**Capstone Project Ananthesh J Shet**

Machine Learning Engineer Nanodegree 18-Dec-2017

**Object Classification Using Convolutional Neural Network for Image Search**

## I. Definition

### Project Overview

There are a lot of images and videos out there in the wild and they can be used for a wide variety of applications. Users search for images for entertainment, academics, presentations, projects etc. Users also need a large number of images belonging to specific category for training applications[[1]](#footnote-1) like Autonomous Cars, Gesture Recognition, Optical Character Recognition, Face Recognition, Remote Sensing, Machine Vision, Robots etc. Users search for these images on the net and it is important to get the right images for the right applications. For a good user experience and for faster search, search engines have to provide a bunch of relevant images based on certain keywords in less time.

Relevant images are those images which have in them the object(s) mentioned in the keyword(s). For e.g., if the user searches for images by entering keyword “car”, then any image having car in it, maybe among other objects, is a relevant image. So the search engine has to first classify objects in the images and return those images containing objects of interest.

### Problem Statement

Classifying objects in an image is very important for a successful image search. Classifying objects incorrectly will lead to unwanted images in the search result and a frustrated user! A general 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 can learn the features themselves.

A search engine fails if the classifying algorithm performs poorly. If the algorithm recognizes an object when it is not present, then it is termed as False Positive(FP). If the algorithm doesn’t recognize the object when it is really present, then it is termed as False Negative(FN). High FP or FN indicate a poorly performing algorithm. This project aims at designing a classification system using simple architecture Convolutional Neural Network(CNN) with low FP and FN.

## II. Analysis

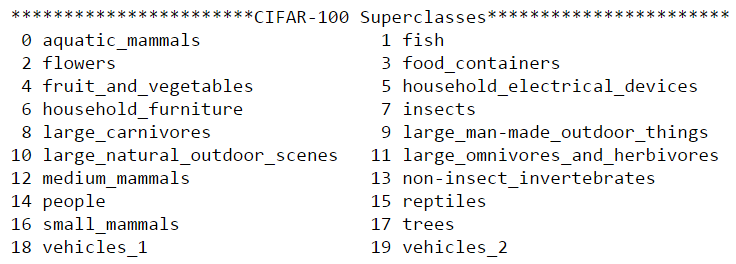
### Data Exploration

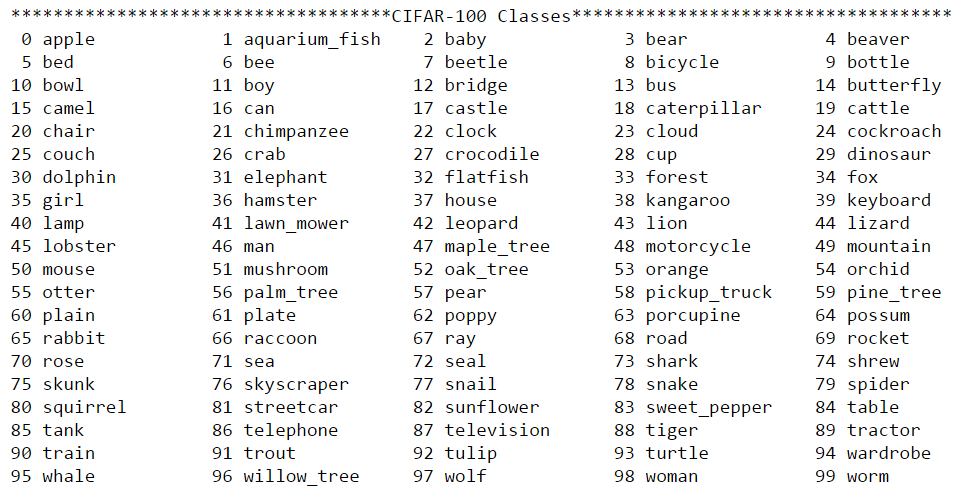
For this project, I used the publicly available dataset from CIFAR-100[[2]](#footnote-2). The CIFAR-100 dataset consists of 60000 32x32 color images in 100 classes with 600 images per class. There are 500 training images and 100 testing images per class. The 100 classes in the CIFAR-100 are grouped into 20 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).

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 |

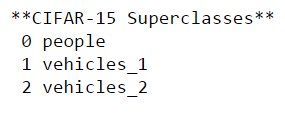
The superclasses and classes are numbered in alphabetical order as shown below:

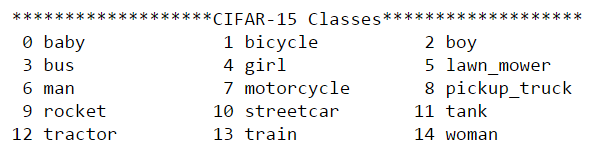




For my project I did not use all the super classes. To keep the computational time less than CIFAR-100 and more challenging than CIFAR-10, I created a sub dataset, namely CIFAR-15 with reduced super classes. I have selected the below super classes by considering an example of user searching for “pedestrians” or “vehicles”.

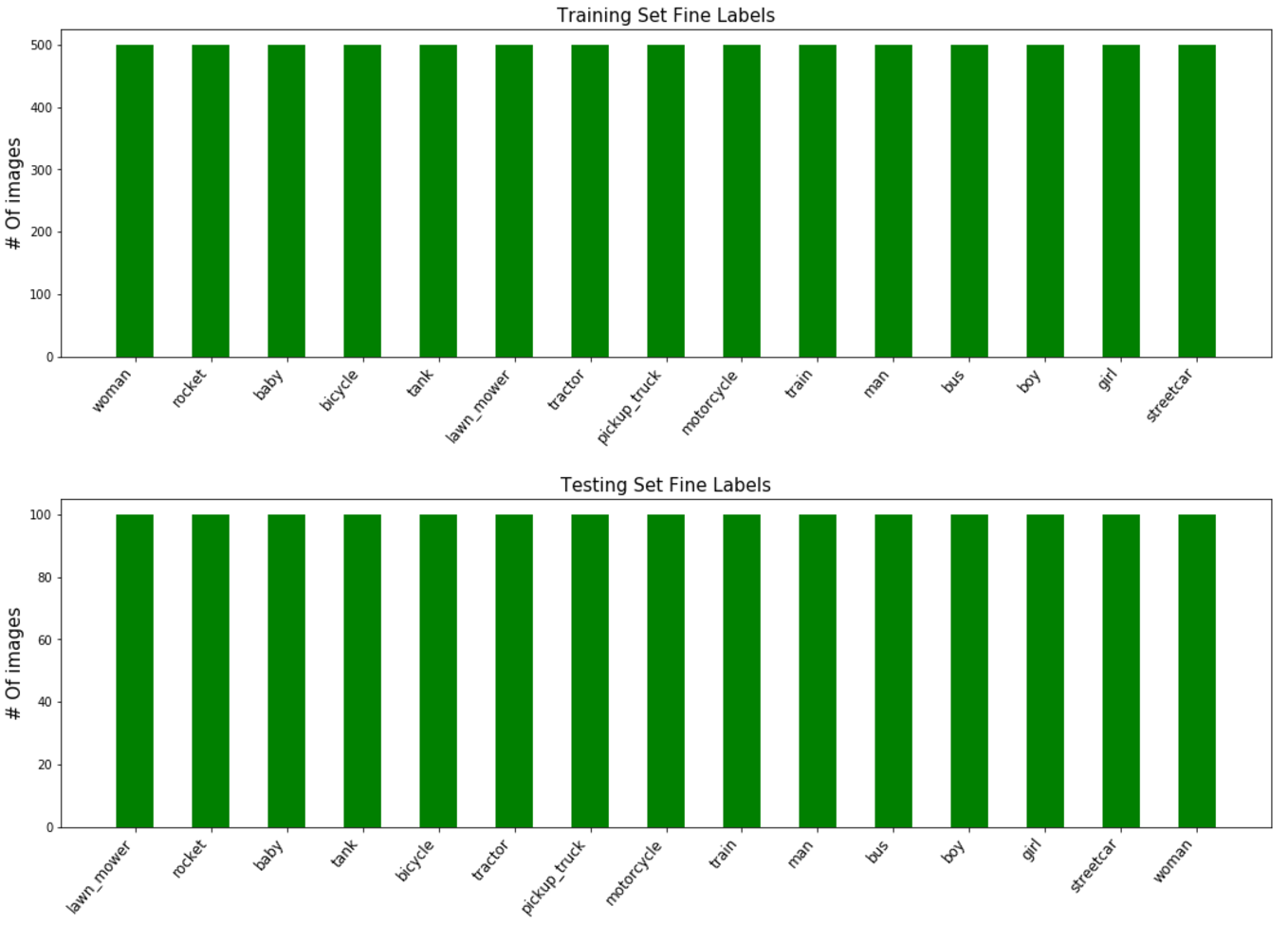
The superclasses and classes in CIFAR-15 are as shown below:





### Exploratory Visualization

Below is the distribution of each fine label in the CIFAR-15 dataset:



It can be seen that only the fine labels of the selected super classes are present in the CIFAR-15 dataset. There are 500 images of each class in the training set and 100 images of each class in testing set.

Below are few images from the CIFAR-15 dataset:



### Metrics

As shown in the Data Exploration section, every image has got a class number, e.g. class “boy” has got class number 2. The class numbers will be one-hot encoded to get a vector of length 15(maximum number of classes in CIFAR-15), namely y\_true. Given an image as input, the model outputs a “prediction” class, namely y\_pred, predicting the class the image belongs to. As the dataset is uniformly distributed across all the 15 classes, I used “accuracy” as my evaluation metric. I also used confusion matrix to get a feel of FP and FN numbers.

### Algorithms and Techniques

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[[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.

CNN learns the basic parts of the objects separately and understands the final object using multiple convolutional layers. The goal of the project would be to use simple model. Simple model implies less memory, less training time and faster execution.

### Benchmark

I used Support Vector Machine(SVM)[[4]](#footnote-4) with Radial Basis Function(RBF)[[5]](#footnote-5) 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.

With the default settings of the sklearn’s[[6]](#footnote-6) Support Vector Classification(SVC)[[7]](#footnote-7), the benchmark model took 91 seconds for training and gave a test accuracy of 25.78%.

## III. Methodology

### Data Preprocessing

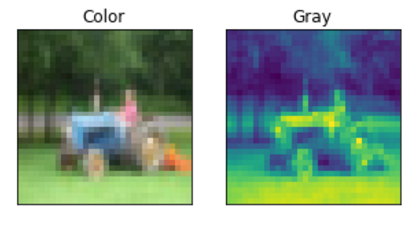
The CIFAR-15 dataset contains training and testing data. The training set is further split into training and validation set. The training set will be used to train the model, i.e. modify its weights, and validation set will be used to check how the model is doing. As the validation set is not used to 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 converted each color image to 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.



Next I normalized the images. Images of same class objects taken in different lighting condition will lead to a huge difference in pixel values but I want 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 normalized the images to be in the range -0.5 to 0.5(why? What about 0 to 1?).

I also augmented the data with random rotation, shifts and flips. This makes the model orientation-invariant and improves its performance.

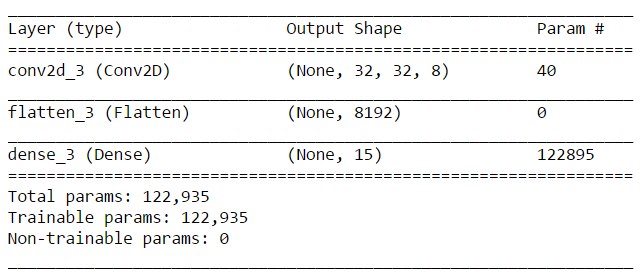
### Implementation

I used Keras[[8]](#footnote-8) for implementing the neural network model.

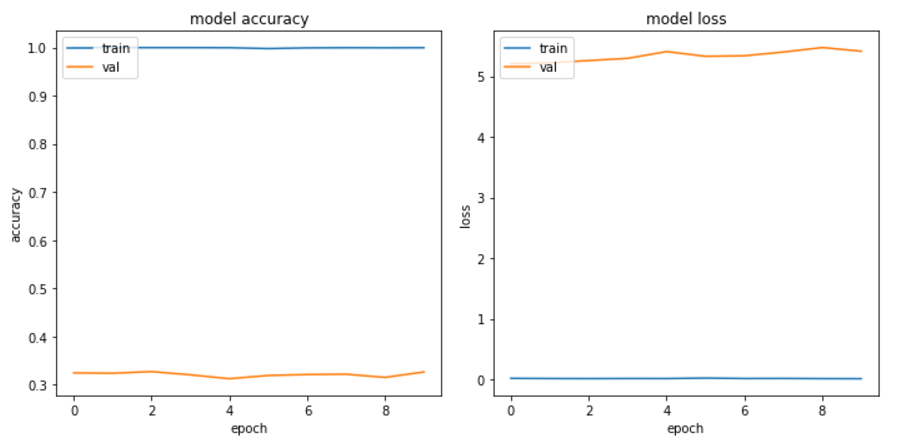
### Refinement

I reached my final model by building upon my initial model and checking the model’s performance with every modification.

1. Model 1: My initial model had one convolution layer and one fully connected layer:

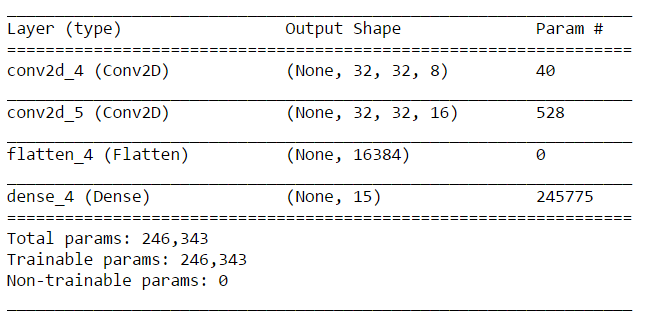


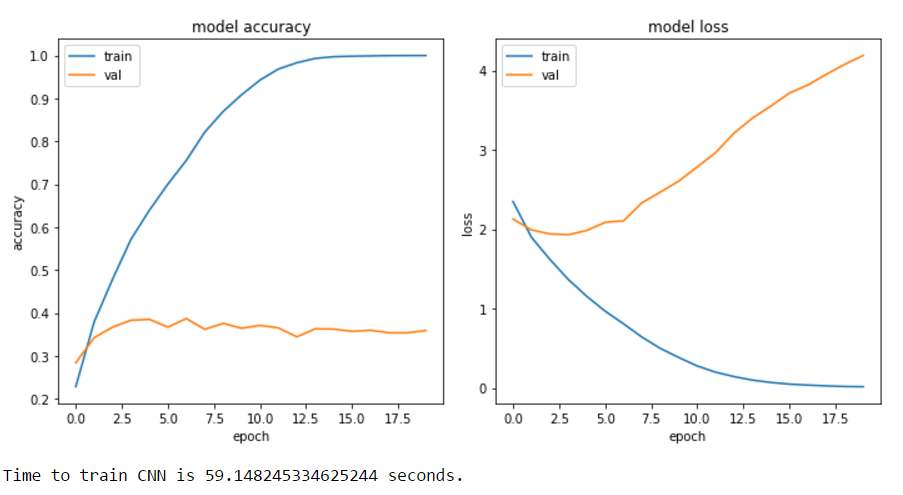
The test accuracy of the model was around 35% which was already better than the test accuracy of the benchmark model. Below is the performance of the model across 20 iterations:



Though the accuracy of the model was better than that of the benchmark model, the graphs above show that model is barely learning. The blue lines shows that the model is doing great on the training set and orange lines show that the model is struggling with validation set. In other words, the model is not able to classify images which it is seeing for the first time.

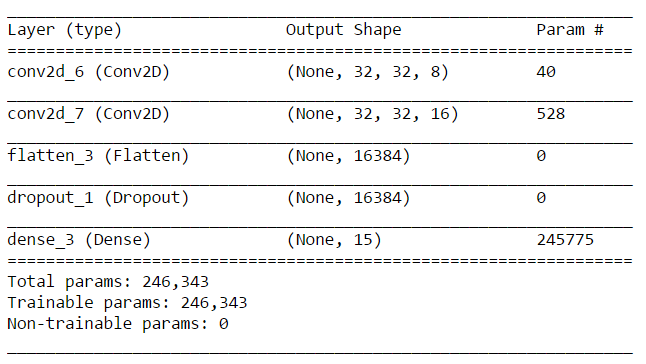
1. Model 2: To help the model learn, I added one more convolution layer through which the model can extract more features from the image:

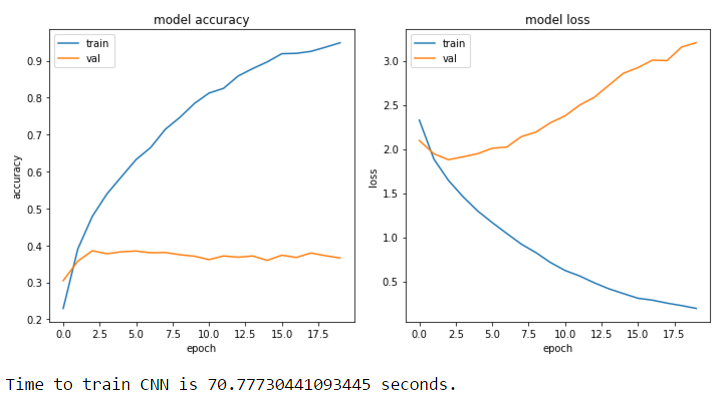




The number of model parameters has increased leading to increase in training time but now the model is learning every iteration as shown by the increase in accuracy and decrease in loss of the training set. Although, the validation set tell a different story. As seen in the loss graph, the loss on validation set seems to decrease for the first few iterations and then the loss increases. This is a typical case of overfitting. What is overfitting? Overfitting can be reduced by introducing non-lineary in the model. Keras provides non-linear layers such as dropout and max-pooling.

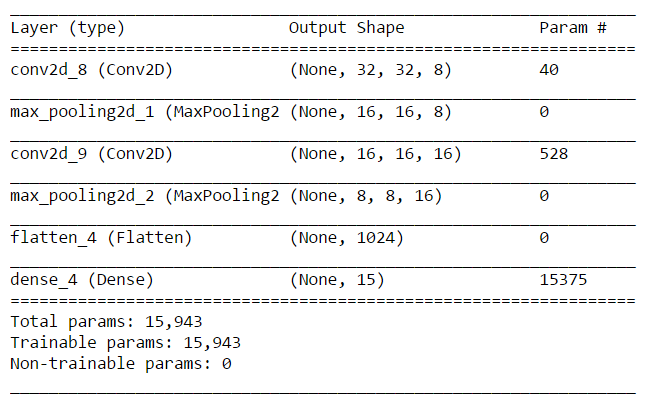
1. Model 3: Added dropout layer:

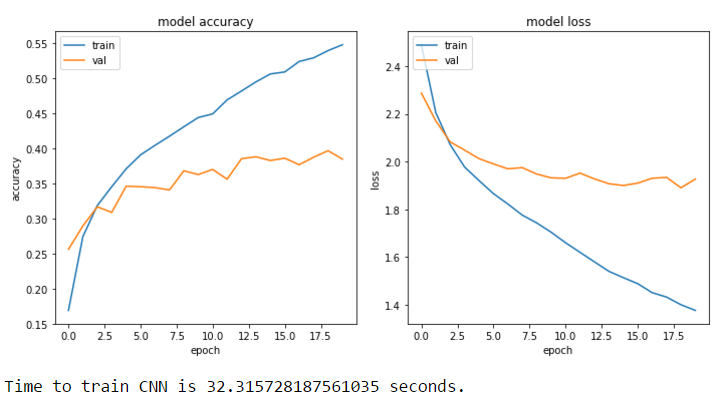




The dropout layer did not reduce overfitting but in turn has negative effect on the model’s performance. So went back to Model 2.

1. Model 4: On Model 2, added max-pooling layer between the convolution layers.





The max-pooling layer was indeed successful in reducing overfitting but it effected the

1. Model 5:
2. Model 6:

## IV. Results

### Model Evaluation and Validation

In this section, the final model and any supporting qualities should be evaluated in detail. It should be clear how the final model was derived and why this model was chosen. In addition, some type of analysis should be used to validate the robustness of this model and its solution, such as manipulating the input data or environment to see how the model’s solution is affected (this is called sensitivity analysis). Questions to ask yourself when writing this section:

* Is the final model reasonable and aligning with solution expectations? Are the final parameters of the model appropriate?
* Has the final model been tested with various inputs to evaluate whether the model generalizes well to unseen data?
* Is the model robust enough for the problem? Do small perturbations (changes) in training data or the input space greatly affect the results?
* Can results found from the model be trusted?

### Justification

In this section, your model’s final solution and its results should be compared to the benchmark you established earlier in the project using some type of statistical analysis. You should also justify whether these results and the solution are significant enough to have solved the problem posed in the project. Questions to ask yourself when writing this section:

* Are the final results found stronger than the benchmark result reported earlier?
* Have you thoroughly analyzed and discussed the final solution?
* Is the final solution significant enough to have solved the problem?

## V. Conclusion

### Free-Form Visualization

In this section, you will need to provide some form of visualization that emphasizes an important quality about the project. It is much more free-form, but should reasonably support a significant result or characteristic about the problem that you want to discuss. Questions to ask yourself when writing this section:

* Have you visualized a relevant or important quality about the problem, dataset, input data, or results?
* Is the visualization thoroughly analyzed and discussed?
* If a plot is provided, are the axes, title, and datum clearly defined?

### Reflection

In this section, you will summarize the entire end-to-end problem solution and discuss one or two particular aspects of the project you found interesting or difficult. You are expected to reflect on the project as a whole to show that you have a firm understanding of the entire process employed in your work. Questions to ask yourself when writing this section:

* Have you thoroughly summarized the entire process you used for this project?
* Were there any interesting aspects of the project?
* Were there any difficult aspects of the project?
* Does the final model and solution fit your expectations for the problem, and should it be used in a general setting to solve these types of problems?

### Improvement

In this section, you will need to provide discussion as to how one aspect of the implementation you designed could be improved. As an example, consider ways your implementation can be made more general, and what would need to be modified. You do not need to make this improvement, but the potential solutions resulting from these changes are considered and compared/contrasted to your current solution. Questions to ask yourself when writing this section:

* Are there further improvements that could be made on the algorithms or techniques you used in this project?
* Were there algorithms or techniques you researched that you did not know how to implement, but would consider using if you knew how?
* If you used your final solution as the new benchmark, do you think an even better solution exists?

1. <https://en.wikipedia.org/wiki/Category:Applications_of_computer_vision> [↑](#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://en.wikipedia.org/wiki/Support_vector_machine> [↑](#footnote-ref-4)
5. <https://en.wikipedia.org/wiki/Radial_basis_function_kernel> [↑](#footnote-ref-5)
6. <http://scikit-learn.org/stable/> [↑](#footnote-ref-6)
7. <http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html> [↑](#footnote-ref-7)
8. <https://keras.io/> [↑](#footnote-ref-8)