

# Object Classification Using Convolutional Neural Network

## 1. Overview

There are a lot of images and videos out there in the wild and they are used in training a lot of applications<sup>1</sup> like Autonomous Cars, Gesture Recognition, Optical Character Recognition, Face Recognition, Image Search, Remote Sensing, Machine Vision, Robots etc. Most of these applications would first want to recognize the objects in a given image and take further steps based on the objects present in the image. Thus classifying objects correctly plays a crucial role as it might lead to good or bad decisions.

The classifying algorithm takes a set of features characterizing an object and uses them to determine the object class. Some algorithms need the user to specify the features whereas some algorithms such as Neural Networks, learn the features themselves. Convolutional Neural Networks learn the basic parts of the objects separately and understand the final object using multiple convolutional layers.

## 2. Problem Statement

For a system to take the right decision at the right time, the system has to correctly classify the objects. If the system recognizes an object when it is not present, then it is termed as False Positive(FP). If the system doesn't recognize the object when it is really present, then it is termed as False Negative(FN). System with high FP leads to unintended actions. System with high FN leads to missed actions. For e.g., in Autonomous Cars application, classifying pedestrians as stationary objects may lead to accidents. This project aims at designing a classification system using simple architecture CNN with low FP and FN.

## 3. Dataset

For this project, I will use the publicly available dataset from CIFAR-100<sup>2</sup>. 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 super classes. Each image comes with a "fine" label (the class to which it belongs) and a "coarse" label (the super class to which it belongs).

---

<sup>1</sup> [https://en.wikipedia.org/wiki/Category:Applications\\_of\\_computer\\_vision](https://en.wikipedia.org/wiki/Category:Applications_of_computer_vision)

<sup>2</sup> <https://www.cs.toronto.edu/~kriz/cifar.html>

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:

```
*****CIFAR-100 Superclasses*****
0 aquatic_mammals
1 fish
2 flowers
3 food_containers
4 fruit_and_vegetables
5 household_electrical_devices
6 household_furniture
7 insects
8 large_carnivores
9 large_man-made_outdoor_things
10 large_natural_outdoor_scenes
11 large_omnivores_and_herbivores
12 medium_mammals
13 non-insect_invertebrates
14 people
15 reptiles
16 small_mammals
17 trees
18 vehicles_1
19 vehicles_2
```

\*\*\*\*\*CIFAR-100 Classes\*\*\*\*\*

|             |                 |               |                 |              |
|-------------|-----------------|---------------|-----------------|--------------|
| 0 apple     | 1 aquarium_fish | 2 baby        | 3 bear          | 4 beaver     |
| 5 bed       | 6 bee           | 7 beetle      | 8 bicycle       | 9 bottle     |
| 10 bowl     | 11 boy          | 12 bridge     | 13 bus          | 14 butterfly |
| 15 camel    | 16 can          | 17 castle     | 18 caterpillar  | 19 cattle    |
| 20 chair    | 21 chimpanzee   | 22 clock      | 23 cloud        | 24 cockroach |
| 25 couch    | 26 crab         | 27 crocodile  | 28 cup          | 29 dinosaur  |
| 30 dolphin  | 31 elephant     | 32 flatfish   | 33 forest       | 34 fox       |
| 35 girl     | 36 hamster      | 37 house      | 38 kangaroo     | 39 keyboard  |
| 40 lamp     | 41 lawn_mower   | 42 leopard    | 43 lion         | 44 lizard    |
| 45 lobster  | 46 man          | 47 maple_tree | 48 motorcycle   | 49 mountain  |
| 50 mouse    | 51 mushroom     | 52 oak_tree   | 53 orange       | 54 orchid    |
| 55 otter    | 56 palm_tree    | 57 pear       | 58 pickup_truck | 59 pine_tree |
| 60 plain    | 61 plate        | 62 poppy      | 63 porcupine    | 64 possum    |
| 65 rabbit   | 66 raccoon      | 67 ray        | 68 road         | 69 rocket    |
| 70 rose     | 71 sea          | 72 seal       | 73 shark        | 74 shrew     |
| 75 skunk    | 76 skyscraper   | 77 snail      | 78 snake        | 79 spider    |
| 80 squirrel | 81 streetcar    | 82 sunflower  | 83 sweet_pepper | 84 table     |
| 85 tank     | 86 telephone    | 87 television | 88 tiger        | 89 tractor   |
| 90 train    | 91 trout        | 92 tulip      | 93 turtle       | 94 wardrobe  |
| 95 whale    | 96 willow_tree  | 97 wolf       | 98 woman        | 99 worm      |

For my project I will not use all the super classes. To keep the computational time less than CIFAR-100 and more challenging than CIFAR-10, I will create a sub dataset, namely CIFAR-20 with the reduced super classes. I have selected the below super classes by considering Autonomous Cars as an example application.

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

**\*\*CIFAR-20 Superclasses\*\***

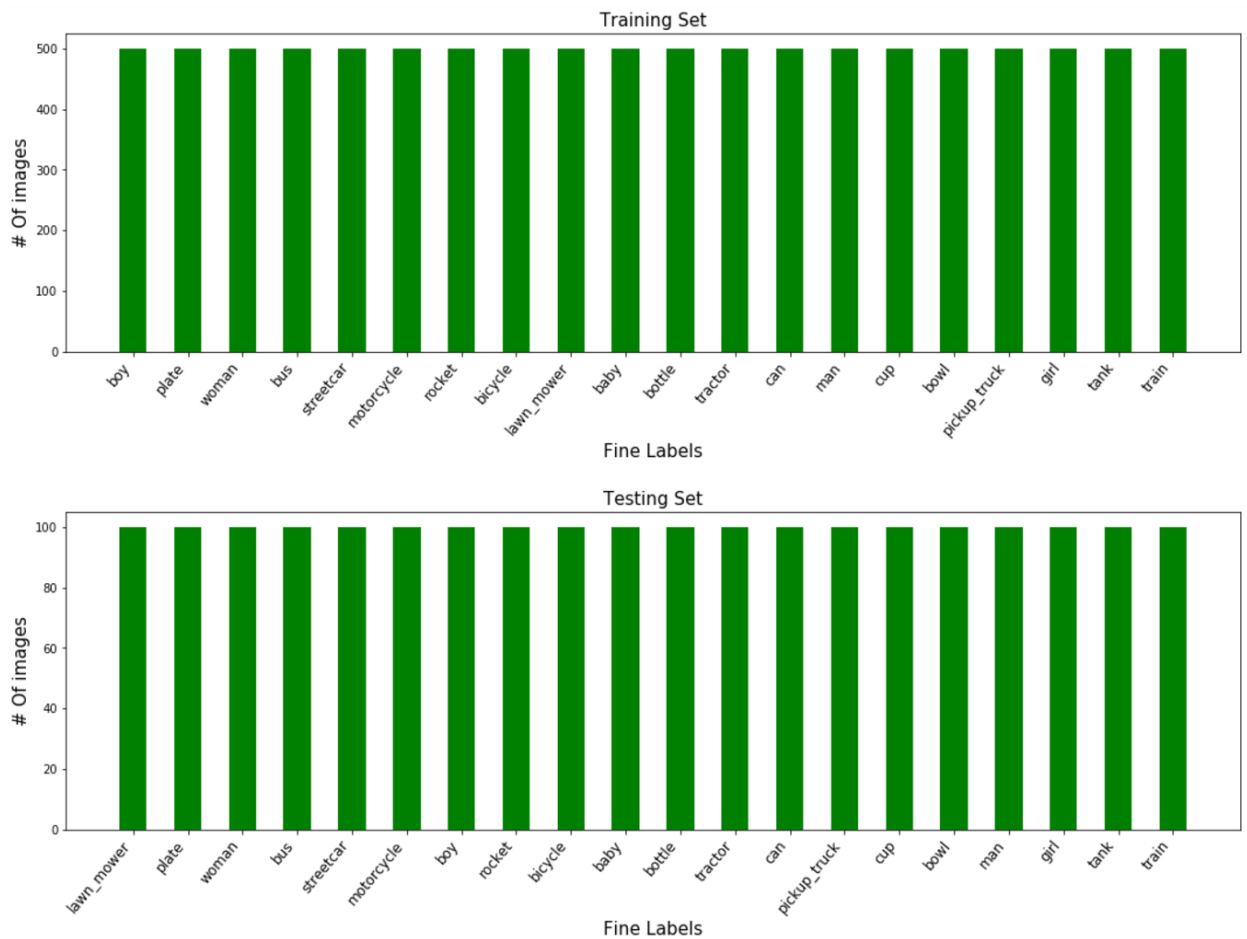
- 0 food\_containers
- 1 people
- 2 vehicles\_1
- 3 vehicles\_2

# \*\*\*\*\*CIFAR-20 Classes\*\*\*\*\*

|                 |               |
|-----------------|---------------|
| 0 baby          | 1 bicycle     |
| 2 bottle        | 3 bowl        |
| 4 boy           | 5 bus         |
| 6 can           | 7 cup         |
| 8 girl          | 9 lawn_mower  |
| 10 man          | 11 motorcycle |
| 12 pickup_truck | 13 plate      |
| 14 rocket       | 15 streetcar  |
| 16 tank         | 17 tractor    |
| 18 train        | 19 woman      |

## 3.1. 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.

### 3.2. Data Visualization

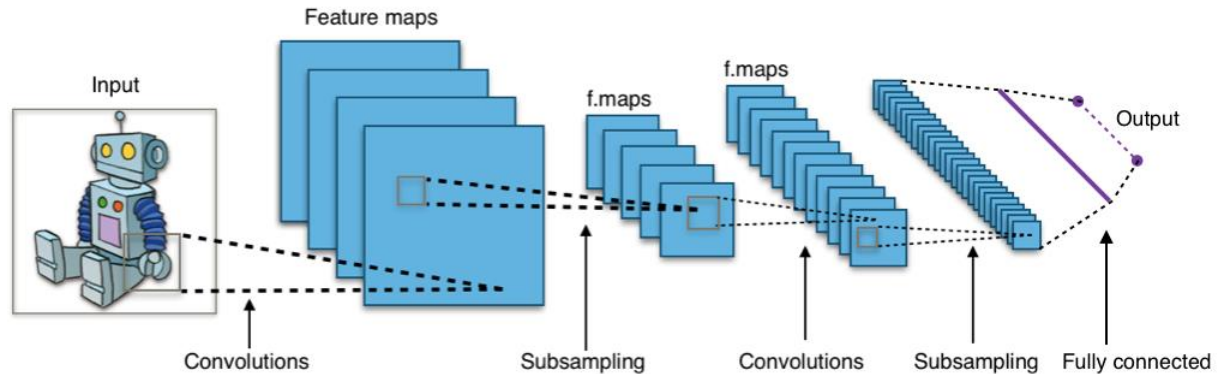
Below are few images from the CIFAR-20 dataset



## 4. Solution Statement

The goal of the project is to design a 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<sup>3</sup>. 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.

## 5. Benchmark Model

I will use Support Vector Machine(SVM)<sup>4</sup> with Radial Basis Function(RBF)<sup>5</sup> kernel as my benchmark model. SVMs are supervised learning methods used for classification, regression and outliers detection. For obtaining non-linear decision boundaries, SVMs use a kernel function. Usually non-linear kernels, such as RBF, yield better performance for classification.

## 6. 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

<sup>3</sup> [https://en.wikipedia.org/wiki/Convolutional\\_neural\\_network](https://en.wikipedia.org/wiki/Convolutional_neural_network)

<sup>4</sup> [https://en.wikipedia.org/wiki/Support\\_vector\\_machine](https://en.wikipedia.org/wiki/Support_vector_machine)

<sup>5</sup> [https://en.wikipedia.org/wiki/Radial\\_basis\\_function\\_kernel](https://en.wikipedia.org/wiki/Radial_basis_function_kernel)

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.

## 7. Project Design

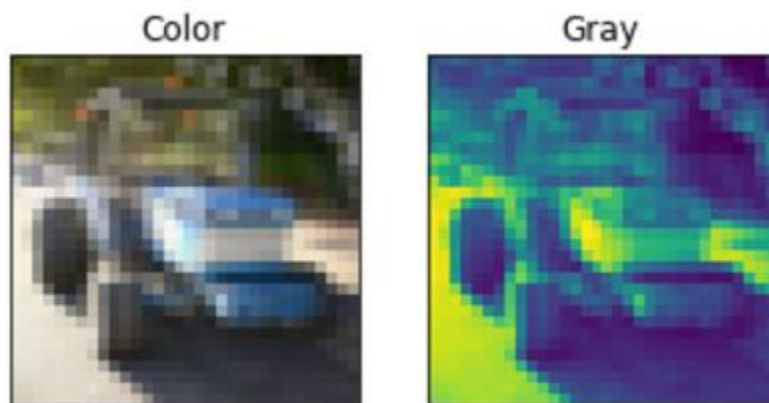
The CIFAR-20 dataset contains training and testing data. The training set will be split into training and validation set. The training set will be used to train the model, i.e. modify its weights, and validation set will be used to check how the model is doing. As the validation set is not used to modify 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.

Training

Validation

Testing

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.

At this stage I will check the accuracy of the benchmark model on CIFAR-20 dataset.

Moving on to designing my model, I will use Keras<sup>6</sup> 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.

---

<sup>6</sup> <https://keras.io/>