Machine learning Engineer Nano Degree

Capstone project

Classifying road Images: Potholes and Normal Images

Anvesh Samudrala

Project Overview:

Potholes are everywhere on the road. Whether it is because of the rain and snow or may be because of the road repair work and Drainage cleaning. In the countries like India, potholes are very dangerous compared to developed countries. As people walk a lot on the roads in India and there is no paved sidewalks, potholes play a major role in some of the fatalities on the road. There will be a lot of traffic jams because of the speed decrease to avoid them. They can able to damage the tires and suspensions of the vehicles. Because of this, Insurance premium would increase. Most of the pothole damages are not compensated completely. With this project, the model could recognize potholes from the normal roads. This could help other domains pretty easy. Some of the examples are for autonomous cars, predicting the right way without potholes. Or could help cities identify the potholes from regular transit cameras that could be used to cover up them cleanly.

Some of the facts about potholes are:

- Potholes are formed by water, freezing and freeze-thaw cycles, excessive heat, wear and tear – and time.
- The areas most prone to pothole development are where drainage is poor (particularly where roads dip, such as the trough under viaducts), where vehicular traffic is greatest – especially heavy vehicle traffic – and where poor maintenance allows small fissures to deteriorate.
- The vast majority of America’s highways were built in the 1950s through the 1970s. Most were built to last 50 years. With time, all the factors that compromise pavement add up. By the same token, some roads have

lasted much longer than planned in part due to good maintenance.

Problem Statement:

The main goal for this project is to use deep learning techniques and classify the provided images as Normal and Potholes. I have used CNN algorithms to classify image as either pothole or a normal image.

Metrics:

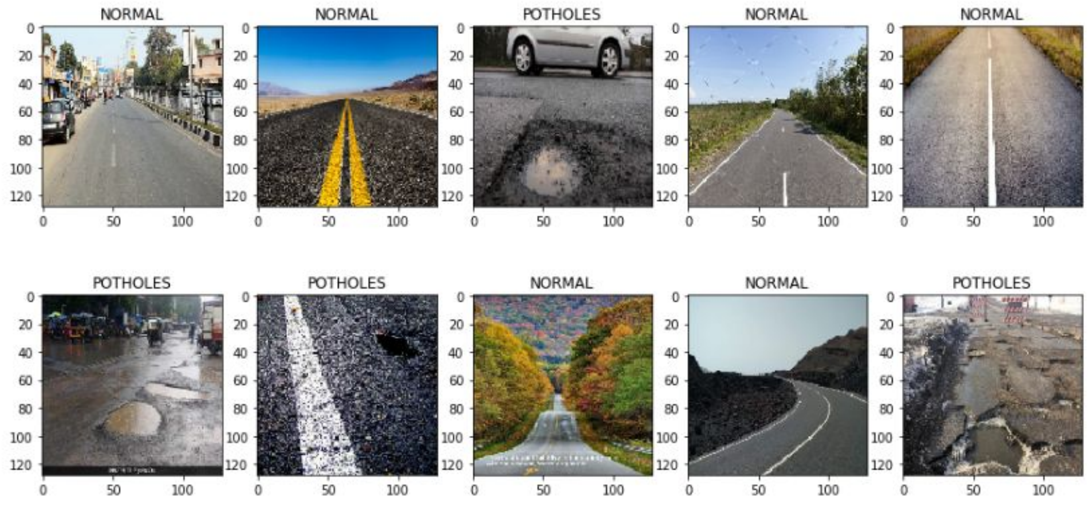
I have used Model`s Accuracy as the performance metrics. Accuracy calculates how accurately the data is being classified to the total data. Accuracy works better if we have almost equal samples in every class. Other metrics like Precision, recall, F1 score are being used if there is imbalanced datasets. Accuracy alone doesn't tell the full story when you're working with a **class-imbalanced data set** where there is a significant disparity between the number of positive and negative labels.For this classification problem, I believe that I can rule out other metrics and choose Accuracy alone to evaluate model performance.

Analysis:

Data cleaning and visualization:

For the classification problem, I have used Kaggle dataset. The dataset contains two folders - normal and potholes. 'Normal' contains images of smooth roads from different angles and 'Potholes' contains images of roads with potholes in them.

In this dataset, There are a total of 329 Pothole images and 352 normal images. The difference is not much between two classes so I would call the dataset as balanced dataset. As these are images, There is no scope for missing values.



Above image is the sample output from the dataset. As you can see, it is pretty hard to identify potholes as they are of the same colour as the road. The images of various sizes. The data cleaning steps would be resizing the images to preferred 224x224 size. As the images are colour, the properties of image is 224x224x3.

Algorithms and techniques:

Convolutional Neural Network is a Deep Learning algorithm, which takes the image as an input and specify the weights and biases to various objects/aspects in the image and that will be used to differentiate between images. It is also called multilayer perceptron. Convolutional layers are the major building blocks used in convolutional neural networks. A convolution is the simple application of a filter to an input that results in an activation. Repeated application of the same filter to an input results in a map of activations called a feature map, indicating the locations and strength of a detected feature in an input, such as an image. The innovation of convolutional neural networks is the ability to automatically learn a large number of filters in parallel specific to a training dataset under the constraints of a specific predictive modeling problems, such as image classification. The result is highly specific features that can be detected anywhere on input images. A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of a series of convolutional layers that convolve with a multiplication or other dot product. we use an activation function to make our output non-linear. In the case of a Convolutional Neural Network, the output of the convolution will be passed through the activation function. This could be Rectified Linear Units(ReLU) activation function. A rectified linear unit has output 0 if the input is less than 0, and raw output otherwise. That is, if the input is greater than 0, the output is equal to the input. ReLUs' machinery is more like a real neuron in your body. It is subsequently followed by additional convolutions such as pooling layers, fully connected layers and normalization layers, referred to as hidden layers because their inputs and outputs are masked by the activation function and final convolution.

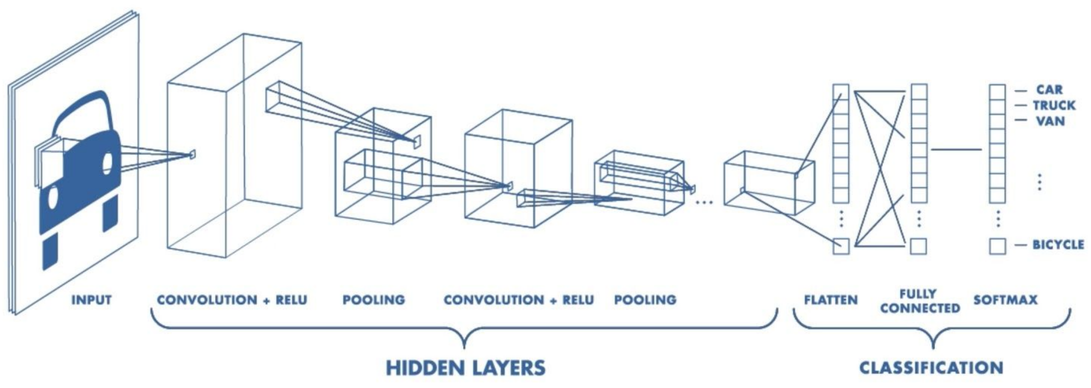


Image from Mathworks.

Benchmark model:

I am using the following model as a benchmark model. The accuracy of the model was recorded as 87%.

Methodology:

Data Pre-processing:

Data Pre-processing include :

- Importing images from the folder to python kernel
- Assign the labels to each of different classes of images
- Set the image size to 224x224 and convert the images to the respective size
- Convert the each image to numpy array which is X.
- Apply Label Encoder for Y (classes). LabelEncoder is a utility class to help normalize labels such that they contain only values between 0 and n_classes-1
- Convert the Y variable to categorical.
- Split the dataset to Test and Train dataset using **train_test_split** considering 25% to total data is for testing dataset.

Implementation:

Convolutional Neural Network is an Deep Learning algorithm, which takes the image as an input and specify the weights and biases to various objects/aspects in the image and that will be used to differentiate between images.

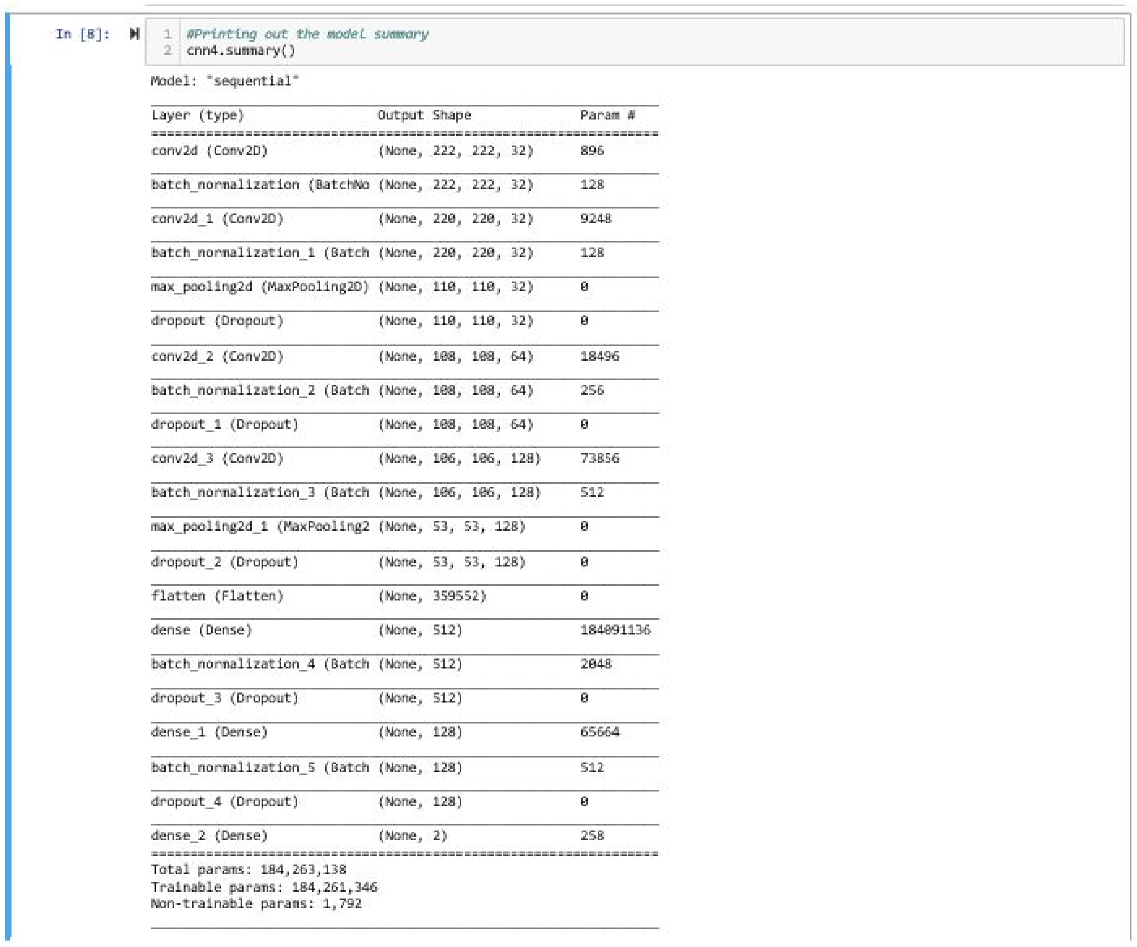
1. The input image is sent into the CNN which will first convert it to the feature image.
2. The feature image will undergo Pooling process.
3. The pooled feature image data will be flatten into array
4. The flatten data will be used to train the NN

This kernel will undergo the following steps:

1. Loading the necessary libraries
2. Initialize the Neural Network
3. Adding Convolution and MaxPooling
4. Flattening

5. Creation of NN layers
6. Create loss function
7. Mention metrics to be used
8. Define optimizer(used *adam*)
9. Training the model
10. Validate on test data
11. Predict the model

The summary of the model looks like following image:



In [8]:

```python
#Printing out the model summary
cnn4.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 222, 222, 32)      896
_________________________________________________________________
batch_normalization (BatchNo (None, 222, 222, 32)      128
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 220, 220, 32)      9248
_________________________________________________________________
batch_normalization_1 (Batch (None, 220, 220, 32)      128
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 110, 110, 32)      0
_________________________________________________________________
dropout (Dropout)            (None, 110, 110, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 108, 108, 64)      18496
_________________________________________________________________
batch_normalization_2 (Batch (None, 108, 108, 64)      256
_________________________________________________________________
dropout_1 (Dropout)          (None, 108, 108, 64)      0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 106, 106, 128)     73856
_________________________________________________________________
batch_normalization_3 (Batch (None, 106, 106, 128)     512
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 53, 53, 128)       0
_________________________________________________________________
dropout_2 (Dropout)          (None, 53, 53, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 359552)            0
_________________________________________________________________
dense (Dense)                (None, 512)               184091136
_________________________________________________________________
batch_normalization_4 (Batch (None, 512)               2048
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               65664
_________________________________________________________________
batch_normalization_5 (Batch (None, 128)               512
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 258
=================================================================
Total params: 184,263,138
Trainable params: 184,261,346
Non-trainable params: 1,792
_________________________________________________________________
```

The model takes input of **224x224x3** and gives the output of **2** classes(**normal** and **potholes**)

Some libraries like **Tensorflow** and **openCV** needs to be installed prior to importing them into Jupyter notebook.

Following modules/libraries were imported to run the model:

```
# Import of keras model for CNN
import tensorflow as tf
import tensorflow.keras
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, BatchNormalization
from tensorflow.keras.layers import Dropout, Flatten, Dense, Input, InputLayer
#Image handling libraries
import numpy as np
import matplotlib.pyplot as plt
import cv2
import pandas as pd
import random as rn
#Sklearn libraries
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import os
```

- Training accuracy has reached to 93% on 510 images after epoch 40.
- Testing accuracy has reached to 93% on 170 images.

Refinement:

Atfirst, I tried using VGG16 model for the classification. As this model is well known for the classification of Imagenet dataset with around 100 different classes. As VGG16 is considered as “Very Deep Convolutional Networks for Large-Scale Image Recognition”. The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. When i used that model, I have achieved model accuracy of around 65%. Then based on article from

TowardsDataScience, I learned that transfer learning doesn't work all the time on all datasets as these datasets were more focused on regular object detections.

Improvements and roadblocks:

The major roadblock for this model is if the image taken during night. As the night images are very dark, It would be pretty hard to identify the potholes. I would highlight that as the improvements required for the model. This classification would help in reinforcement learning or real time object detection.

Results:

Model Evaluation and Valuation:

During the model development phase the validation data was used to evaluate the model. The final model architecture and hyperparameters were chosen because they performed the best among the tried combinations. The Model was evaluated on test Dataset and Accuracy was recorded at 93%.

Compared to benchmark model:

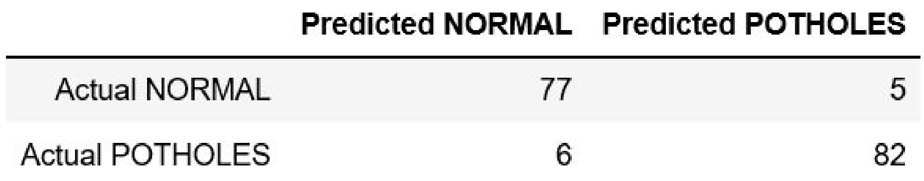
	Benchmark model	Developed model
Training accuracy	91%	93%
Testing accuracy	87%	93%

In the model prediction, I could see a huge improvement in predicting pothole images. For the benchmark model, 17 of normal images are predicted as potholes. Where as in developed model, the number has decreased to 5. This explains that model is more robust in predicting potholes and regular images.

Justification:

- Training accuracy has reached to 93% on 510 images after epoch 40.
- Testing accuracy has reached to 93% on 170 images.

Based on the following confusion matrix:



	Predicted NORMAL	Predicted POTHOLES
Actual NORMAL	77	5
Actual POTHOLES	6	82

- 5 of the normal images are predicted as potholes
- 6 of the potholes predicted as normal.

As these are just downloaded images, we do not know how the model would perform on Real-time data. We do not know whether it would be able to perform the classification in a timely manner so pothole images are not skipped or the classification would be heavily affected by latency. Also, we do not know how the classifier would perform in a real world setting.

References:

Pothole Damage Claims: Drivers Mostly Not Compensated, 9 Jan. 2017, www.pothole.info/2017/01/pothole-damage-claims-drivers-mostly-not-compensated/.

The Pothole Facts, www.pothole.info/the-facts/.

Le, James. “The 4 Convolutional Neural Network Models That Can Classify Your Fashion Images.” *Medium*, Towards Data Science, 7 Oct. 2018, towardsdatascience.com/the-4-convolutional-neural-network-models-that-can-c lassify-your-fashion-images-9fe7f3e5399d.

Nelson, Dan. “Image Classification with Transfer Learning and PyTorch.” *Stack Abuse*, Stack Abuse, 8 Aug. 2019, stackabuse.com/image-classification-with-transfer-learning-and-pytorch/.

Pradneshmhatre. “Pothole-Detector-with-Keras.” *Kaggle*, Kaggle, 25 Dec. 2019, www.kaggle.com/pradneshmhatre/pothole-detector-with-keras.