# Udacity Machine Learning Capstone Project

## Image Classification using CNN and Transfer Learning on the "Yelp!" dataset

Project by,
Gautham Venkatesha Reddy
March 2nd, 2018

UDACITY

# 1. Definition

## Project Overview

The project explores the possibility of building a Neural network architecture to classify images on the "Yelp" dataset. The dataset set consists of nearly 200,000 images with labels such as 'Inside', 'Outside', 'Food', 'Drink' and 'Menu'. Yelp has gathered nearly 146 million data points since it's inception **[1]** and it is humanly impossible to classify and sift each data point manually. Also, due to the variability in data, traditional style of programming can render the application inept it it's functionality. Thus, Machine learning and deep learning models can be trained to achieve superior performance on the given dataset.

Correctly classifying images of food will not only help 'Yelp' become a dominant player it's field, it will also indirectly help small businesses improve their quality of service and help prioritize and display relevant images of their signature dishes on their Yelp business page. This will attract more customers and help generate monetary value for these businesses.

## Problem Statement

As mentioned in the Project Overview, the aim is of this project is to correctly classify if the images provided by Yelp are either 'Inside', 'Outside' (Of the restaurant), 'Food', 'Drink' or 'Menu'. If the image is that of a food or a drink, then we further classify the type of food or drink that is present in the image. For example, if the image contains a burger, fries, noodles, milkshake etc.

Since this project seeks to solve Image classification, we can address this problem by building a Convolutional Neural Network (CNN) **[2]** architecture to classify our images. This type of machine learning model provides highly accurate results for a complex dataset. Although, I have large troves of data available at my disposal, I have chosen to work with a relatively small subset of data (Around 3000-5000 images) and employing transfer learning architectures such as VGG19**[3]**, ResNet**[4]**, Inception**[5]** and Xception**[6]** architectures. This is certainly advantageous since our training time will be low without compromise in the performance.

The weights for the above transfer learning architectures were trained on the 'ImageNet' data and since our labels are quiet different from the labels in the ImageNet dataset, I have decided to retrain the whole transfer learning architectures.

### Metrics

To measure the performance of our model, I have chosen 'Accuracy' as a metric. Since the frequency of occurrence of labels are well balanced, accuracy provides the most intuitive and best measure of performance.

Accuracy if defined as,

$$Accuracy = \frac{tp + tn}{tp + fp + tn + fn},$$

where,

$$tp = True\ positive$$
$$fp = False\ positive$$
$$tn = True\ negatives$$
$$fn = False\ negatives$$

or put simply,

$$Accuracy = \frac{Total\ number\ of\ correct\ classifications}{Total\ number\ of\ data}$$

We can also further explore the quality of our classification by creating and analyzing a confusion matrix.

## 2. Analysis

### Data Exploration and Analysis [7]

The dataset downloaded from the 'Yelp' dataset challenge site provides a folder for images (200,000 images) and a 'photos.json' file which contains the information of the image in the images folder. This information is stored in a JSON format as shown below,

{"photo_id":"QYQHXQS5u7EWqRHljS3AOQ","business_id":"6nMYROXu0VX4Ytpdsfi3XA","caption":"Night time","label":"outside"}
{"photo_id":"vfIRRaVHyxHPGLd3f5g2qQ","business_id":"jaoIDcqPZVXwwSPXFRSC-Q","caption":"","label":"inside"}
{"photo_id":"q81QJEGXWNV5ZEXFWCl8eg","business_id":"LR_99E7tVUfxqemvcGkDzw","caption":"Homemade Hummus","label":"food"}
{"photo_id":"NU8IU0qHsM6d6aXbPhr44w","business_id":"ey9cTjWXgTJ5OlD0HPd_A","caption":"Latte de Julia","label":"drink"
{"photo_id":"sfUbV8xo528CrtzDkIITjQ","business_id":"L1XHTn7S-6har9UGAPjcWQ","caption":"","label":"menu"}

Some of the images for each labels are shown below,

**Fig 1, Outside**

**Fig 2, Food**


**Fig 3, Inside**


**Fig 4, Menu**

The above images are from the Yelp dataset. As we can see, these images have varying width and height and thus we must first resize them into a constant dimension before we can feed these images into a CNN for training. For the first part, I have chosen 600 X 400 dimension tensor. (Since this is approximately the average size of images)

**Fig 4, Drink**

As for the second part of the project, where we classify the type of food or drink present in the image, I found it extremely unreliable to filter the images using 'photos.json' file to filter the type of food/drink present in the image. Since, the captions can have many different foods present in the image. For example, the caption may contain burgers and fries and it is difficult to label the image as either a burger or fries or there may be many other foods present in the images. The network may give bad predictions and may not train well.

For example, picture in the left (Figure 5) shows an image which was searched with "fries" as a keyword in the caption column. Although, the caption and the image does contain the word "fries", there are many different foods present in the image.

**Fig 5**

Another example on the right (Figure 6) contains an image of burger and fries, although the keyword used to filter this image was "burger" and the burger looks like it is covered in a decorative manner.

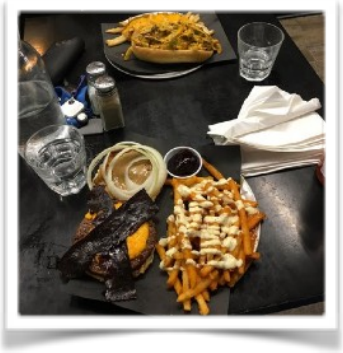


**Fig 6**

Since such images will not properly aide the network to train and generalize well on the dataset, I have decided to take another approach to solve this problem. "Fatkun" **[8]** is a Google Chrome plug-in which allows the user to download google images in a batch. Thus, with a search keyword such as "French fries" or "burgers" I was able to quickly download 700-800 relevant images per class.

The images for the second part of the project will be reshaped to 224X224 size since most of the images downloaded from Google Images are approximately within this dimension. These dimensions will be passed as input shapes to the neural network.

### Benchmark

In this project, a simple Convolutional Neural Network with 5 hidden layers is chosen as a benchmark for all neural networks developed in this project. This small neural network certainly provides reasonably good accuracy as opposed to random guessing. This benchmark accuracy metric could be surpassed by a more complex, large neural network such as VGG19, ResNet, Inception and Xception architectures. Other similar attempts have been made on food classification using a ResNet architecture **[9]**. The programmer achieved around 85-90% accuracy with these large architectures. This project aims to achieve similar magnitude of accuracy on our dataset.

## 3. Methodology

### Data Preprocessing

The data obtained are that of images and thus the pixel values which range between 0-255 must be normalized to lie between 0 and 1 to help convergence. Further more, the dimensions of the images must be expanded when passing through the neural network architecture when using Keras library.

All neural networks need to take a number of fixed size features as inputs and because the images have varying dimensions, we must first need to reshape the images. For the first part of the project (Predicting if the image is Inside, Outside, Food, Drink or Menu), the images are resized to 600X400 (explained in the Data analysis and exploration step) and in the second part (predicting the type of food and drink), the images are resized to 224X224.

Since, the Yelp dataset images contain nearly 200,000 images, I have chosen to work with only a small subset of images of around only 3,000-4,000 images. Even though with such small samples I was able to achieve high accuracy and reduced the training time. The images are chosen such that the frequency of classes are well balanced, for which accuracy metric is chosen.

### Implementation

For simple neural networks with 4-5 hidden layers, it is very easy to train and thus a simple optimizer such as ‘RMSPROP’ works well and we achieve convergence very quickly. But for large neural networks such as VGG19, ResNet etc, RMSPROP does not converge well and weights do not update quickly. Since there are many parameters to train on, ‘Stochastic Gradient Descent’ achieves convergence extremely well. The learning rate chosen on large neural network is ‘0.0001’ with a momentum of 0.9.

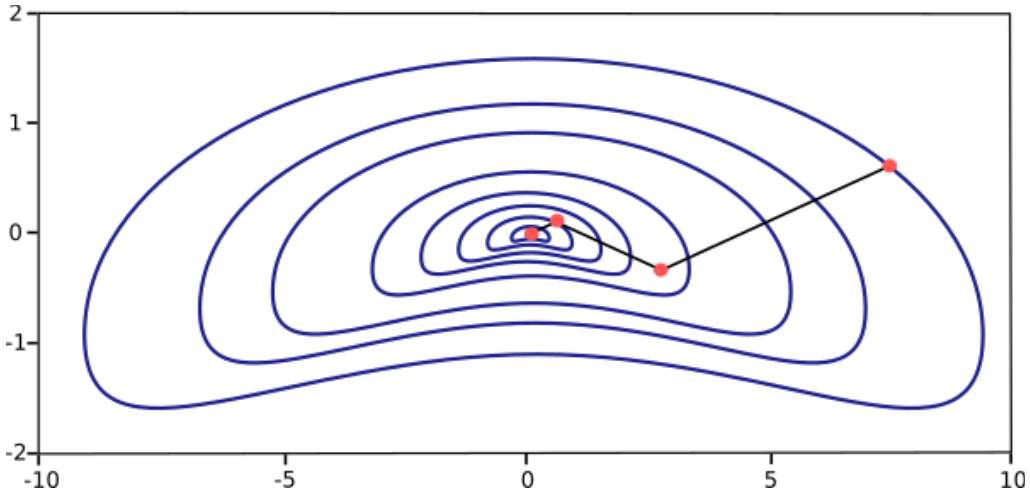



**Fig 7, Stochastic Gradient descent**

**Pseudo code for SGD: [10]**

**Require:** Learning rate $\epsilon_k$ (Learning rate at iteration 'k')

**Require:** Initial parameter $\Theta$

**while** stopping criteria not met **do**

Sample mini-batch of '*m*' examples from the training set $\{\mathbf{x_1}, \ldots \mathbf{x_2}\}$

Corresponding targets $\mathbf{y^{(i)}}$

Computer gradient estimate: $\hat{\mathbf{g}} \leftarrow + \frac{\mathbf{1}}{\mathbf{m}} \bigtriangledown_{\Theta} \Sigma_i L(f(x^i; \Theta), y^i)$

Apply update: $\Theta \leftarrow \Theta - \epsilon \hat{g}$

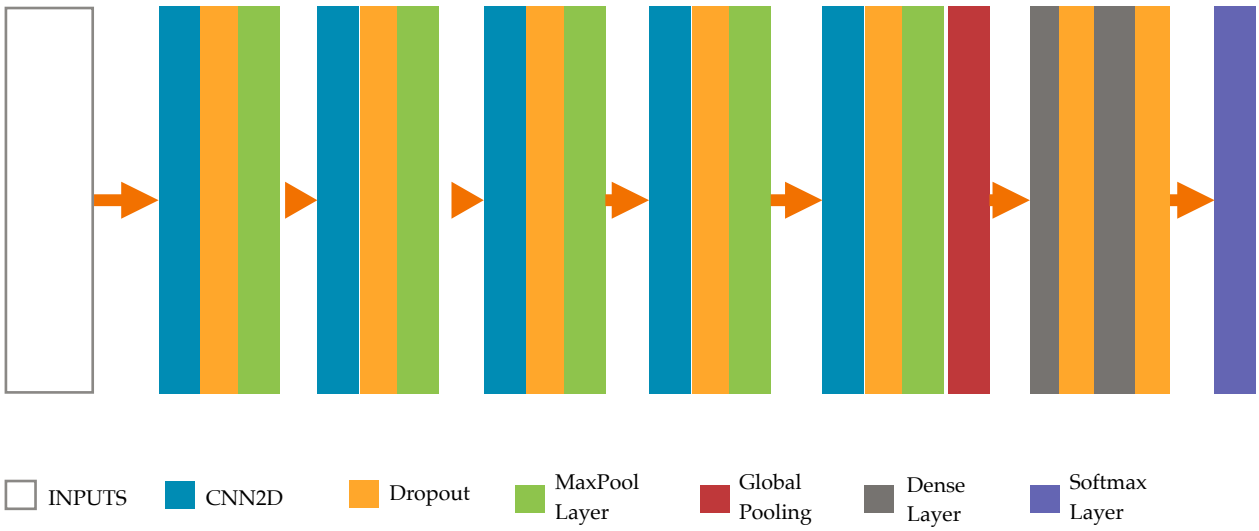
**end while**

## Transfer Learning

Since the dataset collected and labels of predictions are significantly different from those trained on 'Imagenet' datasets, I have decided to retrain the whole architectures without freezing any layers. Thus, the model will then learn weights and filters pertaining to the dataset provided. Also, since the number of datapoint are around 4000-5000 images, this should be sufficient for the neural network to learn weights correctly and generalize well on the test data.

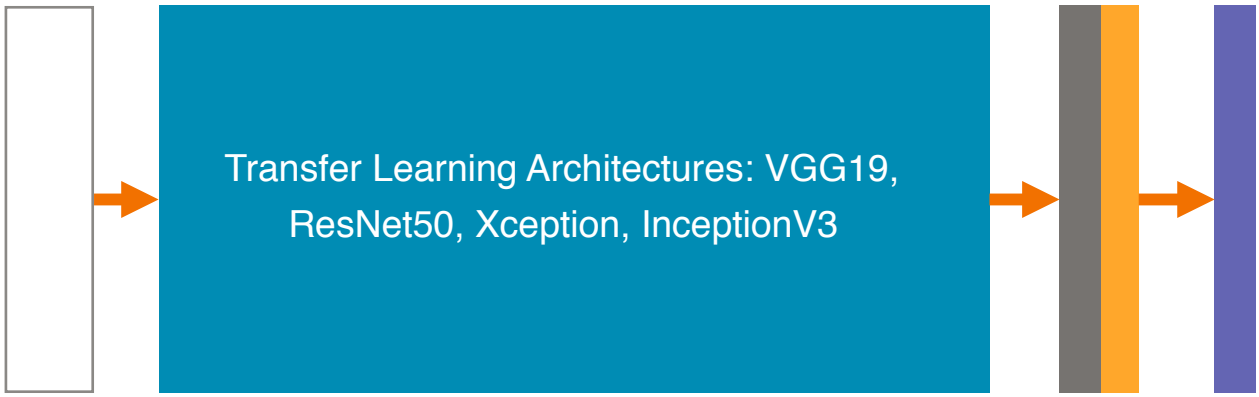
## CNN architectures:

*Self built CNN architecture:*




The above architecture shows a graphical representation of a self built simple CNN architecture with mostly 5 (4 for one of the architecture) consecutive CNN layers (with filter sizes 16, 32, 64, 128 and 256 respectively, with 'relu' activation) followed by a Dropout layer (30%) and Max Pooling layer with pool size of 2. In the last layer of CNN, I have added Global Average pooling layer. This is followed by two Dense layers and dropout layers. Next is the softmax layer for prediction. The input layer for the first part of the project (Identifying the environment) has dimensions 600X400X3 and for the second part (Identifying the type of food and drink) the input layer has dimensions 224X224X3.

*Transfer Learning architectures:*



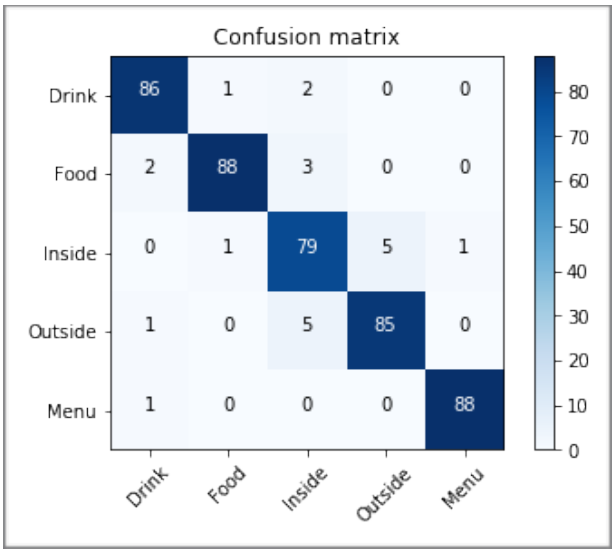

The above architecture is that of Transfer learning, followed by a fully connected dense layer and dropout of 50% and a softmax layer for prediction.

# 4. Results

## Accuracy on Part 1 (Environment) training

| CNN Architectures | Accuracy |
|---|---|
| Simple CNN with 4 hidden layers | 76.11% |
| VGG19 | 94.64% |
| ResNet50 | 95.08% |

The above table shows the performance accuracy of various CNN architectures when recognizing it's environment. We can observe that 'ResNet50' gives slight better results than VGG19 and vastly superior accuracy compared to a simple CNN architecture with 4 hidden layers. Thus, I have chosen ResNet50 as my architecture to identify it's environment.

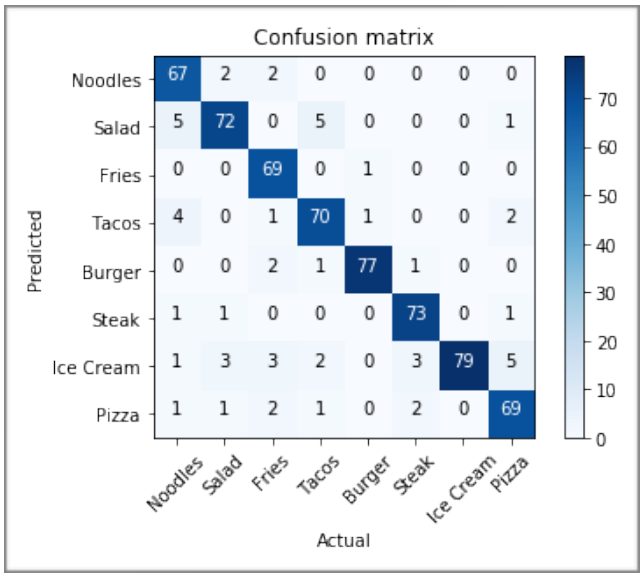



The adjacent figure shows a confusion matrix on the test dataset using the ResNet50 model. We can see that there are many false positives for labels 'Inside' and 'Outside'. This is because, sometimes the neural network may mistaken a photo for being 'Outside' rather than 'Inside' if the image is dimly lit and also if it is too bright. These are mistakes that a human might potentially make. It also sometimes mistaken a 'food' for a 'drink' and vice versa. And if the image of the food is shot at a distant then it might classify the image as being 'Inside' or 'Outside'

## Accuracy on Part 2 (Type of food) training

| CNN Architecture | Accuracy |
|---|---|
| Simple CNN with 5 hidden layers | 66.56% |
| VGG19 | 89.38% |
| ResNet50 | 91.28% |
| Xception | 88.11% |
| InceptionV3 | 88.27% |

The above table shows the accuracy performance on predicting the type of food present in the image. We see that once again, ResNet50 provides very good results. Thus, I have chosen to use ResNet50 architecture to predict the type of food.



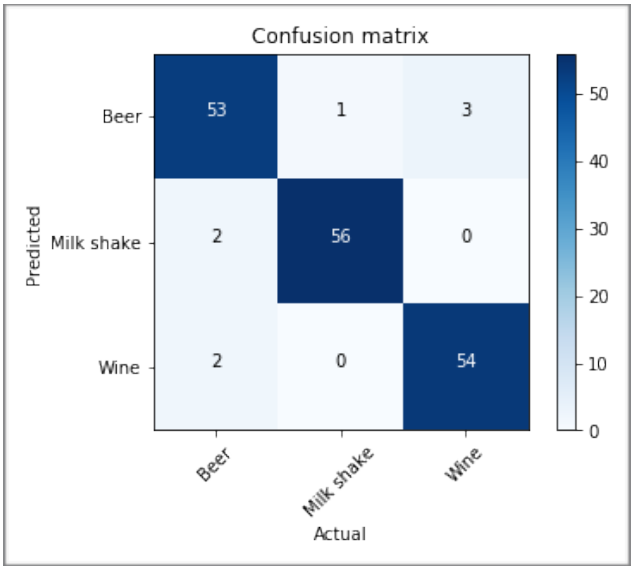

The adjacent figure shows a confusion matrix of the ResNet50 model predictions. We find many false positives for the 'Salad' label. It predicts few images as being either 'Noodles' or 'Tacos'. This may because there are many green vegetables such as broccoli and lettuce on these food and thus the model may end up predicting salad as being either a 'Taco' or 'Noodles'. Sometimes the model may predict 'Tacos' as 'Noodles' because of vegetables and color. 'Ice Cream' has many false positive labels because of large varieties in color and shape.

## Accuracy on Part 3 (Type of drink) training

| CNN Architecture | Accuracy |
|---|---|
| Simple CNN with 5 hidden layers | 75.43% |
| VGG19 | 91.22% |
| ResNet50 | 95.32% |

ResNet50 once again gave best results. Thus, I have chosen ResNet50 as the model.

'Beer' and 'Wine' has made many false positives by predicting the other. Referring to 'capstone_drinks.ipynb' notebook, we can see that image of grapes were predicted as being 'Beer' and certain alcohol bottles were misclassified. This is a fault when collecting the datasets. Thus carefully selecting the dataset can provide even better results.




**Video Demonstration:** https://www.youtube.com/watch?v=19gDX1P7N5k&feature=youtu.be

## 5. Conclusions

In conclusion, this project has touched and exceeded the said benchmark. The accuracy of each of the network is above 90% and I believe the models provide a reliable and functional application to classify images for 'Yelp!'.

To reduce incorrect predictions, 'Yelp!' can take another approach by choosing images if and only if the neural network predicts a high probability for a given image by setting a certain threshold. For example, if an image of a burger has a probability of say 57% and the threshold is 75%, the said image could then be manually classified by a human to ensure correct labelling rather than a CNN classifying the image.

### Reflection

This application could be used by 'Yelp!' to classify images with minimal or no human intervention. It would certainly help boost their productivity and reduce their overheads giving them a competitive edge in the marketplace and provide good services to their users.

This project aimed to solve image classification using learning algorithms such as Convolutional neural networks and Transfer learning. We first collect and preprocess data into acceptable tensors and train the model using various CNN architectures. We also analyzed the quality of our performance using Confusion matrix and reasoned as to why the model made certain false predictions. Furthermore, we can improve the performance using various other metrics and building even deeper architectures to give more exceptional performance and 'Yelp!' can take better approaches towards their data collection techniques and also gather more data to further their performance.

I faced many challenges while trying to gather correct datasets (especially for food and drink types) and I had to explore different ways to gather the data with appropriate labels. It is always a challenge to collect and work with large amounts of data and nothing is more crucial than patience.

### Improvements and further features

By collecting more data with correct and appropriate labels, we can further improve the performance on our architecture. Data augmentation techniques may also help increase the performance slightly. Due to the lack of time and computational resources, I was unable to achieve these tasks. But 'Yelp!' can certainly undertake such work.

TensorFlow is a one of the best libraries for an industrial scale deployment of deep learning algorithms. Now that prototyping of the application is complete using Keras, we can now develop this application in TensorFlow.

"YOLO" **[11]** is an object recognition problem and since images have multiple foods/drinks present, this algorithm can detect such objects in images. 'YOLO' algorithm can be incorporated for video classification as shown in this link and this would certainly make a fun application for 'Yelp!' users. This, when combined with other computer vision features may ensure that a good shot of your food is always taken.

# 6. Bibliography


1. About Yelp: https://www.yelp.com/about
2. Convolutional networks for images, speech, and time series, Y LeCun, Y Bengio - The handbook of brain theory and neural networks, 1995
3. Very Deep Convolutional Networks for Large-Scale Image Recognition, Karen Simonyan, Andrew Zisserman
4. Deep Residual Learning for Image Recognition, Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun
5. Rethinking the Inception Architecture for Computer Vision, Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna
6. Xception: Deep Learning with Depthwise Separable Convolutions, François Chollet
7. Yelp Dataset: https://www.yelp.com/dataset/challenge
8. FatKun: https://chrome.google.com/webstore/detail/fatkun-batch-download-ima/nnjjahlikiabnchcpehcpkdeckfgnohf
9. Stratospark's GitHub page: https://github.com/stratospark/food-101-keras
10. Deep Learning, MIT Press, Ian Goodfellow and Yoshua Bengio and Aaron Courville
11. You Only Look Once: Unified, Real-Time Object Detection, Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, University of Washington