# Deep Learning for Image Classification with Caltech 101

Udacity Machine Learning Engineer Nanodegree Program Capstone Project

Renee Reynolds

11 March 2020

---

## Definition

### *Project Overview*

The field of computer vision is becoming increasingly popular as developments in computing power, data availability, and deep learning grow in tandem with the range of computer vision use-cases – from facial recognition to autonomous driving and cashier-less shopping. One key component of successful computer vision applications is object detection. This would be determining the location of an individual's face, detecting objects on the roads, or identifying the items that an Amazon Go shopper chooses during their visit, for example. In this project, we seek to implement an object detection and classification task by categorizing a selection of 101 different object types using a (feed-forward) neural network and a convolutional neural network on the Caltech 101 dataset.

### *Problem Statement*

The goal of this project is to accurately classify the objects that appear in the images in the Caltech 101 dataset. We propose a (feed-forward) neural network and a convolutional neural network (CNN) to solve this object detection problem. We choose neural networks because of their capacity to learn complex features from inputs, and a CNN specifically for their effectiveness in learning patterns in images. Thus, we hypothesize that a CNN will perform better than the standard feed-forward, fully-connected neural network on our image classification task.

### *Metrics*

We evaluate the accuracy of our models using cross-entropy loss, as it is a common choice for multi-class classification models. A lower cross-entropy loss indicates higher model accuracy.

# Analysis

## *Data Exploration*

The data that we use for this project is the Caltech 101 dataset. This dataset comprises images collected in September 2003 by Fei-Fei Li, Marco Andreetto, and Marc 'Aurelio Ranzato at Caltech. Included with each image is a bounding box of the object in it as well as a carefully human-labeled outline of the object. The details of this dataset are as follows:

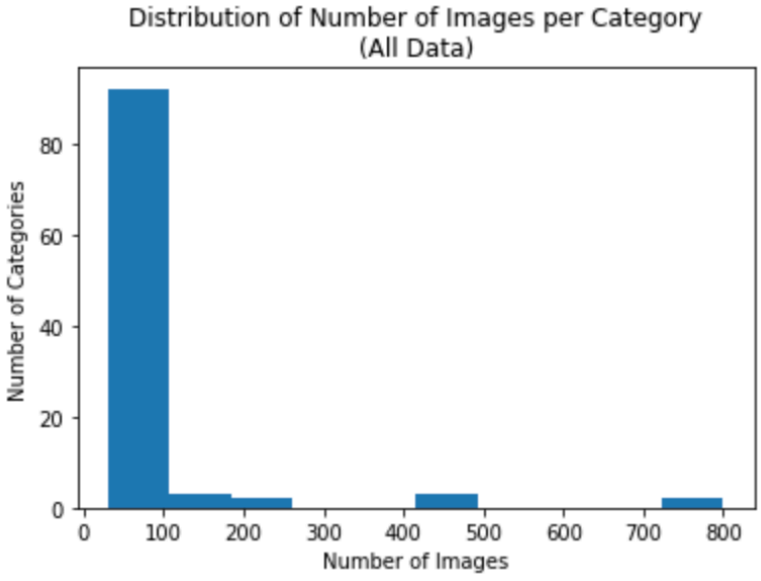
- 9,146 observations (i.e. images saved in .jpg format)
- 101 object categories, plus one additional background/noise category
- 40-800 images per category (on average ~50 images per category)
- 300 x 200 pixels per image

Some of the object categories include: "airplanes", "beaver", "crocodile", "dollar_bill", and "stop_sign".

**Data Source:** http://www.vision.caltech.edu/Image_Datasets/Caltech101/

## *Exploratory Visualization*

We visualize the distribution of images per category in the dataset:



The distributions of the number of images per category in the Caltech 101 dataset is skewed right, with a peak between 0 and 100 images. The average number of images per categories is 89.657 and the median number of images per category is 59.0.

### *Algorithms and Techniques*

We choose to implement a feed-forward, fully-connected neural network and a convolutional neural network for our image classification task, as neural networks are capable of learning complex features from inputs through their layered design.

### *Benchmark*

The benchmark model for our analysis is the logistic regression model, as it is a simple and historical model for performing classification tasks. The logistic regression model uses the logistic function to convert log-odds to a probability (i.e. a value between 0 and 1). Thus, for our classification problem, the logistic regression model will output a 102 x 1 probability vector for each image, where entry *i* is the probability that object *i* appears in our image, for *i* = 1, 2, … , 102 (101 object categories + 1 background/noise category = 102). We expect objects that appear in an image to have a high probability, while objects that do not appear in an image to have a low probability.

## Methodology

### *Data Preprocessing*

First, we preprocess our data by scaling each image down from ~300 x 200 pixels to a 64 x 64 x 3 RGB image for efficiency.

Next, we split our data into training and test sets. We assume our image classification task requires a significant amount of data to train, so we reserve 90% of our data (~8,200 observations) for our training set and the remaining 10% (~910 observations) as our test set. To ensure that our training and test sets have the same distribution of objects, we first stratify the data into its 102 categories; then, we sample 90% of each stratum for the training set and add the remaining 10% to the test set.

### *Implementation*

Using the Keras python library, we implement our neural network with only two hidden layers (as computing power is limited) with ReLU activation functions and 1,024 nodes per hidden layer. The output layer has 102 nodes with a softmax activation function to estimate the probability of the input image having each of the 102 categories of objects. Like the

benchmark logistic regression model, we train this simple neural network for 100 epochs and with a batch size of 64 images.
For our convolutional neural network, we use a ResNet50 model with weights pre-trained on the multi-million-image corpus, ImageNet. We remove the last layer and instead train our own custom fully-connected hidden layer of 1,024 nodes with a ReLU activation function, followed by an output layer of 102 nodes with a softmax activation function. Again, we train this CNN for 100 epochs with a batch size of 64 images for consistency across all models.

### *Refinement*
Given more time, we could improve our neural network by tuning our hyperparameters. This includes changing the number of hidden layers, the number of nodes in each hidden layer, the activation functions of the hidden layers, and adding dropout layers.

## Results

### *Model Evaluation & Validation*
The benchmark logistic regression model (i.e. a "neural network" with no hidden layers) achieved test loss of 2135.318 with test accuracy of 0.380. The simple feed-forward, fully-connected neural network achieved test loss of 4.219 and test accuracy of only 0.079. Finally, the ResNet50 CNN achieved test loss of 32.931 and test accuracy of 0.416. All three models were trained for 100 epochs with a batch size of 64 images. These hyperparameters were chosen using "rule of thumb" as there are not definitive strategies for choosing the number of epochs or batch size, and similarly for the hyperparameters of our neural network (i.e. number of hidden layers and number of nodes per hidden layer).

### *Justification*
Compared to our benchmark logistic regression model, our feed-forward, fully-connected neural network performed significantly worse in terms of test accuracy, while our CNN performed slightly better. Although the CNN did produce a higher test accuracy, this improvement was only very slight, and could possibly be improved by reducing potential overfitting to our training set. To improve our feed-forward, fully-connected neural network, we could experiment with different combinations of hyperparameters.

## Conclusions
In conclusion, our models did not quite follow our hypotheses. Our feed-forward, fully-connected neural network underperformed compared to our benchmark logistic regression model, while our CNN provided only a slight improvement over our benchmark.

Given more time and resources, further improvements in hyperparameter tuning could provide increases in accuracy towards solving our image classification problem.