**AUTOMATIC CHALLAN GENERATION**

**CHAPTER 1**

**Introduction**

**1.1 Introduction:**

Most of cities and small regions the traffic Challan generation system is manual process. Automatic Challan Generation for traffic violation is a Deep Learning(CNN) based project that will automatically detect the violating vehicles and accurately punish them. This project is designed to reduce the work of traffic police officers so that they can focus on other violations like illegal parking, driving on the wrong side and drunk driving. This project will work in order to reduce the violations and make the city a better and a safe place for pedestrians and vehicles. The need for automation comes from the growing number of vehicles on the road every day. It has become an impossible task for traffic police officers to watch and control every road and every vehicle. It is up to the human beings to maintain discipline but in a densely populated country like India patience runs thin and forces the people to break the law.

In this Automatic Challan Generation, the methodology what we follow is at first we are going to train the model by giving some images which contain pictures of the persons who are wearing helmet and not wearing helmet. This Automatic Challan Generation model predicts whether a person going on road has kept the helmet or not. If a person is not wearing a helmet, then the system predicts the person and challan will be generated for that person. For this we have used Convolution Neural Networks model.

**1.2 Objective Of Research:**

The Automated Traffic Monitoring System is one of the effective tools for enforcement of traffic rules on Indian roads in a transparent manner. The system aims at harnessing strength of technology and minimise human intervention to bring about the speed and transparency in the whole process of traffic regulation which will go a long way in solving the problems of traffic on roads to a great extent. The Automated Traffic Monitoring System for enforcement of traffic rules has been in existence in one or the other forms in Western Countries for more than 50 years.

The project “Automated Traffic Monitoring System” is being proposed to bring substantial changes in the Traffic Enforcement system. The system has been conceived with following vision:

i)Will help in bringing more safety on roads.

1. Will result in reduction of rash and negligent driving.
2. Will avoids conflicts between police and public.
3. Will increase awareness of traffic rules and regulations.
4. Will reduce processing and disposal time of traffic violations . vi) Will bring transparency in enforcement of traffic laws and rules .

vii) Will be Used as effective tool of e-governance to manage, monitor and administrator. viii)To reduce rash and negligent driving by quality enforcement.

**1.3 Problem Statement:**

Statistics have shown time and again that a majority of road accidents occur due to negligence of traffic rules like underage driving, jumping signals, over speeding, drunken driving etc. These situations could have been easily controlled if traffic rules were laid down more effectively and enforced more strongly. In most of these cases the offenders tend to get away by bribing the traffic police and carry on repeat the same mistakes again and again. Also the overhead of printing and distributing DL and VRC this results in unnecessary use of paper which could easily be avoided if the whole process is digitized.

In this Automatic Challan Generation, the methodology what we follow is at first we are going to train the model by giving some images which contain pictures of the persons who are wearing helmet and not wearing helmet. This Automatic Challan Generation model predicts whether a person going on road has kept the helmet or not. If a person is not wearing a helmet, then the system predicts the person and challan will be generated for that person. For this we have used Convolution Neural Networks model.

**1.4 Industry Profile:**

Automatic Challan Generation mostly helpful for Traffic Control System. For Traffic Control System it helps to generate challan automatically instead of manually generating. This system will bring awareness of wearing helmets while driving. This also increase awareness of traffic rules and regulations and to reduce rash and negligent driving by quality enforcement.

# CHAPTER 2 Review of Literature

In the paper of Shiv KumarGoelDr. Manoj Kumar Shukla they discussed about the ever increasing rate of traffic rule violation and the e-penalty measures that could be adapted to strengthen the enforcement of road and traffic safety. All this gave us the motivation to design a government induced penalty system which will force the violator not to repeat their mistake again. The key area of this system is to identify the person who is not wearing the helmet and penalize them.

Earlier the traffic police officer has to generate the challan manually. They have to stay on roads hours and hours and then they have to identify the persons whether they have wear helmets or not. They have to focus not only to check whether the persons are wearing helmets or not but also they have to focus on illegal parking, driving on the wrong side and drunk driving .So, this had become burden to them inorder to reduce it we have implemented this Automatic Challan Generation system.

To design this Automatic Challan Generation we used a Deep Learning concept Convolution Neural Networks (CNN).

# CHAPTER 3 Data Collection

For this Automatic Challan Generation we used images to train the model. For this, we collected images from various sites and divided the images as training set and testing set. From the total images, 80% of the images are partitioned as training set and the remaining 20% images are partitioned as test set.

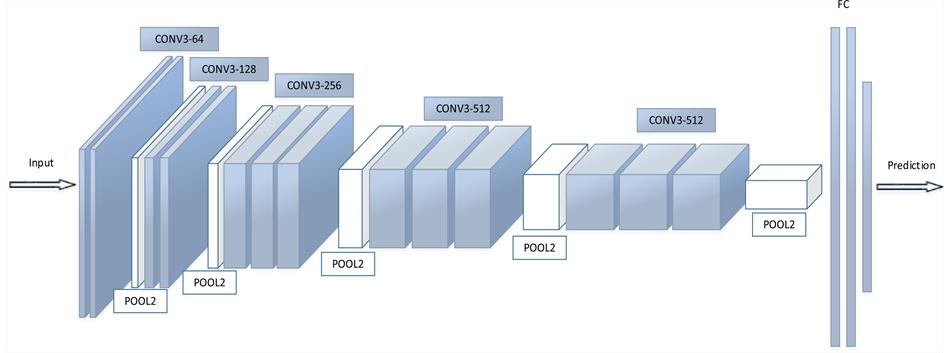
A convolution neural network consists of an input and an output layer as well as multiple hidden layers. The hidden layers of a CNN(Convolution Neural Networks) typically consists of convolution layers, RELU layer i.e, activation function, pooling layers, fully connected layers and normalization layers.

# CHAPTER-4 Methodology

**4.1 Exploratory Data Analysis:**

ConvNets are the superheroes that took working with images in deep learning to the next level. With ConvNets, the input is a image, or more specifically, a 3D Matrix.

Let’s start by looking at how a ConvNet looks!



## fig: Convolution Neural Network

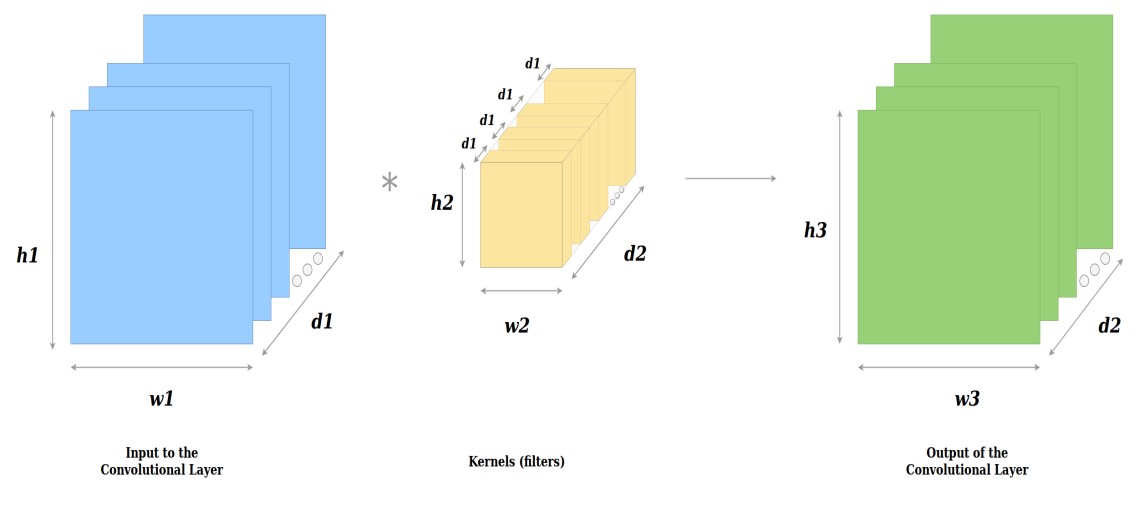
A ConvNet usually has 3 types of layers:

1. Convolution Layer(CONV)

2.Pooling Layer(POOL)

3.Fully Connected Layer(FC)

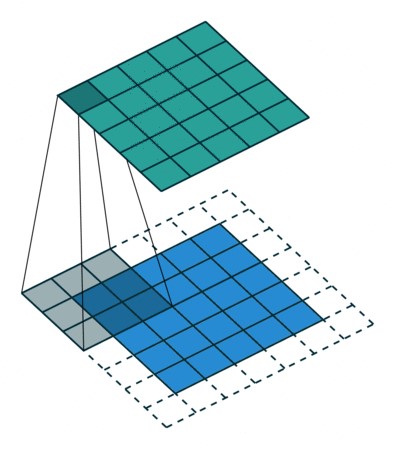
**1. Convolution Layer(CONV):**

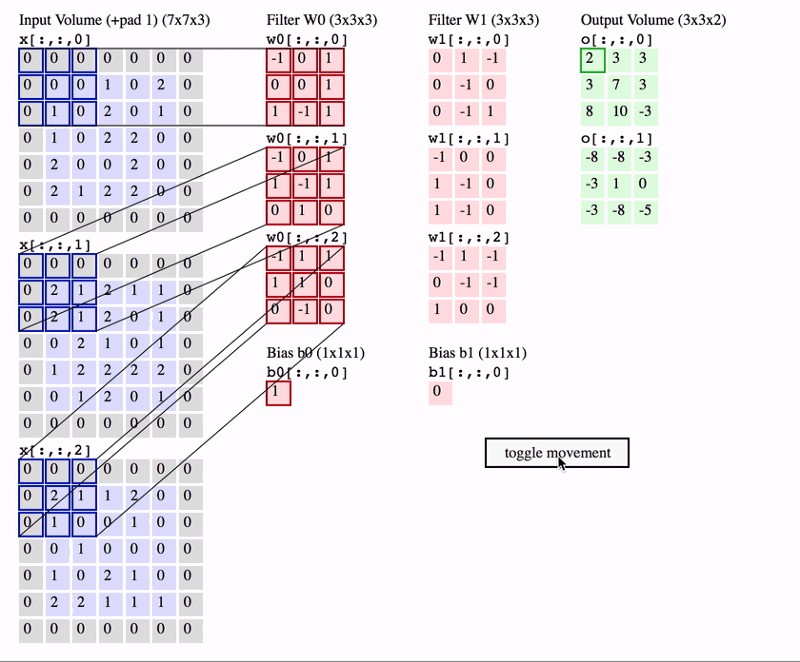


Let's throw light on some obvious things from above

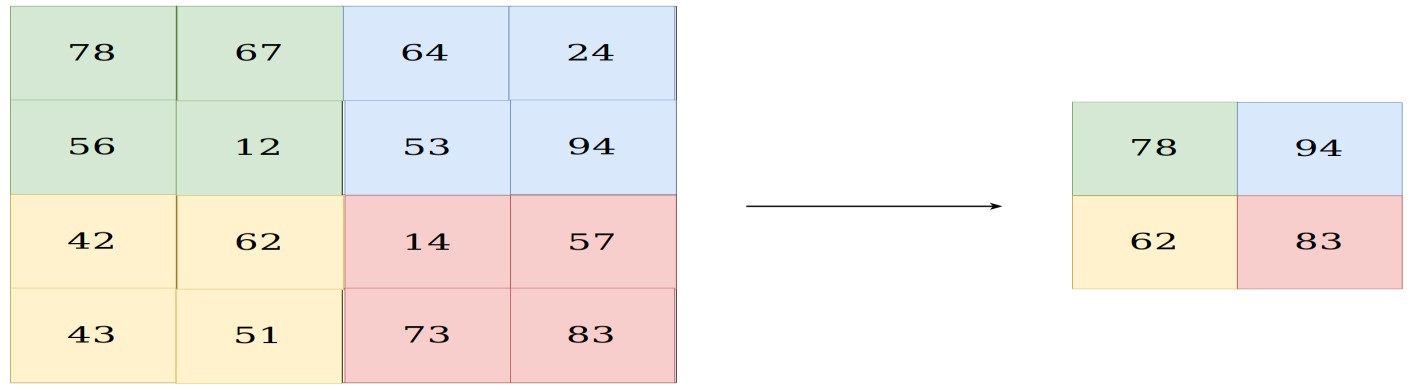
1. The depth (*d1*) (or the number of channels) of the input and of *one* kernel **is the same**.
2. The depth (*d2*) of the output **is equal** to the number of kernels (i.e. the depth of the orange 3-dimensional matrix).

Alright, so we have inputs, kernels and outputs. Now let’s look at what happens with a 2D input and a 2D kernel, i.e. *d1=1*.





## 2.Pooling Layer(POOL)



**fig: Max Pooling**

There are two types of pooling:

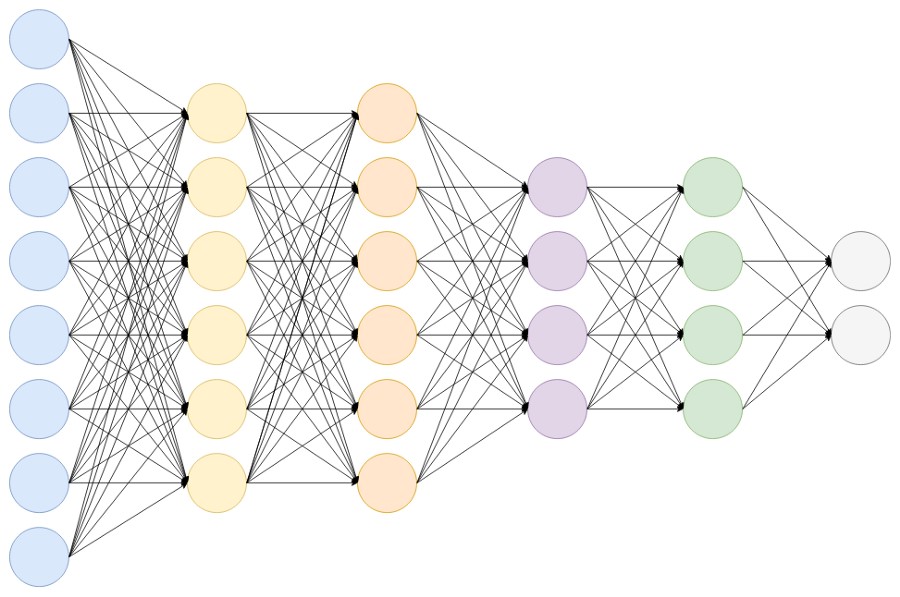
1. Max Pooling
2. Average Pooling

The main purpose of a pooling layer is to reduce the number of parameters of the input tensor and thus

* Helps reduce overfitting
* Extract representative features from the input tensor
* Reduces computation and thus aids efficiency

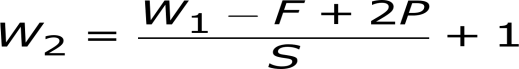
The input to the Pooling layer is tensor.

## 3.Fully Connected Layer(FC)



**fig: Fully Connected Network**

let’s visualize how to calculate the dimensions of the output tensor from the input tensor.



where,

W1—is the width / height of the input tensor

F—is the width / height of the kernel

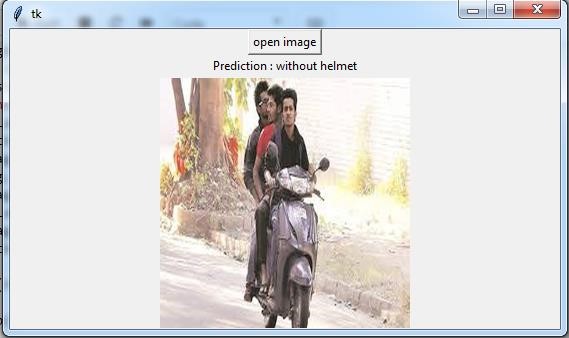
P—is the padding

S—is the stride

W2—is the output width / height

### 4.1.1 Figures





## 4.2 Statistical techniques and visualization

NumPy stands for ‘Numerical Python’ or ‘Numeric Python’. It is an open source module of Python which provides fast mathematical computation on arrays and matrices. Since, arrays and matrices are an essential part of the Machine Learning ecosystem, NumPy along with Machine Learning modules like Scikit-learn, Pandas, Matplotlib, TensorFlow, etc. complete the Python Machine Learning Ecosystem.

NumPy provides the essential multi-dimensional array-oriented computing functionalities designed for high-level mathematical functions and scientific computation. Numpy can be imported into the notebook using

NumPy’s main object is the homogeneous multidimensional array. It is a table with same type elements, i.e, integers or string or characters (homogeneous), usually integers. In NumPy, dimensions are called axes. The number of axes is called the rank.

There are several ways to create an array in NumPy like np.array, np.zeros, no.ones, etc. Each of them provides some flexibility.

**Keras:**

keras is a high-level neural networks API, capable of running on top of Tensorflow, Theneo and CNTK. It enables fast experimentation through a high level, user-friendly, modular and extensible API. Keras can also be run on both CPU and GPU.

Keras was developed and is maintained by [Francois Chollet](https://twitter.com/fchollet?ref_src=twsrc%5Egoogle%7Ctwcamp%5Eserp%7Ctwgr%5Eauthor) and is part of the Tensorflow core, which makes it Tensorflows preferred high-level API.

Keras can be installed using pip or conda

Keras provides [seven different datasets,](https://keras.io/datasets/) which can be loaded in using Keras directly. These include image datasets as well as a [house price](https://www.kaggle.com/vikrishnan/boston-house-prices) and a [movie review](https://www.kaggle.com/iarunava/imdb-movie-reviews-dataset) datasets.

To feed the images to a [convolutional neural network](http://cs231n.github.io/convolutional-networks/) we transform the dataframe to four dimensions. This can be done using numpys reshape method. We will also transform the data into floats and normalize it.

The easiest way of creating a model in Keras is by using the sequential API, which lets you stack one layer after the other. The problem with the sequential API is that it doesn’t allow models to have multiple inputs or outputs, which are needed for some problems.

Nevertheless, the sequential API is a perfect choice for most problems.

To create a [convolutional neural network](http://cs231n.github.io/convolutional-networks/) we only need to create a sequential object and use the add function to add layers. The code above first of creates a Sequential object and adds a few convolutional, maxpooling and dropout layers. It then flattens the output and passes it two a last dense and dropout layer before passing it to our output layer. If you aren’t confident build a convolutional neural network(CNN) check out [this great tutorial.](http://cs231n.github.io/convolutional-networks/)

The sequential API also supports another syntax where the layers are passed to the constructor directly. Alternatively, the functional API allows you to create the same models but offers you more flexibility at the cost of simplicity and readability.

It can be used with multiple input and output layers as well as shared layers, which enables you to build really complex network structures.

When using the functional API we always need to pass the previous layer to the current layer. It also requires the use of an input layer.

Before we can start training our model we need to configure the learning process. For this, we need to specify an optimizer, a loss function and optionally some metrics like accuracy.

The [loss function](https://www.youtube.com/watch?v=IVVVjBSk9N0) is a measure on how good our model is at achieving the given objective.

An [optimizer](https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f) is used to minimize the loss(objective) function by updating the weights using the gradients.

Augmentation is a process of creating more data from existing once. For images you can to little transformations like rotating the image, zooming into the image, adding noise and many more.

This helps to make the model more robust and solves the problem of having not enough data. Keras has a method called ImageDataGenerator which can be used for augmenting images.

This ImageDataGenerator will create new images that have been rotated, zoomed in or out, and shifted in width and height.

Now that we defined and compiled our model it’s ready for training. To train a model we would normally use the fit method but because we are using a datagenerator we will use fit\_generator and pass it our generator, X data, y data as well as the [number of epochs and the batch size.](https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9) We will also pass it a validation set so we can monitor the loss and accuracy on both sets as well as steps\_per\_epoch which is required when using a generator and is just set to the length of the training set divided by the batch\_size.

We can visualize our training and testing accuracy and loss for each epoch so we can get intuition about the performance of our model. The accuracy and loss over epochs are saved in the history variable we got whilst training and we will use Matplotlib to visualize this data.

**OpenCV** (*Open source computer vision*) is a library of programming functions mainly aimed at real-time computerized vision.Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel). The library is cross platform and free for use under the [open-source](https://en.wikipedia.org/wiki/Open-source_software) [BSD license.](https://en.wikipedia.org/wiki/BSD_license)

OpenCVsupportsthe [deeplearning](https://en.wikipedia.org/wiki/Deep_learning) frameworks [TensorFlow,](https://en.wikipedia.org/wiki/TensorFlow) [Torch](https://en.wikipedia.org/wiki/Torch_(machine_learning))[/PyTorch](https://en.wikipedia.org/wiki/PyTorch) and [Caffe.](https://en.wikipedia.org/wiki/Caffe_(software))

**4.3 Data Modeling and visualization**

## Step 1 — Collecting the Dataset

In order to train our machine, we need a huge amount of data so that our model can learn from them by identifying out certain relations and common features related to the objects.

Fortunately many such datasets are available on internet. Here is a link for the helmets and without helmets dataset which consist of 500 images — 250 of each. This will help in training as well testing our classifier.

## Step 2 — Importing Libraries and Splitting the Dataset

To use the powers of the libraries, we first need to import them.

After importing the libraries, we need to split our data into two parts- taining\_set and test\_set.

In our case, the dataset is already split into two parts. The training set has 400 images each of helmets and without helmets while the test set has 50 images of each.

## Step 3 — Building the CNN

This is most important step for our network. It consists of three parts -

1. Convolution
2. Polling
3. Flattening

## Step 4 — Full Connection

Full connection is connecting our convolutional network to a neural network and then compiling our network.

Here we have made 2 layer neural network with a sigmoid function as an activation function for the last layer as we need to find the probability of the object being a helmet or a without helmet.

## Step 5 — Data Augmentation

While training your data, you need a lot of data to train upon. Suppose we have a limited number of images for our network. What to do now?

You don’t need to hunt for novel new images that can be added to your dataset.

Why? Because, neural networks aren’t smart to begin with. For instance, a poorly trained neural network would think that these three tennis balls shown below, are distinct, unique images.

So, to get more data, we just need to make minor alterations to our existing dataset. Minor changes such as flips or translations or rotations. Our neural network would think these are distinct images anyway.

Data augmentation is a way we can reduce overfitting on models, where we increase the amount of trainingdata using information only in our training data. The field of data augmentation is not new, and in fact, various data augmentation techniques have been applied to specific problems.

## Step 6 — Training our Network

So, we completed all the steps of construction and its time to train our model.If you are training with a good video card with enough RAM (like an Nvidia GeForce GTX 980 Ti or better), this will be done in less than an hour. If you are training with a normal cpu, it might take a lot longer.With increasing number of epochs, the accuracy will increase.

## Step 7 — Testing

Now lets test a random image.

And, yes !!our network correctly predicted the image of the dog!! Though it is not 100% accurate but it will give correct predictions most of the times. Try adding more convolutional and pooling layers, play with the number of nodes and epochs, and you might get high accuracy result.

# CHAPTER 5 Findings and Suggestions

<https://pixabay.com/photos/motorcycle-rider-bike-helmet-18056/>

[https://www.kickstarter.com/projects/coros/coros-linx-smart-cycling-helmet-safelytune- in-to](https://www.kickstarter.com/projects/coros/coros-linx-smart-cycling-helmet-safely-tune-in-to) <https://www.pakwheels.com/blog/wear-a-helmet-while-riding-bike/>

[https://www.deccanchronicle.com/nation/current-affairs/180316/traffic-police-on-thelookout-for-two-wheeler-riders-wearing-low-quality-headgear.html](https://www.deccanchronicle.com/nation/current-affairs/180316/traffic-police-on-the-lookout-for-two-wheeler-riders-wearing-low-quality-headgear.html)

# CHAPTER 6 Conclusion

Finally, we conclude that by using Automatic Challan Generation we can reduce the efforts of traffic police and help them to focus on other violations like drunk driving, lane cutting, over speeding etc.