**Capstone Project Proposal Ananthesh J Shet**

Machine Learning Engineer Nanodegree 31-Oct-2017

**Object Classification Using Convolutional Neural Network For Reverse Parking**

1. **Overview**

Reverse parking accidents are very common and involve damaging other vehicles and injuring/killing people. According to National Highway Traffic Safety Administration(NHTSA)[[1]](#footnote-1), over a period of 3 years(2012-2014), an average of 1,898 people were killed in non-traffic crashes. Nonoccupants(such as pedestrians and bicyclists) accounted for 34% of these people – 42% of whom were killed by vehicles moving forward and 35% by vehicles backing up. An average of 92,000 people were injured in non-traffic crashes. Nonoccupants accounted for 33% of these people – 49% of whom were injured by vehicles moving forward and 40% by vehicles backing up.

Today’s vehicle are fitted with rear-view cameras to assist the driver with reverse parking. There are backup aid systems too which provide visual and audible warning whenever the system detects an object behind the vehicle. These information are only as useful as the attentiveness of the driver as it is ultimately the driver who as to take corrective measures such as braking.

The system can be further improved by letting the system to automatically take corrective measures if the driver doesn’t respond within a set time interval from the time an object has been detected. This will greatly help in reducing the number of accidents during backing up.

1. **Problem Statement**

For the system to take the right corrective measures at the right time, the system has to correctly identify/classify the objects. If the system thinks that there is an object when in reality there is not, then it is termed as False Positive(FP). If the system thinks there is no object when in reality there is, then it is termed as False Negative(FN). System with high FP leads to unnecessary braking causing discomfort to the driver/passengers. System with high FN leads to more number of accidents. This project aims at designing a classification system with low FP and FN.

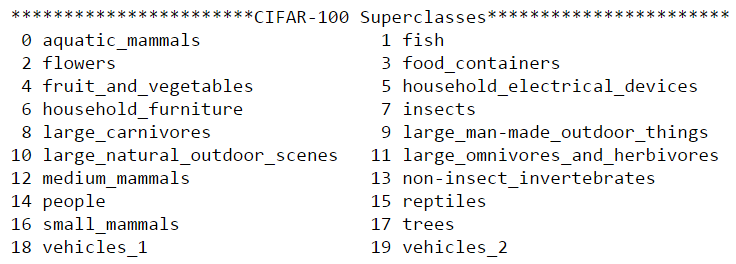
1. **Dataset**

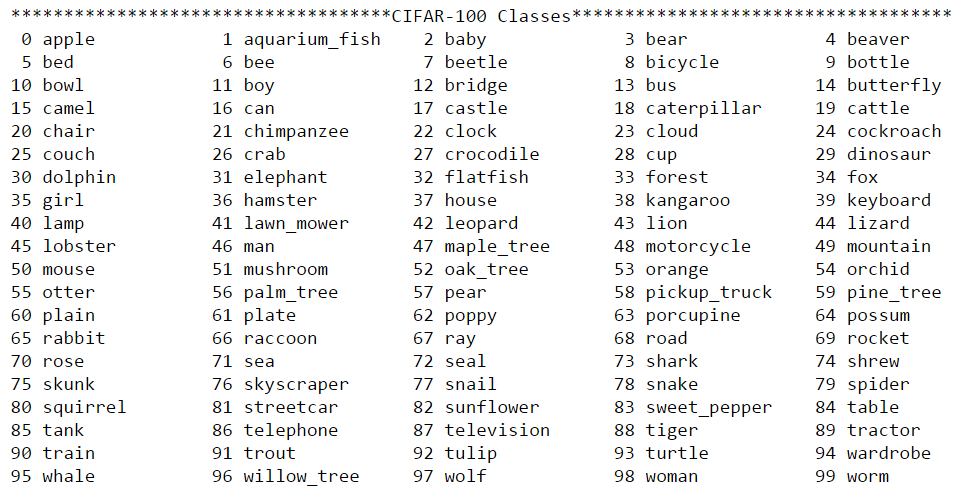
For this project, I will use the publicly available dataset from CIFAR-100[[2]](#footnote-2). The CIFAR-100 dataset consists of 60000 32x32 color images in 100 classes with 600 images per class. There are 500 training images and 100 testing images per class. The 100 classes in the CIFAR-100 are grouped into 20 superclasses. Each image comes with a "fine" label (the class to which it belongs) and a "coarse" label (the super class to which it belongs).

Here is the list of classes in the CIFAR-100:

|  |  |  |
| --- | --- | --- |
| **#** | **Superclass** | **Classes** |
| 0 | aquatic mammals | beaver, dolphin, otter, seal, whale |
| 1 | fish | aquarium fish, flatfish, ray, shark, trout |
| 2 | flowers | orchids, poppies, roses, sunflowers, tulips |
| 3 | food containers | bottles, bowls, cans, cups, plates |
| 4 | fruit and vegetables | apples, mushrooms, oranges, pears, sweet peppers |
| 5 | household electrical devices | clock, computer keyboard, lamp, telephone, television |
| 6 | household furniture | bed, chair, couch, table, wardrobe |
| 7 | insects | bee, beetle, butterfly, caterpillar, cockroach |
| 8 | large carnivores | bear, leopard, lion, tiger, wolf |
| 9 | large man-made outdoor things | bridge, castle, house, road, skyscraper |
| 10 | large natural outdoor scenes | cloud, forest, mountain, plain, sea |
| 11 | large omnivores and herbivores | camel, cattle, chimpanzee, elephant, kangaroo |
| 12 | medium-sized mammals | fox, porcupine, possum, raccoon, skunk |
| 13 | non-insect invertebrates | crab, lobster, snail, spider, worm |
| 14 | people | baby, boy, girl, man, woman |
| 15 | reptiles | crocodile, dinosaur, lizard, snake, turtle |
| 16 | small mammals | hamster, mouse, rabbit, shrew, squirrel |
| 17 | trees | maple, oak, palm, pine, willow |
| 18 | vehicles 1 | bicycle, bus, motorcycle, pickup truck, train |
| 19 | vehicles 2 | lawn-mower, rocket, streetcar, tank, tractor |

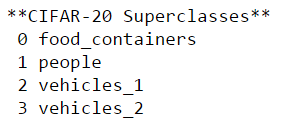
The superclasses and classes are numbered in alphabetical order as shown below:

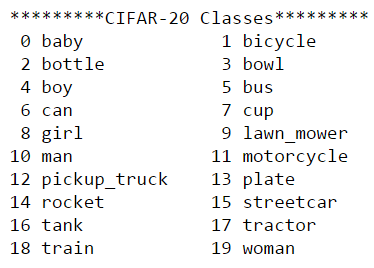




For my project I will not require all the super classes. The most probable super classes to be found in parking lots/roads and relevant for my project are 3, 14, 18 and 19. I created a sub dataset, namely CIFAR-20 with the selected super classes.

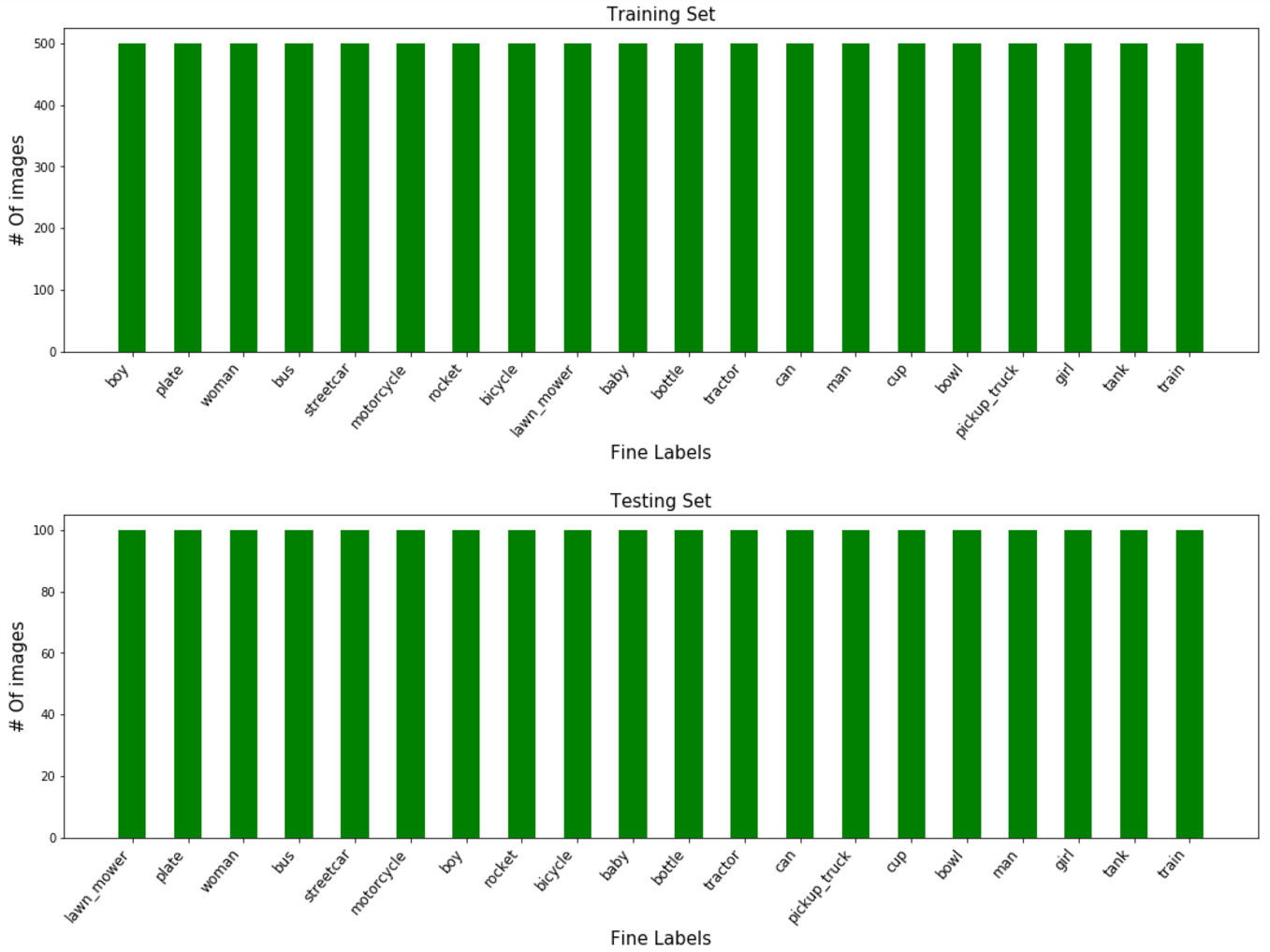
The superclasses and classes in CIFAR-20 are as shown below:





* 1. **Data Exploration**

Below is the distribution of each fine label in the CIFAR-20 dataset



It can be seen that only the fine labels of the selected super classes are present in the CIFAR-20 dataset. There are 500 images of each class in the training set and 100 images of each class in testing set.

* 1. **Data Visualization**

Below are few images from the CIFAR-20 dataset



1. **Solution Statement**

The goal of the project is to design a Convolutional Neural Network(CNN) to classify objects. Convolutional networks were inspired by biological processes in which the connectivity pattern between neurons is inspired by the organization of the animal visual cortex[[3]](#footnote-3). CNNs require very little pre-processing compared to other image classification algorithms. This implies that the network learns the filters which in traditional algorithms have to be hand-designed.

Below is an image of a typical CNN:



A typical CNN consists of an input, an output and multiple hidden layers. The hidden layers are either convolutional, pooling/subsampling or fully connected. Convolutional layers apply a convolution operation to the input, passing the result to the next layer. Pooling layers combine the outputs of neuron clusters at one layer into a single neuron in the next layer. Fully connected layers connect every neuron in one layer to every neuron in another layer.

The project aims at developing a simple CNN model involving mostly of convolutional layers. Simple model implies less memory, less training time and faster execution.

1. **Benchmark Model**

I will use the All-CNN[[4]](#footnote-4) model, with modification to last layer, as benchmark. All-CNN is convolutional neural network with focus on simple architecture using only convolutions and subsampling. It claims to match or even slightly outperform the state of the art on CIFAR-10 and CIFAR-100.

The network, with modification in the last layer to suite my CIFAR-20 dataset, is as below:

|  |
| --- |
| **Input 32x32 RGB image** |
| **3x3 conv. 96 ReLU** |
| **3x3 conv. 96 ReLU** |
| **3x3 conv. 96 ReLU with stride r =2** |
| **3x3 conv. 192 ReLU** |
| **3x3 conv. 192 ReLU** |
| **3x3 conv. 192 ReLU with stride r = 2** |
| **3x3 conv. 192 ReLU** |
| **1x1 conv. 192 ReLU** |
| **1x1 conv. 10 ReLU** |
| **global averaging over 6 × 6 spatial dimensions** |
| **20-way softmax** |

1. **Evaluation Metric**

As shown in the Dataset section, every image has got a class number, e.g. class “boy” has got class number 4. The class numbers will be one-hot encoded to get a vector of length 20, namely y\_true. Given an image as input, the CNN outputs a “prediction” class, namely y\_pred, predicting the class the image belongs to. As the dataset is uniformly distributed across all the 20 classes, I will use “accuracy” as my evaluation metric.

accuracy = mean(equal(argmax(y\_true), argmax(y\_pred))

argmax() returns the index of vector element having maximum value

equal() returns a boolean vector

mean() return the mean of a vector

I will also make use of confusion matrix to get a feel of FP and FN numbers.

1. **Project Design**

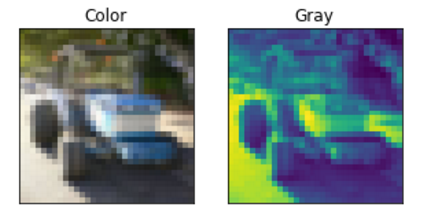
The CIFAR-20 dataset contains training and testing data. The training set will be split into training and validation set. The training set will be used to train the model, i.e. modify its weights, and validation set will be used to check how the model is doing. As the validation set is not used to modify the model’s weights, it also gives a good indication if the model is overfitting to the training set. The testing set will be used to find the final accuracy of the model. The testing set acts like real world data as the model never saw the testing set.

Testing

Validation

Training

As part of preprocessing, I will convert each color image into grayscale. As the goal of the project is to classify objects, the shape of the object plays an important role than its color. Converting to grayscale also reduces the input size helping the model to train faster.



I will also normalize the images as part of preprocessing. Images of same class objects taken in different lighting condition will lead to a huge difference in pixel values but I would like my model to classify both these images as same class. Normalizing the images will make sure that all the images have pixel values in similar range. I will normalize the images to be in the range -0.5 to 0.5.

I will check the accuracy of the benchmark model, All-CNN, on CIFAR-20 dataset.

Moving on to designing my model, I will use Keras[[5]](#footnote-5) and start with a simple architecture of one convolutional layer(CONV) and one fully connected layer(FC) later adding more simple layers(CONV, FC) based on the accuracy of the model. If my model is overfitting or not able to get a better accuracy or match the accuracy of the benchmark model, then I will consider including other layers such as pooling, dropout.

Based on the accuracy, I will decide to augment data in the preprocessing step. Considering my project, I would use “rotation” as my augmentation criterion.

1. <https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812311> [↑](#footnote-ref-1)
2. <https://www.cs.toronto.edu/~kriz/cifar.html> [↑](#footnote-ref-2)
3. <https://en.wikipedia.org/wiki/Convolutional_neural_network> [↑](#footnote-ref-3)
4. <https://arxiv.org/pdf/1412.6806.pdf> [↑](#footnote-ref-4)
5. <https://keras.io/> [↑](#footnote-ref-5)