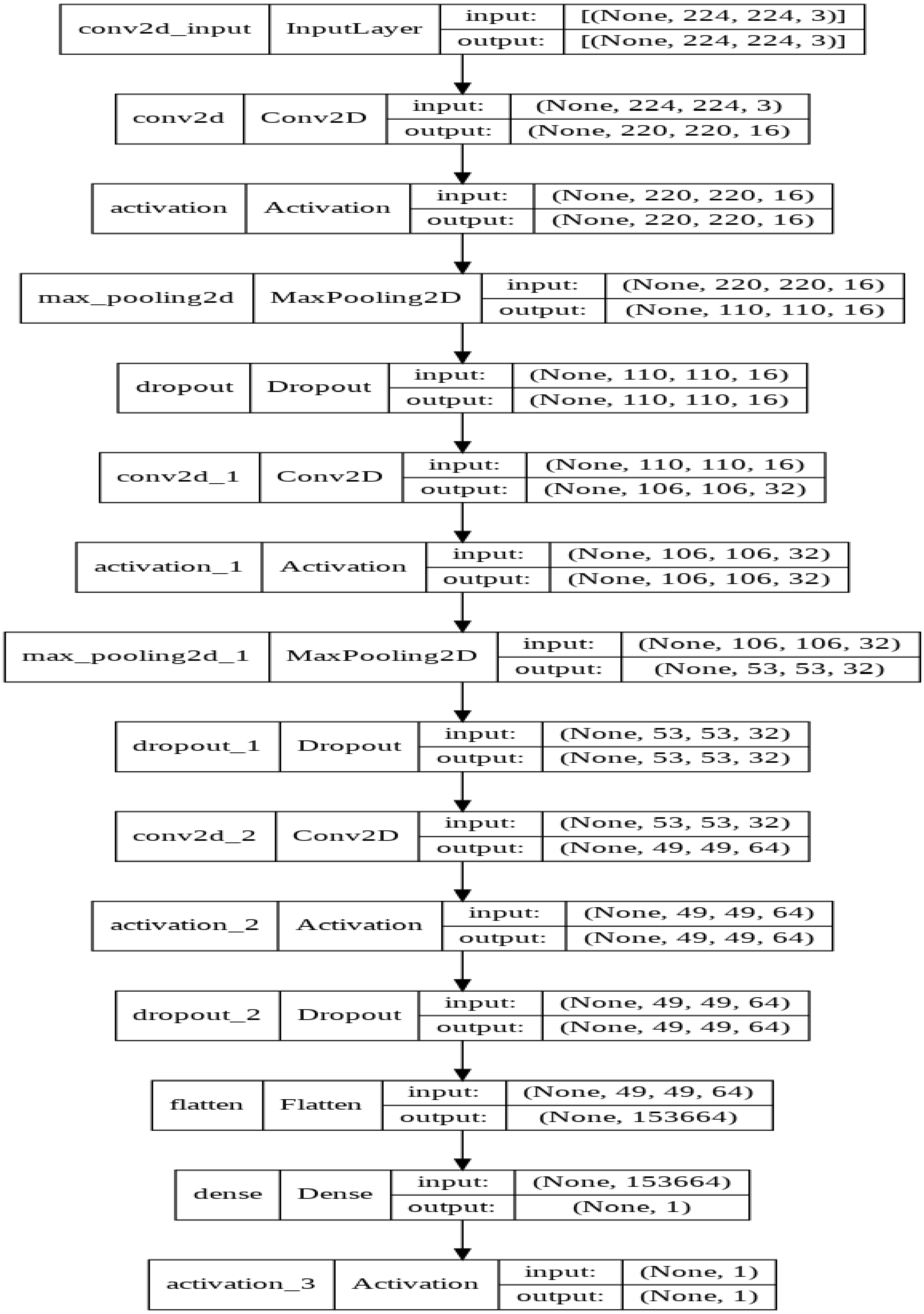
ANN performance metrics:

Train accuracy : 55

Test accuracy : 47

Training Time : 3 mins　　Inference Time : 12 secs

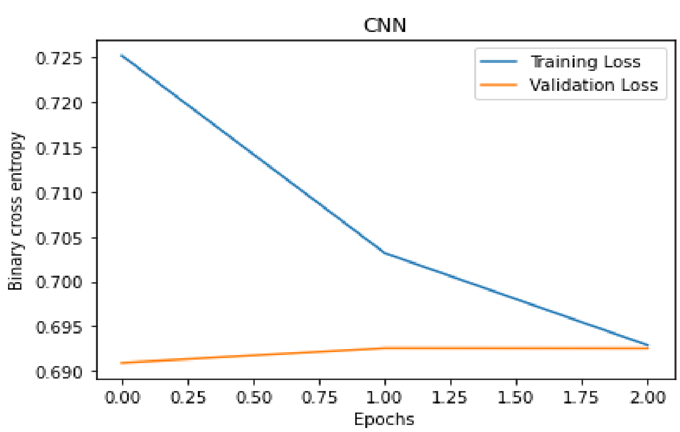
3.2.2.Convolutional Neural Network with dropout: Convolutional Neural Network is a network of fully connected hidden neuron layers and fully connected output layers. Below image represents the architecture of CNN implemented with dropout. This is implemented using Keras and Tensorflow.


conv2d_input | InputLayer | input: [(None, 224, 224, 3)] | output: [(None, 224, 224, 3)]
conv2d | Conv2D | input: (None, 224, 224, 3) | output: (None, 220, 220, 16)
activation | Activation | input: (None, 220, 220, 16) | output: (None, 220, 220, 16)
max_pooling2d | MaxPooling2D | input: (None, 220, 220, 16) | output: (None, 110, 110, 16)
dropout | Dropout | input: (None, 110, 110, 16) | output: (None, 110, 110, 16)
conv2d_1 | Conv2D | input: (None, 110, 110, 16) | output: (None, 106, 106, 32)
activation_1 | Activation | input: (None, 106, 106, 32) | output: (None, 106, 106, 32)
max_pooling2d_1 | MaxPooling2D | input: (None, 106, 106, 32) | output: (None, 53, 53, 32)
dropout_1 | Dropout | input: (None, 53, 53, 32) | output: (None, 53, 53, 32)
conv2d_2 | Conv2D | input: (None, 53, 53, 32) | output: (None, 49, 49, 64)
activation_2 | Activation | input: (None, 49, 49, 64) | output: (None, 49, 49, 64)
dropout_2 | Dropout | input: (None, 49, 49, 64) | output: (None, 49, 49, 64)
flatten | Flatten | input: (None, 49, 49, 64) | output: (None, 153664)
dense | Dense | input: (None, 153664) | output: (None, 1)
activation_3 | Activation | input: (None, 1) | output: (None, 1)

The model consists of 4 layers:

- Convolution layer 1 (convolution with 16 features applied + 2D max pooling + Dropout with 0.4)
- Convolution layer 2 (convolution with 32 features applied + 2D max pooling + Dropout with 0.6)
- Convolution layer 3 (convolution with 64 features applied on + 2D max pooling + Dropout with 0.8)
- All layers are fully connected layer and ReLU activation function
- Output/Logit layer with 1 neuron with sigmoid activation function
- Max Pooling is used to reduce the problem of overfitting.

In a fully connected layer, few neuron outputs are dropped out to prevent overfitting. The Dropout makes sure that dropout occurs only during training and not during testing. Trained the model using RMSProp optimizer with learning rate set to 1e-4 and chose that model which has the minimum cross-entropy error.



Loss function above depicted validation loss has increased. Whereas, Training data loss has been decreased continuously. This might be due to overfitting of the model.

CNN with dropout performance metrics:

Train Accuracy : 70 Precision : 50 Recall : 25

Test Accuracy : 53 Precision : 66 Recall : 20

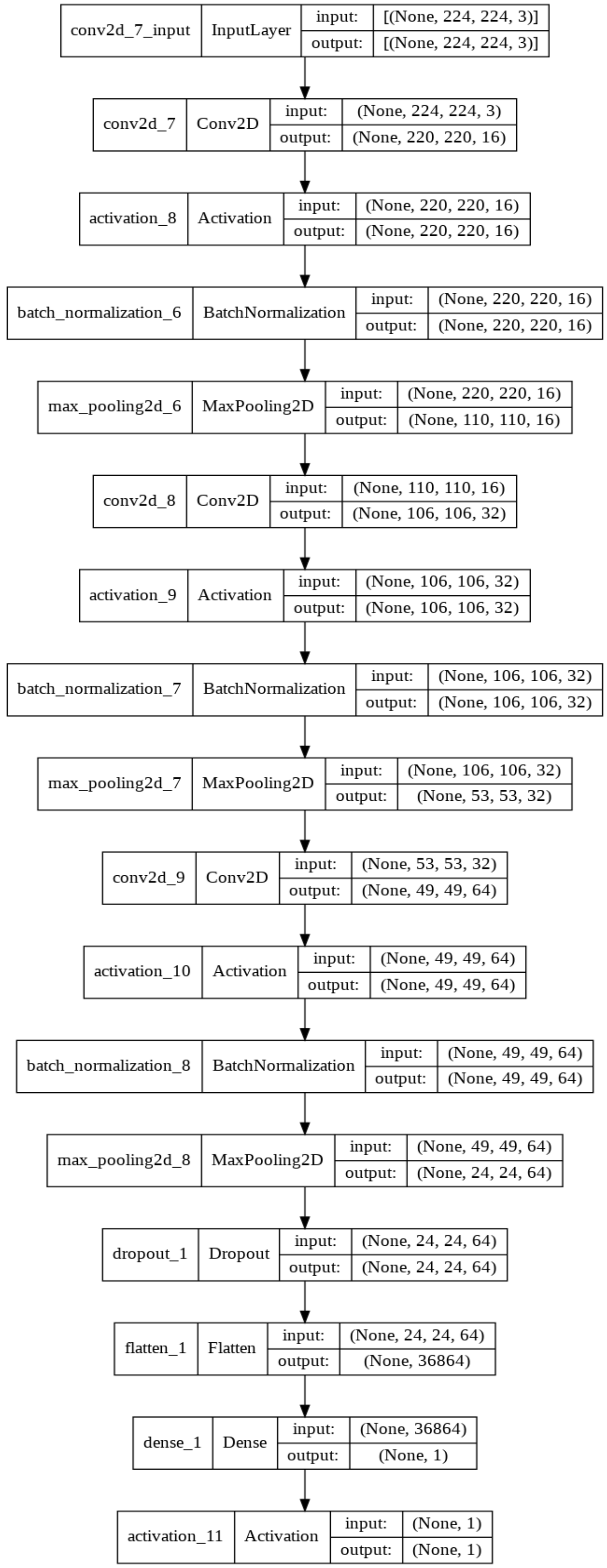
Training speed : 3 mins Inference speed : 11 secs

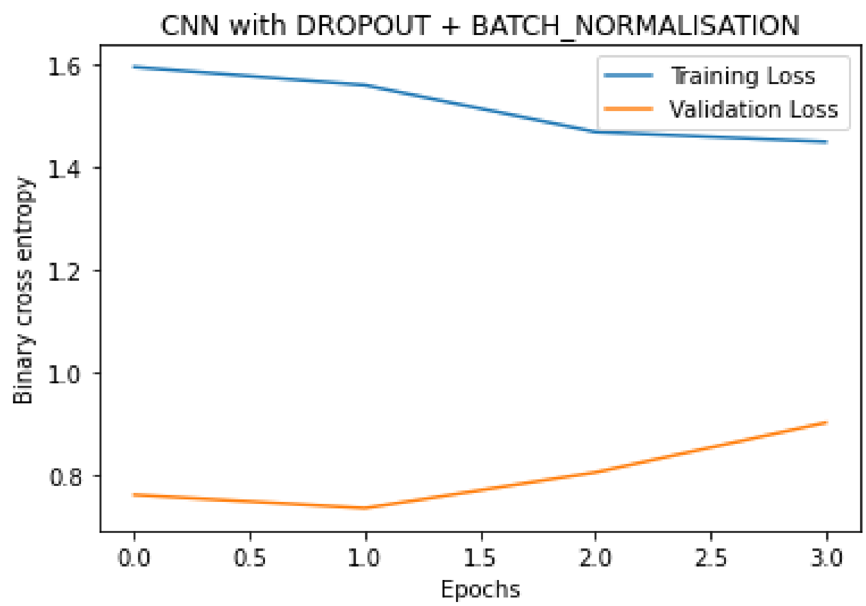
3.2.3.Convolutional Neural Network with dropout and batch-normalization:

Below image represents the architecture of CNN implemented with dropout and batch-normalization. This is implemented using Keras and Tensorflow.

The model consists of 4 layers:

- Convolution layer 1 (convolution with 16 features applied + 2D max pooling + Batch-Normalization)
- Convolution layer 2 (convolution with 32 features applied + 2D max pooling + Batch-Normalization)
- Convolution layer 3 (convolution with 64 features applied on + 2D max pooling + Batch-Normalization + Dropout with 0.5)
- All layers are fully connected layer and ReLU activation function
- Output/Logit layer with 1 neuron with sigmoid activation function


conv2d_7_input | InputLayer | input: [(None, 224, 224, 3)] | output: [(None, 224, 224, 3)]
conv2d_7 | Conv2D | input: (None, 224, 224, 3) | output: (None, 220, 220, 16)
activation_8 | Activation | input: (None, 220, 220, 16) | output: (None, 220, 220, 16)
batch_normalization_6 | BatchNormalization | input: (None, 220, 220, 16) | output: (None, 220, 220, 16)
max_pooling2d_6 | MaxPooling2D | input: (None, 220, 220, 16) | output: (None, 110, 110, 16)
conv2d_8 | Conv2D | input: (None, 110, 110, 16) | output: (None, 106, 106, 32)
activation_9 | Activation | input: (None, 106, 106, 32) | output: (None, 106, 106, 32)
batch_normalization_7 | BatchNormalization | input: (None, 106, 106, 32) | output: (None, 106, 106, 32)
max_pooling2d_7 | MaxPooling2D | input: (None, 106, 106, 32) | output: (None, 53, 53, 32)
conv2d_9 | Conv2D | input: (None, 53, 53, 32) | output: (None, 49, 49, 64)
activation_10 | Activation | input: (None, 49, 49, 64) | output: (None, 49, 49, 64)
batch_normalization_8 | BatchNormalization | input: (None, 49, 49, 64) | output: (None, 49, 49, 64)
max_pooling2d_8 | MaxPooling2D | input: (None, 49, 49, 64) | output: (None, 24, 24, 64)
dropout_1 | Dropout | input: (None, 24, 24, 64) | output: (None, 24, 24, 64)
flatten_1 | Flatten | input: (None, 24, 24, 64) | output: (None, 36864)
dense_1 | Dense | input: (None, 36864) | output: (None, 1)
activation_11 | Activation | input: (None, 1) | output: (None, 1)



CNN is modeled with decay 1e-6 and RMSProp optimiser and learning rate with below values.

- For learning rate: 0.0001

  Train -- accuracy : 51 precision : 48 Recall : 50

  Test -- accuracy : 55 precision: 50 Recall:70

  Training Speed : 5 mins Inference Speed : 12 secs

- For learning rate: 0.0005

  Train -- accuracy : 52 precision : 49 Recall : 49

  Test -- accuracy : 49 precision: 49 Recall: 47

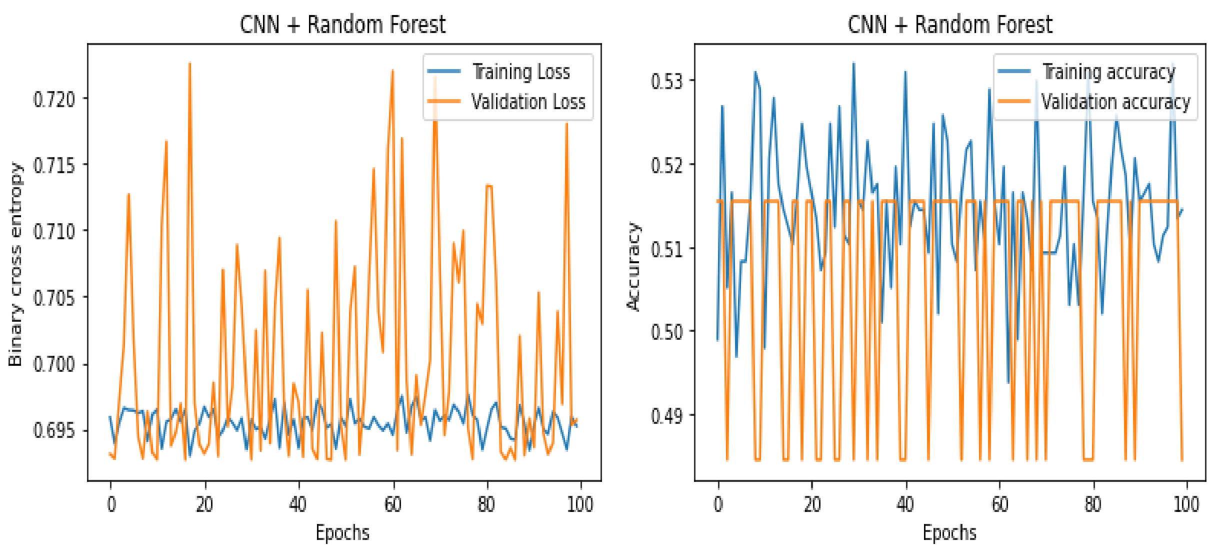
  Training Speed : 5 mins Inference Speed : 12 secs

Learning rate 0.0001 gave better performance metrics with RMSprop optimizer binary cross entropy loss.

3.2.4.Convolutional filters and Random Forest:

Convolutional filters are used to extract features from the images that can be passed as input to the Random forest machine learning algorithm. When the data is small, Machine learning models perform better than the deep learning models. Simple network of convolutions can be considered to extract features.

Feature Extractor has 4 layers of convolutions and Max pooling layers with sigmoid as activation function. Output of these convolution layers will be the extracted important features from the image. These features are passed as an input to svm for final classification output. Labels are encoded before modeling support vector machine algorithm.



For 100 epochs, RMSProp optimizer and categorical cross entropy above plots represent loss and accuracy for both training and validation datasets.

Evaluation metrics for this model are as below

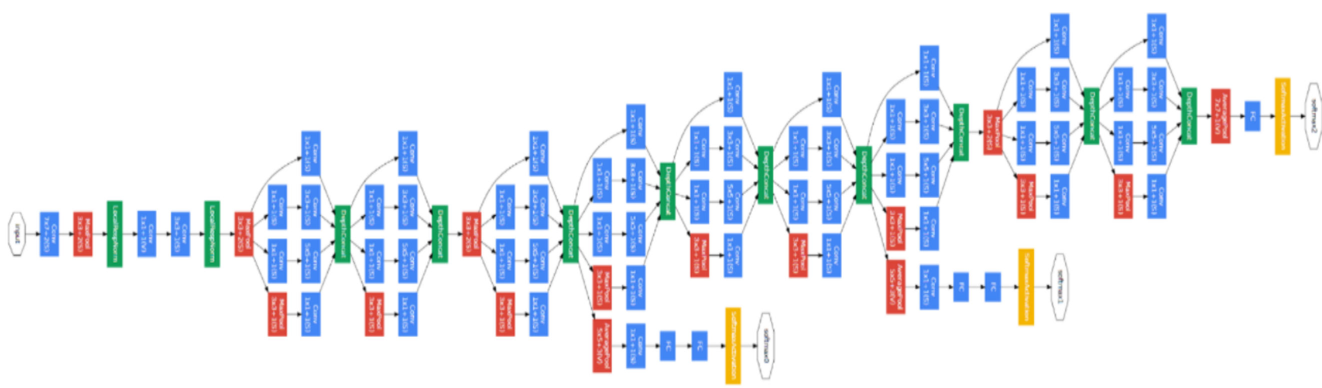
Train Accuracy :   51

Test Accuracy :   49

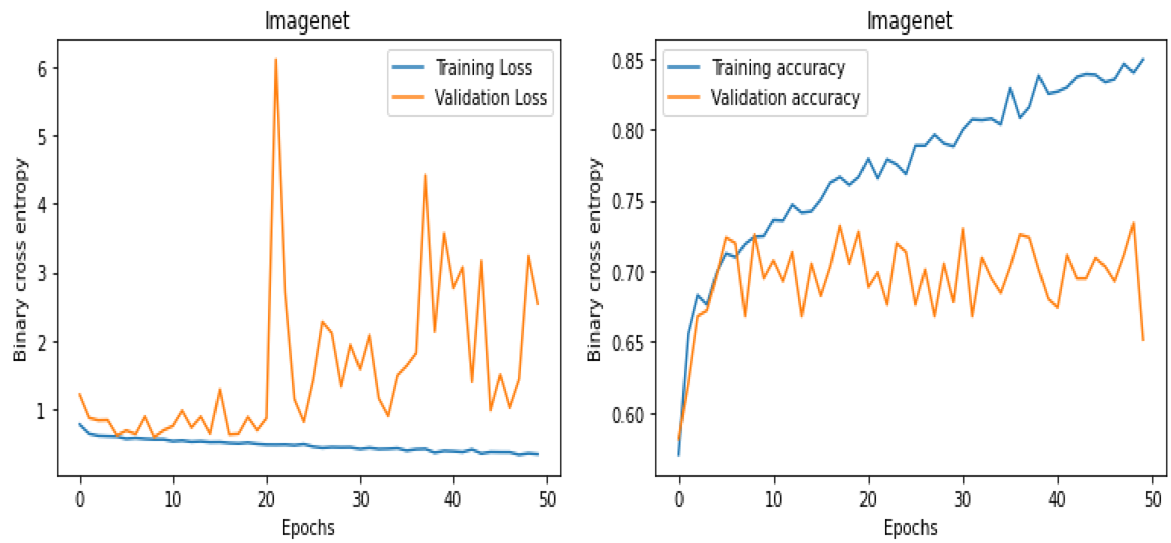
Training Time : 26 mins   Inference Time : 12 secs

3.2.5.ImageNet:

This is a transfer learning technique, weights of pre pre-trained model, imagenet, are used while classifying current tasks. Imagenet is a database of 14 million images. There are a few pre-trained models performed on imagenet database and model weights can be used. The InceptionV3 model is used here. Below is the architecture of inceptionV3.



InceptionV3 architecture has fully connected 27 layers of neurons. Below plots represent loss and accuracy for training and validation sets.



Validation accuracy has not increased more than 73. Whereas accuracy for training data reached about 84.

Train accuracy : 85 Precision : 84 Recall : 84.5

Test accuracy : 73 Precision : 63 Recall : 62

Training Speed : 64 mins Inference Speed : 14 secs

Comparison Table:

Model	ANN	CNN	CNN with Batch Normalization and Dropout	Convolutional filter and Random Forest	ImageNet
Accuracy	47	53	55	49	73

4.Conclusion:

- Imagenet inceptionV3 performed exceptionally well compared to other deep learning models.
- Due to more layers and convolutions used in imagenet architecture it achieved higher accuracy though training data is small.
- CNN with Batch Normalization and Dropout model performed better after inceptionV3 model.
- Accuracy of models improved as complexity kept on increasing.

5.Future Work:

More feature engineering techniques can be employed, Regularization methods which improve model by decreasing overfitting of the model can be used, more data augmentation techniques need to be used as training data is less, advanced deep learning models which might give much performance can be used, architectures like UNet, DNet which are more suitable in dealing with medical images can be used, rather classification image segmentation methods can be employed.

I think more data, good feature engineering techniques, advanced model architecture can help in improving results.

I would like to learn image data handling techniques and transfer learning techniques.