

# **Machine Learning (CSE4020)**

**PROJECT TITLE-**

## **TYRE DEFECT DETECTION**

**(Final Project Report)**

**By**

**RASHI KASERA (17BCE2421)**

**Course Instructor**

**Dr. Usha Devi G**

**School of Computer Science and Engineering**

**Abstract**—This project intends to train a deep neural network for training the model over various defected tire images till it has achieved intelligence to distinctly classify different tire defects and be able to tell apart various types of defects in tires by using computer vision. Main objective of this project is to create a system which can detect defects and allow the professionals to spend more time on trying to find a fix for it instead of spending time diagnosing it. It wishes to provide an automated Inspection system that is Reliant and Fast with Improved Business Sustainability at a Lower Cost.

The model will be created using Convolutional Neural networks (CNNs). In neural networks, Convolutional neural network (ConvNets or CNNs) is one of the main categories to do image recognition, image classifications hence using this algorithm is perfect for the model as this project needs to classify images into various labels according to the tire defects. Along with that fastAI library is used build on top of Pytorch which allows to train a Model on a certain Data Bunch very easily by binding them together inside a Learner object. This will ensure that a large number of data-sets can be trained quickly in order to yield efficient results in a minuscule amount of time.

## I. INTRODUCTION

Automatic detection technology plays an important role in industrial quality inspection, which lowers the risk of human intervention in a hazardous environment. Compared with human inspection, automatic defect detection has high efficiency and excellent performance while reducing labor costs. Most existing defect detection methods are based on hand-crafted features, defects in images can be detected by using these low-level features. A suitable detection method can greatly improve the processing speed while ensuring accuracy. The current system is highly skill dependent, and relies on the observation and correct diagnostic being made by the professional. Each year, about 11,000 tire-related crashes occur. Many of these wrecks results from either tire under-inflation or worn-down tread. Experts also estimate that 200 people die annually from weak-tire accidents. Therefore this proposal has introduced a new way to detect the manufactured tire detection.

For the model used in the project, a pretrained model of Resnet 34 is used which is trained on ImageNet and has been adapted to the data-set of the project. This was done to have a model that has already been trained to recognize a few features. Resnet has been used because it vanishes gradient problem and no learning is required. The objective was to preserve the gradient.

Fast ai library in python is also used because it is fast robust, portable and scalable. It is based on top of PyTorch. The fastai library allows to train a Model on a certain DataBunch very easily by binding them together inside a Learner object. It provides tools to preprocess and group the data. The vision module of the fastai library contains all the necessary functions to define a data-set and train a model for computer vision tasks. CNN-Learner helps in using the pre-trained model by modifying the last layers of the model to fit the data-set (custom head). The model has been trained in two phases:

- 1) Freeze the body weights and only train the head

- 2) Unfreeze the layers of the backbone (gradually if necessary) and fine-tune the whole model.

## II. PROBLEM DESCRIPTION

Each year, about 11,000 tire-related crashes occur. Many of these wrecks result from either tire under inflation or worn-down tread. Experts also estimate that 200 people die annually from weak-tire accidents. The manufacturing companies have to use several methods to assure that the product they deliver is with no defect. The common defect detection techniques include ultrasonic testing, thermal detection, X-ray detection, digital holography technology. Ultrasonic testing being the and so on. The ultrasonic and thermal detection are respectively based on the difference in the sound wave and the temperature. These two ways of detection are not intuitive.

### Existing System

The images of the tire are never used directly in the defect detection techniques. X-Ray images are used for it. As for now, X-Ray images are used in the industry for the particular defects. It is not convenient to convert a normal image into X-Ray image for everyone.

### Proposed System

Our objective is to create a system which can detect defects and allow the professionals to spend more time on trying to find a fix it instead of spending time diagnosing it. The type of defect (Tread Wear Indicator, Bulges, Sidewall Cracking, Linear Air and Exposed Cord) will be identified. A convolutional neural network (CNN) is a specific type of artificial neural network that uses perceptron, a machine learning unit algorithm for supervised learning, to analyze data. As the Data set which is considered, consists of images therefore CNN is the fundamental technique for classifying images. Resnet will take care of the maximum learning of weights possible without over-fitting or negative deviation of accuracy through means of vanishing or exploding gradients problem.

This project wishes to provide an automated Inspection system that is Reliant and Fast with Improved Business Sustainability at a Lower Cost.

## III. LITERATURE SURVEY

Defects in tire can be detected using many techniques. Some of these are:

**Defect Detection in tire X-Ray Images Using Weighted Texture Dissimilarity.** This technique is proposed by Guo, Q., Zhang, C., Liu, H., & Zhang, X [1] The paper proposes a detection method by using weighted texture dissimilarity to measure perceptual texture distortion, in which the distortion of a pixel is defined as the dissimilarity between the original feature value and the represented one of pixels.

Texture features of individual pixel are extracted using the

local kernel regression descriptor and represented as vectors of features. It follows the anomaly of each pixel which is determined by weighting the dissimilarity between the pixel's local neighbors and pixel itself. Lastly, the defects are caught by segmenting the anomaly map with a simple thresholding process. The paper introduce an efficient detection way for automatic quality inspection, which takes benefits of feature similarity of tire images and captures the texture distortion of each pixel by weighted averaging of the dissimilarity between this pixel and its neighbors. The proposed detection algorithm works well with both tread image and sidewall image. The main problem with the calculation of the anomaly data is how to get the dissimilarity metric.

Another technique is which is proposed by **Qidan Zhu, Xi-aotian Ai** [2] in their paper. This technique works something like this: The tire image is input into the network, the first step is to extract the defect features. This task is done by the convolutional part of the classification network. After that, the region proposal network uses the defect features obtained by the convolution to generate a proposal box and filter it. Finally, it uses the generated proposal box to crop the image features to get the predicted target feature area, and then perform pooling for the region of interest. The feature vectors with the same dimension and size can be obtained, and finally they are sent to a fully connected network for classification and the final frame is obtained. This method doesn't need to manually extract features and is a self-learning process. The developed tire defect autodetection software not only can detect tire defects in real time automatically, reduce people's work, but also can store the results in the background and build a defect database for future study and retrieval which acts as an advantage. Its required to optimize and improve the neural network parameters, improve the accuracy and reduce the false recognition rate, and will join the sound and light alarm system.

We can also detect tire defects using **Defect Automatic Detection for tire X-ray Images using Inverse Transformation of Principal Component Residual** which is proposed by **XueHong Cui, Yun and ChuanXu Wang** [3]. The method proposed is divided in sub methods such as Principal components and their corresponding textures in X-ray image, Defect isolation based on inverse transformation of principal component residual, Selecting the right number of principal components, Statistically controlled binarization method. The flow is input image, forward transform. Remove high energy components, inverse transform, thresholding and binarization for defect image. This method can either reveal the defect location or given roughly defect shape so that the tires can be repaired and defects can be identified. This method may not suitable to deal with too large-sized defects which severely damage the regularity nature texture of tire.

Another technique that can be used is **Detection of Impurity and Bubble Defects in Tire X-Ray Image Based on Improved Extremum Filter and Locally Adaptive-threshold Binarization** proposed by **Xiunan Zheng, Jianpei Ding, Zengzhi Pang, Jinping Li** [4]. This method proposes extremum filter that is designed to separate the background from the image. Then the background is processed by the improved self-adapt local binarization. After binarization the defect will be separated from background. Finally, the noise in the image can be

removed by the morphological filter. It can detect the abnormal texture, as well as can distinguish impurities from the defects. For blur bubbles, the difference of the gray value between bubbles and the normal part is very small which is hard to distinguish, so that the bubbles cannot be separated very well. Result for the gray bubbles are not better. It must be improvised in future.

From the above proposed methods we are using Deep Learning technique. This technique has many more applications namely: **Automatic detection of motion artifacts on MRI using Deep CNN** proposed by **Fantini I, Ritter L, Yasuda C** [5]. They proposed a technique to automatically detect the images containing motion artifacts on brain Magnetic Resonance Images (MRI) using Deep Convolutional Neural Networks (CNN). It's advantages are: Test is performed using images from subjects that were not part of the training group. Manual process is very difficult and fatiguing. To overcome this, motion artifacts can be detected. Some disadvantages are: Small amount of sample data (48 acquisitions) has been taken under consideration. Analysis can be done on the user.






Another application is **Deep Learning- Based Crack Damage Detection Using Convolutional Neural Networks** proposed by **Young- Jin Cha and Wooram Choi** [6]. This application uses CNN to build a classifier which detects concrete cracks from images. Input layer is the first layer. After passing through the architecture input data is generalized with spatial size reduction at L5. In the rectified linear unit (ReLU) layer, the vector, including the 96 elements is fed. After the convolution of C4 the softmax layer predicts whether each input image is cracked or intact. BN and dropout layers are also used. These two layers cannot be visualized. After L1, L3, and L5, BN layers come and after the BN layer of L5 a dropout layer is located. Compared to traditional approaches the proposed system of CNN- based detection of concrete cracks requires no feature extraction and calculation. and also gave lower level of noise. Another huge advantage is CNN's ability to learn features from a vast amount of training data. The method implemented by CNN requires a large amount of training data, which is a disadvantage for the system. Due to the nature of photographic images CNN's incapability of sensing internal features is another limitation of the system.

CNN can also be used for **Abnormality Detection in Mammography using Deep Convolutional Neural Networks** proposed by **Pengcheng Xi, Chang Shu and Rafik Goubran** [7]. In the proposed system they trained deep CNNs on cropped image patches and adapted them to full mammogram images. With training image patches from calcification and mass cases, a binary classifier is trained with state-of-the-art deep CNN architectures using transfer learning. The pre-trained CNNs are modified at output layers to have two output classes. The output layers are then fine-tuned while the first part of the network is frozen. The fine-tuned patch neural network is then used to localize mammographic abnormalities in full-size mammograms. Approach enables localizing abnormalities in one single forward pass. Feeding the full-size mammogram image into the patch classifier and computing class activation mapping near the end of the output layers produces a heatmap for the localization of abnormalities. Their approach demonstrates that deep convolutional neural network classifiers have remarkable localization capabilities despite no supervision on the location of abnormalities. In computer- aided

mammography, deep CNN classifiers cannot be trained directly on full mammogram images because of the loss of image details from resizing at input layers.

Yet another application is **Semantic Segmentation of Sewer Pipe Defects Using Deep Dilated Convolutional Neural Network [8]** proposed by **M. Z. Wanga and J.C.P. Chenga**. To address the problem of large spatial information loss in existing deep learning models, DilaSeg is proposed to improve the feature map resolution and to produce dense feature maps. The dense feature maps are obtained mainly through normal convolution, dilated convolution, multiscale dilated convolution and bilinear interpolation. After obtaining the dense feature maps, softmax function is applied for calculating the loss and producing the predicted labels for each pixel. DilaSeg improved segmentation accuracy significantly in terms of all the evaluation indices. It also increase feature map resolution. The detection speed reflects the possibility of real time segmentation. The training duration and converge iteration implements that it is difficult to train the model to achieve desired performance.

#### IV. DATA SET

Defect Name	Consequences	Picture	Specifications
Tread Wear Indicator (TWI)- Δ mark.	TWI- To identify the tread life.		<ul style="list-style-type: none"> <li>• Shape:- Isosceles Triangle</li> <li>• Thickness of Line:- 0.7 mm and above</li> <li>• Letter Height:- 0.5 mm and above</li> </ul>
Bulge	Indicates that cords have been destroyed in the carcass.		An externally visible bulge on the sidewall of the tyre
Side Wall Cracking	Ageing of tyre		Cracks on sidewall and tread
Exposed Cord	Cord open during high speed of vehicle		1. A fingernail does not catch on the exposed cord when it is scratched. 2. Not less than 0.5mm of the inner liner gauge remains.
Linear Air	Liners come out during high speed vehicles.		Depth- 0.5mm Area- 25MM 2 Length- 5mm Frequency- 3 location

No data-set was available for the tire images, that's why we created our own data-set using google images. We googled images of tires and used them as our data-set.

S.No.	Name of defect	Number of images
1	Normal	199
2	Bulges	62
3	Sidewall Cracking	97
4	Tread Wear Indicator	68
5	Linear Air	61
6	Exposed Cord	57

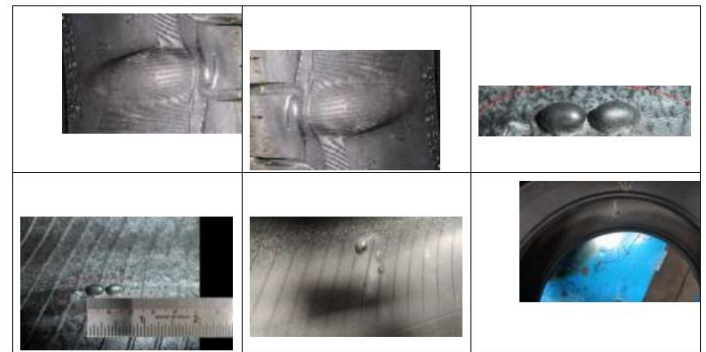
#### TIRE SIDEWALL CRACKING



#### NORMAL



#### LINEAR AIR



## TREAD



## TIRE BULGES



## V. PROPOSED MACHINE LEARNING MODEL

### Algorithms:

#### Gradient Descent:

Gradient descent is an optimization algorithm used to minimize some function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient. In machine learning, we use gradient descent to update the parameters of our model. Parameters refer to coefficients in Linear Regression and weights in neural networks. I have used gradient descent in my project to calculate the appropriate learning rate to boost the performance of the model.

#### Multiclass Classification:

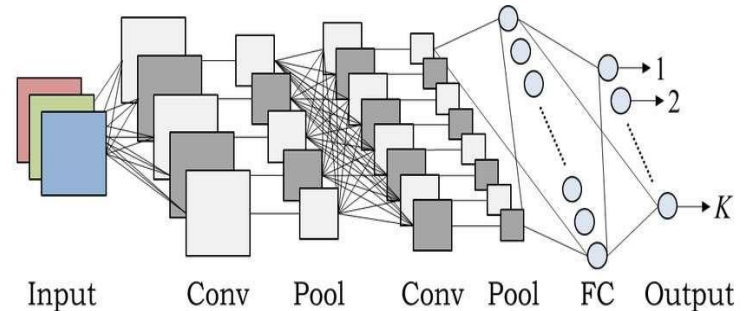
Classification problems having multiple classes with imbalanced dataset is known as multiclass classification. A classification task with more than two classes and where each sample is assigned to one and only one label is known as multiclass classification. An imbalanced dataset is a dataset where the data is not equally distributed among the classes. In this project we have more than one class of tire defects and the number of images in each class is also not equal. Thus we have used multiclass classification in the project.

#### CNN:

In neural networks, Convolutional neural network (ConvNets

or CNNs) is one of the main categories to do images recognition, images classifications. Objects detections, recognition faces etc., are some of the areas where CNNs are widely used.

CNN image classifications take an input image, process it and classify it under certain categories. Computers see an input image as an array of pixels and it depends on the image resolution. Based on the image resolution, it will see  $h \times w \times d$ . Technically, deep learning CNN models to train and test, each input image will pass it through a series of convolution layers with filters (Kernels), Pooling, fully connected layers (FC) and apply Softmax function to classify an object with probabilistic values between 0 and 1.



#### Softmax function:

At the end of a neural network classifier, we get a vector of “raw output values”: for example  $[-0.5, 1.2, -0.1, 2.4]$  But we’d like to convert these raw values into an understandable format: probabilities. We convert a classifier’s raw output values into probabilities using softmax function.

#### LIBRARY USED: Fastai

Fastai library is fast, robust, portable, and scalable. It is based on top of PyTorch. The fastai library allows you to train a Model on a certain DataBunch very easily by binding them together inside a Learner object. It provides tools to preprocess and group our data.

The vision module of the fastai library contains all the necessary functions to define a Dataset and train a model for computer vision tasks.

- vision.data
- vision.transform
- vision.learner
- vision.image

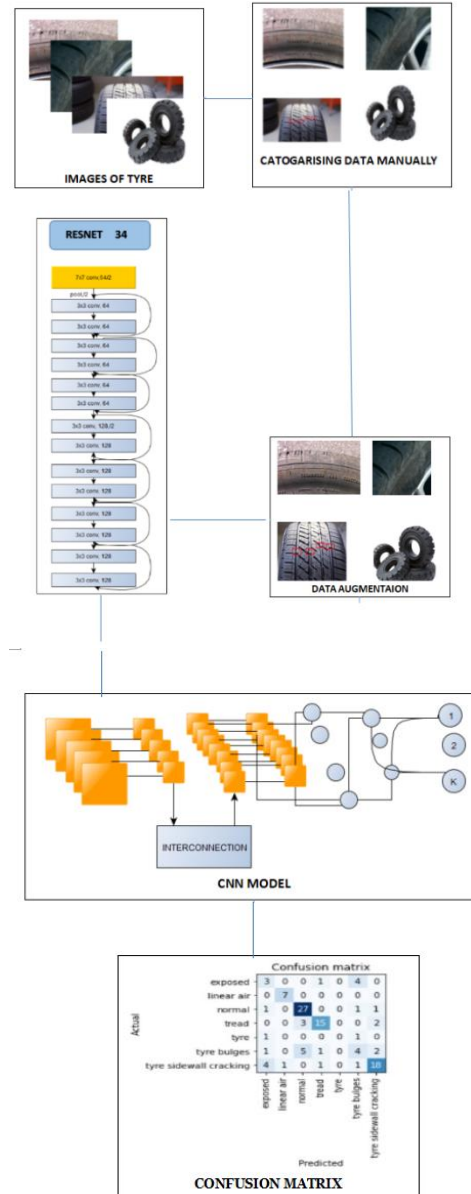
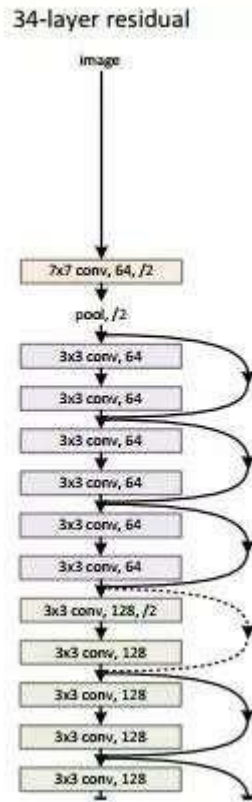
#### ARCHITECTURE USED: ResNet

In deep learning having a lot of hidden layers paves way to the vanishing / exploding gradients problem which actually makes the neural network unusable. ResNet takes care of the maximum learning of weights possible without over-fitting or negative deviation of accuracy through means of vanishing or exploding gradients problem. One does not have to worry about the number of hidden layers that should be used for the neural network architecture.

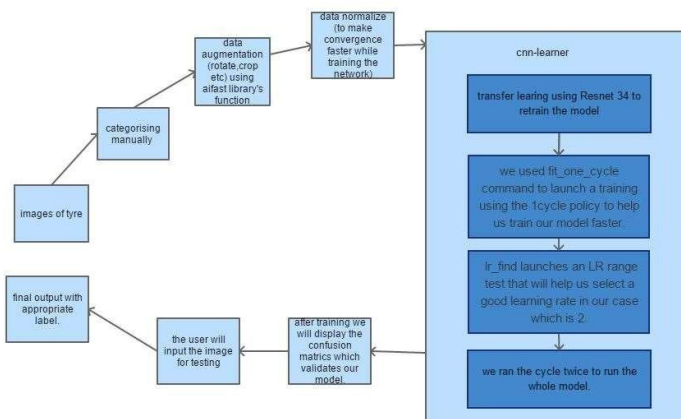
For our mode, we have used a pretrained model of Resnet 34 trained on ImageNet and have adapted it to our dataset. This was done to have a model that has already been trained to recognize a few features.



## VII. ARCHITECTURE DIAGRAM



## VI. FLOW DIAGRAM



## EXPLANATION

- 1)The image data has been collected and loaded in drive to make the dataset.
- 2)The image data has been then segregated manually followed by cleaning (i.e data augmentation) to optimize the image.
- 3)These cleaned image dataset is trained by resnet34 model followed by cnn training.(the architecture and explanation of

same is mentioned above)

4)The confusion metrics shows the Actual vs Predicted representation of the trained model.

5)Finally we input the tyre data and predict the defect in it through the trained data.

## VIII. PSEDOCODE

```
import fastai vision // the fastai library of pytorch
path = (set path location)
drive.mount('gdrive', force_remount=True)
directory path=(set path )
{
    classes = ['exposed','linear air','normal','tread','tyre bulldges','tyre sidewall cracking']
    for c in classes
        print (c)
        verify_images(path)// verify the image(data) present in folders mentioned
        data= imageDataBunch, from_folder(path) // loading the folder data in variable.

learn=cnn_learner(data, models.resnet34, metrics=error_rate )
{
    // cnn_learner used to classify image data and uses architecture resnet 34 .
    //other parameter, error rate matrix is also generated
    learn.fit_one_cycle(4) // launch a training using 1 cycle policy to train model faster
    learn.save()//to save the model using predefined save method
    learn.unfreeze() // unfreeze the other layer and train the complete network
    learn.lr_find// find the learning rate with minimum loss.
    learn.recorder.plot() //rating learning rate with loss
    learn.fit_one_cycle(4)// launch a training using 1 cycle policy to train model faster in a particular range
    learn.save()
    learn.load()//remembers the classes, the transforms you used or the normalization in the data, the model, its weight.
    test image= path
    learn=load_learner(path)
    pred_class(test image)// to predict the defect
}
}
```

## EXPLANATION

**Step 1:** Initially we are importing the fastai vision.Fastai is the python library used for deep learning applications that provides a single interface for text,tabular data,collaborative

filtering and time series.

**Step 2:** Specify the path of the image folder.

**Step 3:** Mount the root drive in order to get image.

**Step 4:** Define the classes based on the different type of defects listed in the drive.

**Step 5:** The images are verified to ensure that there are no corrupt image.

**Step 6:** ImageDataBunch.from\_folder is used to load the data in data variable as the datas follow the structured representation.

**Step 7:** Convolution neural network is use to classify the image data with resnet34 model in order to generate the error rate metrics.

**Step 8:** fit\_one\_cycle will launch a training using 1 cycle policy to train the model faster.

**Step 9:** The layers that are fully connected are considered trained and other layers can be unfreezed to train the complete network.

**Step 10:** lr\_find is use to find the perfect learning rate.

**Step 11:** Here we launch a training using one-cycle policy of fastai to train model faster using differential learning rates.

**Step 12:** Saving the trained image using predefined save model

**Step 13:** Now we are testing the input image with our trained model and to predict the defect.

## IX. CODE

```
tyre (2).ipynb
File Edit View Insert Runtime Tools Help All changes saved

[104] !curl -s https://course.fast.ai/setup/colab | bash
<>
[105] from google.colab import drive
drive.mount('gdrive', force_remount=True)
root_dir = "/gdrive/My Drive/"
base_dir = root_dir + 'fastai-v3/'
Mounted at gdrive

[106] !pip install "torch==1.4" "torchvision==0.5.0"
from fastai.vision import *
```

First google colab with fastai library has been setup, google drive (which contains the dataset) is mounted with google colab and path to the dataset has been set.

## EXPLORING DATA

```
[107] folder = 'normal'
      file = 'normal.csv'

[108] folder = 'tyre bulges'
      file = 'tyre bulges.csv'

[109] folder = 'tyre sidewall cracking'
      file = 'tyre sidewall cracking.csv'

[110] folder = 'exposed'
      file = 'exposed.csv'

[111] folder = 'linear air'
      file = 'linear air.csv'

[112] folder = 'tread'
      file = 'tread.csv'

[113] path = Path(base_dir + 'data/tyre')
      dest = path/folder
      dest.mkdir(parents=True, exist_ok=True)
```

```
[114] path.ls()

PosixPath('gdrive/My Drive/fastai-v3/data/tyre/tyre bulges'),
PosixPath('gdrive/My Drive/fastai-v3/data/tyre/tyre sidewall cracking'),
PosixPath('gdrive/My Drive/fastai-v3/data/tyre/exposed'),
PosixPath('gdrive/My Drive/fastai-v3/data/tyre/linear air'),
PosixPath('gdrive/My Drive/fastai-v3/data/tyre/normal'),
PosixPath('gdrive/My Drive/fastai-v3/data/tyre/models'),
PosixPath('gdrive/My Drive/fastai-v3/data/tyre/tyrebulges.jpg'),
PosixPath('gdrive/My Drive/fastai-v3/data/tyre/linearair.png'),
PosixPath('gdrive/My Drive/fastai-v3/data/tyre/export.pkl'),
PosixPath('gdrive/My Drive/fastai-v3/data/tyre/tread'),
PosixPath('gdrive/My Drive/fastai-v3/data/tyre/treadtyre bulges'),
PosixPath('gdrive/My Drive/fastai-v3/data/tyre/exposed.jpg')

[115] classes = ['exposed', 'linear air', 'normal', 'tread', 'tyre bulges', 'tyre sidewall cracking']

[ ] #download_images('gdrive/My Drive/fastai-v3/download.dms', dest, max_pics=20, max_workers=0)

[ ] # If you have problems download, try with 'max_workers=0' to see exceptions:
    #download_images(path/file, dest, max_pics=20, max_workers=0)

[116] for c in classes:
      print(c)
      verify_images(path/c, delete=True, max_size=500)

exposed
linear air
normal
tread
tyre bulges
tyre sidewall cracking
```

Then the folders of the different types of tyre defects along with the folder which contains images of normal tyres are specified. The classes of defects are exposed, linear air, normal, tread, tyre bulge, tyre sidewall and cracking. Then it is verified that these folders contains only images by using `verify_images`.

```
[117] np.random.seed(42)
      data = ImageDataBunch.from_folder(path, train=".", valid_pct=0.2,
      ds_tfms=get_transforms(), size=224, num_workers=4).normalize(imagenet_stats)

[118] data.classes

[ ] ['exposed',
     'linear air',
     'normal',
     'tread',
     'tyre',
     'tyre bulges',
     'tyre sidewall cracking']

[119] data.show_batch(rows=4, figsize=(7,8))

normal normal normal normal
tread tyre sidewall cracking normal tread
tyre bulges normal normal tread
```

Since the images are sorted by folder, it is convenient to use `ImageDataBunch` using the `from_folder` method. Basically, the `from_folder` method takes the following arguments:

**path** : Path to image dataset

**ds\_tfms** : Transformations that are applied on the images to augment the dataset

**size** : Size of images / Chosen value : 224 by 224 pixels

**valid\_pct** : 0.2 (i.e. 20% of the dataset will be used for validation since there aren't specific folders for training and validation sets).

Data is of type `ImageDataBunch` and has a `classes` attribute. We can see from the code above that our `DataBunch` successfully recognizes the three labels.

The `.normalize(imagenet_stats)` method above is used to normalize the dataset based on the stats of the RGB channels from the ImageNet dataset.

Data has a handy method called `show_batch` that can view a sample of the dataset.

## TRAINING THE MODEL

```
+ Code + Text

[119] tyre bulges normal normal tread
tyre bulges tread normal normal

[120] data.classes, data.c, len(data.train_ds), len(data.valid_ds)

[ ] ([ 'exposed',
      'linear air',
      'normal',
      'tread',
      'tyre',
      'tyre bulges',
      'tyre sidewall cracking'],
      7,
      419,
      104)

[121] learn = cnn_learner(data, models.resnet34, metrics=error_rate)
```

**data.classes** — What are the classes of tyres in our dataset?

**data.c** — How many classes are there in our dataset?



**len(data.train\_ds)** — What is the size of our training dataset?  
**len(data.valid\_ds)** — What is the size of our validation dataset?

A pre-trained ResNet34 Convolutional Neural Net model is used, and transfer learning to learn weights of only the last layer of the network is used.

With transfer learning, the module begins with an existing (trained) neural network used for image recognition — and then tweak it a bit (or more) here and there to train a model for the particular use case.

`cnn_learner` function, a built in fastai function that loads famous CNN (Convolutional Neural Network) Architectures has been used.

Basically, the functions needs the three following arguments :  
 The DataBunch

The specification of the model to be downloaded and trained (here we will use a resnet34)

And the metric (error rate is chosen here)

## EXPERIMENT

Number of cycles has been altered to check which one gives the minimum error rate.

1 epoch:

```
[187] learn = cnn_learner(data, models.resnet34, metrics=error_rate)

[191] learn.fit_one_cycle(1)
```

epoch	train_loss	valid_loss	error_rate	time
0	0.257483	1.355826	0.352381	00:09

2 epochs:

```
[187] learn = cnn_learner(data, models.resnet34, metrics=error_rate)

[188] learn.fit_one_cycle(2)
```

epoch	train_loss	valid_loss	error_rate	time
0	2.470074	1.512510	0.409524	00:10
1	1.769362	1.182127	0.323810	00:09

3 epochs:

```
[187] learn = cnn_learner(data, models.resnet34, metrics=error_rate)

[189] learn.fit_one_cycle(3)
```

epoch	train_loss	valid_loss	error_rate	time
0	1.081604	1.180590	0.333333	00:09
1	0.951487	1.341638	0.295238	00:09
2	0.811854	1.321103	0.295238	00:09

4 epochs:

```
[187] learn = cnn_learner(data, models.resnet34, metrics=error_rate)

[190] learn.fit_one_cycle(4)
```

epoch	train_loss	valid_loss	error_rate	time
0	0.612890	1.274266	0.276190	00:09
1	0.542131	1.262484	0.304762	00:09
2	0.522288	1.327481	0.342857	00:09
3	0.481612	1.318715	0.342857	00:09

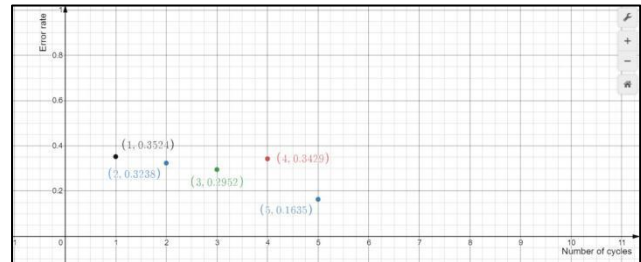
5 epochs:

```
[187] learn = cnn_learner(data, models.resnet34, metrics=error_rate)

[210] learn.fit_one_cycle(5)
```

epoch	train_loss	valid_loss	error_rate	time
0	0.114489	1.863574	0.187500	00:10
1	0.140928	2.283376	0.173077	00:10
2	0.147651	2.184931	0.168269	00:10
3	0.170799	2.061754	0.168269	00:10
4	0.162034	1.985363	0.163462	00:10

Graph (error rate vs no. Of epochs):



Since 5 epochs gave an error rate of 0.1722, which means 82.78% accuracy, it has been chosen.

```
[242] learn.save('stage-1')

[243] learn.unfreeze()

[244] learn.lr_find()

LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.

[126] learn.recorder.plot(suggestion=True)
```

Min numerical gradient: 3.31E-06  
 Min loss divided by 10: 1.00E-04

```
[127] min_grad_lr = learn.recorder.min_grad_lr
min_grad_lr

3.311311214825911e-06
```

The model is then saved and given the name (stage-1) One of

the most important parameters to tune in a deep learning model is the learning rate. Function `lr_find()` helped in picking the right learning rate for the model to learn with. Before `lr_find()` is executed, whole network is unfreezed. Unfreezing the network before running `lr_find()` gives better results. `unfreeze()` enables to update the weights of the entire network. `lr_find()` essentially trains the model with linearly increasing learning rates that usually range from around  $10e-7$  to 1. Once that is run, `learn.recorder.plot()` is used to plot the graph of the loss vs. the learning rate. Parameter `suggestion=True` gives ideal learning rate for the project (red dot). `learn.recorder.min_grad_lr` gives us the learning rate.

After finding the learning rate, number of cycles has been altered to check which one gives the minimum error rate.

1 epoch:

```
[200] learn.fit_one_cycle(1, min_grad_lr)
```

epoch	train_loss	valid_loss	error_rate	time
0	0.176220	1.302446	0.285714	00:10

2 epochs:

```
[199] learn.fit_one_cycle(2, min_grad_lr)
```

epoch	train_loss	valid_loss	error_rate	time
0	0.199305	1.313312	0.276190	00:10
1	0.188315	1.299252	0.285714	00:09

3 epochs:

```
[203] learn.fit_one_cycle(3, min_grad_lr)
```

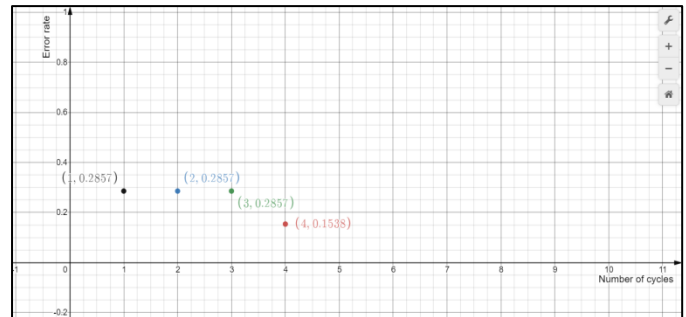
epoch	train_loss	valid_loss	error_rate	time
0	0.162654	1.289011	0.285714	00:10
1	0.195904	1.285603	0.276190	00:10
2	0.189262	1.286126	0.285714	00:10

4 epochs:

```
learn.fit_one_cycle(4, min_grad_lr)
```

epoch	train_loss	valid_loss	error_rate	time
0	0.144076	1.276402	0.192308	00:10
1	0.150964	1.274149	0.168269	00:10
2	0.134246	1.288029	0.163462	00:10
3	0.136792	1.286102	0.153846	00:10

Graph (error rate vs no. Of epochs)



The error rate obtained after running 4 cycles is 0.1695, which means 83.05% accuracy and results in an increased accuracy as compared to the previous epochs.

```
[129] learn.save('stage-2')
[130] learn.load('stage-2');
[131] interp = ClassificationInterpretation.from_learner(learn)
```

```
[132] interp.plot_confusion_matrix()
```

Confusion matrix

	exposed	linear air	normal	tread	tyre	tyre bulges	tyre sidewall cracking
Actual exposed	4	0	1	0	0	2	2
Actual linear air	0	8	0	0	0	0	0
Actual normal	1	0	29	1	0	3	0
Actual tread	0	0	1	11	0	1	2
Actual tyre	1	0	0	0	0	1	0
Actual tyre bulges	1	0	1	0	0	9	1
Actual tyre sidewall cracking	1	2	0	0	0	3	18

Confusion matrix is used to check where the model is giving erroneous predictions.

## OUTPUT

At last we will check if the model predicts the images correctly.

```
[133] learn.export()
[134] defaults.device = torch.device('cpu')
[139] img = open_image('gdrive/My Drive/fastai-v3/data/tyre/exposed.jpg')
img
```



```
[140] learn = load_learner(path)

[141] pred_class, pred_idx, outputs = learn.predict(img)
      if(pred_idx==1):
          print("Linear air")
      elif(pred_idx==0):
          print("Exposed")
      elif(pred_idx==2):
          print("Normal")
      elif(pred_idx==3):
          print("Tread")
      elif(pred_idx==5):
          print("Tyre Bulges")
      elif(pred_idx==6):
          print("Tyre Sidewall Cracking")
```

Exposed

```
[152] img = open_image('gdrive/My Drive/fastai-v3/data/tyre/tread.jpg')
      img
```



```
[153] learn = load_learner(path)
```

```
[154] pred_class, pred_idx, outputs = learn.predict(img)
      if(pred_idx==1):
          print("Linear air")
      elif(pred_idx==0):
          print("Exposed")
      elif(pred_idx==2):
          print("Normal")
      elif(pred_idx==3):
          print("Tread")
      elif(pred_idx==5):
          print("Tyre Bulges")
      elif(pred_idx==6):
          print("Tyre Sidewall Cracking")
```

Tread

```
[149] img = open_image('gdrive/My Drive/fastai-v3/data/tyre/sidewallcracking.jpg')
      img
```



```
[150] learn = load_learner(path)
```

```
[151] pred_class, pred_idx, outputs = learn.predict(img)
      if(pred_idx==1):
          print("Linear air")
      elif(pred_idx==0):
          print("Exposed")
      elif(pred_idx==2):
          print("Normal")
      elif(pred_idx==3):
          print("Tread")
      elif(pred_idx==5):
          print("Tyre Bulges")
      elif(pred_idx==6):
          print("Tyre Sidewall Cracking")
```

Tyre Sidewall Cracking

```
[158] img = open_image('gdrive/My Drive/fastai-v3/data/tyre/normal.jpg')
      img
```



```
[159] learn = load_learner(path)
```

```
[160] pred_class, pred_idx, outputs = learn.predict(img)
      if(pred_idx==1):
          print("Linear air")
      elif(pred_idx==0):
          print("Exposed")
      elif(pred_idx==2):
          print("Normal")
      elif(pred_idx==3):
          print("Tread")
      elif(pred_idx==5):
          print("Tyre Bulges")
      elif(pred_idx==6):
          print("Tyre Sidewall Cracking")
```

Normal

```
[146] img = open_image('gdrive/My Drive/fastai-v3/data/tyre/tyrebulges.jpg')
      img
```



```
[147] learn = load_learner(path)
```

```
[148] pred_class, pred_idx, outputs = learn.predict(img)
      if(pred_idx==1):
          print("Linear air")
      elif(pred_idx==0):
          print("Exposed")
      elif(pred_idx==2):
          print("Normal")
      elif(pred_idx==3):
          print("Tread")
      elif(pred_idx==5):
          print("Tyre Bulges")
      elif(pred_idx==6):
          print("Tyre Sidewall Cracking")
```

Tyre Bulges

```
[143] img = open_image('gdrive/My Drive/fastai-v3/data/tyre/linearair.png')
img
```



```
[144] learn = load_learner(path)

[145] pred_class, pred_idx, outputs = learn.predict(img)
if(pred_idx==1):
    print("Linear air")
elif(pred_idx==0):
    print("Exposed")
elif(pred_idx==2):
    print("Normal")
elif(pred_idx==3):
    print("Tread")
elif(pred_idx==5):
    print("Tyre Bulges")
elif(pred_idx==6):
    print("Tyre Sidewall Cracking")
```

Linear air

## X. RESULT

By taking the average of 4 epochs and learning rate of  $3.113 \times 10^{-6}$ , the model can predict the tyre defect with the accuracy of 83.05%. The model can detect the tyre defects from the objects mentioned in class and can classify them as exposed, linear air, tread, tyre bulges, normal and tyre sidewall cracking. Fast ai vision library is used to make the prediction fast. Cnn\_learner that is built in fastai function, has been used to load the CNN architecture. Specifically, Resnet 34 model has been trained. The process is explained with the help of flow chart and architecture diagram and the defects have been detected.

## XI. CONCLUSION

In this work. We propose a unified tyre defect detection system to find the defects such as Tread Wear Indicator, Bulges, Sidewall Cracking, Linear Air and Exposed Cord. Pre-trained models are used to save computation time and resources. Fast Ai library allows us to train models on certain DataBunch very easily. Finally, Resnet reserves the gradient that leads to better results. The results are shown on the website that has been deployed on python flask. Up to now, defect detection method has achieved the result for the five types of defects.

## XII. FUTURE GOALS

Our next goal is to include more images in our data set to get the accurate results. A website can be deployed for the same, where the user can upload the image of a tyre and get to know what is defect in the tyre. This makes it handy for common people to use it.

## XIII. REFERENCES

- [1] Q. Guo, C. Zhang, H. Liu and X. Zhang, "Defect Detection in tyre X- Ray Images using Weighted Texture Dissimilarity," *Journal of Sensors*, 2016.
- [2] Q. Zhu and X. Ai, "The Defect Detection Algorithm for Tire X-Ray Images Based on Deep Learning," in *IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)*, Chongqing, 2018.
- [3] X. Cui, Y. Liu and C. Wang, "Defect automatic detection for tire X-ray images using inverse transformation of principal component residual," in *Third International Conference on Artificial Intelligence and Pattern Recognition (AIPR)*, Lodz, 2016.
- [4] X. Zheng, J. Ding, Z. Pang and J. Li, "Detection of Impurity and Bubble Defects in Tire X-Ray Image Based on Improved Extremum Filter and Locally Adaptive-threshold Binaryzation," in *International Conference on Security, Pattern Analysis, and Cybernetics (SPAC)*, Jinan, 2018.
- [5] I. Fantini, L. Rittner, C. Yasuda and R. Lotufo, "Automatic detection of motion artifacts on MRI using Deep CNN," in *International Workshop on Pattern Recognition in Neuroimaging (PRNI)*, Singapore, 2018.
- [6] Y.-J. Cha, C. Wooram and O. Büyüköztürk, "Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks," *Computer-Aided Civil and Infrastructure Engineering*, vol. 32, no. 5, 2017.
- [7] P. Xi, C. Shu and R. Goubran, "Abnormality Detection in Mammography using Deep Convolutional Neural Networks," in *IEEE International Symposium on Medical Measurements and Applications (MeMeA)*, Rome, 2018.
- [8] J. C. Cheng and M. Chang, "Semantic Segmentation of Sewer Pipe Defects Using Deep Dilated Convolutional Neural Network," in *2019 Proceedings of the 36th ISARC*, Alberta, 2019.

