A Survey of Caltech-101 Object Classification Dataset

Hanming Wang
Boston University
hita@bu.edu

Leo Gan
Boston University
leogan@bu.edu

Tilak Agarwal
Boston University
tilak02@bu.edu

Hunter Chun
Boston University
hunterch@bu.edu

Abstract

In this survey, we took research on a small images dataset called "Caltech-101". We also studied a CNN(Convolutional Neural Network) method to solve object classification based on this dataset. We then evaluate the performance of different CNNs, 5 different model structures are constructed with the modification of layers. Finally, we designed a nearest neighbor algorithm to solve this problem based on Locality Sensitive Hashing. With the models constructed, we evaluate their performance by their efficiency and accuracy.

1 Code Deliverable

https://github.com/labmem008/CS542-Team-13

2 Introduction

Image classification and recognition has numerous of practical usages in real life applications. It is inevitably important to select a accurate and efficient model for the application. CNN is a model that has advantages particularly in image classification. With the construction of models with different sets of layer, the performance are evaluated to investigate the possible impact on the accuracy by the addition or deletion of different layers. Nearest neighbor algorithm is the more general machine learning algorithm which finds the closest category that the input image belongs to. In our nearest neighbor algorithm, locality-sensitive hashing is used to improve the accuracy. By testing and comparing both approaches to object classification, both advantages and disadvantages of both models can be analysed.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

2.1 Dataset

Description For the model training and validation, the dataset is called "Caltech-101" which is a collection of images which has a total of 101 object categories. In total, there are 8677 different images in the dataset. Each object category contains different number of images, the least has 31 images, and the most has 800 images. All images in the dataset is colored with three color channels(Red, Green, Blue).

2.2 Objective

The objective of this project is to implement a simple object classification model in two general approaches. We studied and implemented two methods, the neural network method, and locality-sensitive hashing. In the following section, we will introduce how we implement these two methods.

3 Convolutional Neural Network Approach

Data Processing After the dataset is imported from the web source, they are stored as image files in the extension of jpg(s). When storing all images into an array, they are resized into the same dimension 300x200. This resolution is used because it is the average out of all images. By using the average resolution, it requires the least number of operations to resize all images to one resolution. The array has the dimensions of (8677, 200, 300, 3), 8677 is the total number of images, 200,300 is the resolution, and 3 is the number of color channels. With all images stored in one array, this is the independent variable of the model X. Since a single color channel would have the value between 0-255, all values of channels are divided by 255 to normalize the data to the range between 0 and 1. Then all values of category data are stored in another array of size (8677,1), which is the dependent variable of the model. The dataset is not yet seperated by training and validation, so for the seperation of the dataset, our project takes random partition of the set to seperate the training and validation sets using the library sklearn. Now the models are ready to be constructed and trained.

Methods Convolutional Neural Network(CNN) is considered as the "state of the art" of image classification. For a CNN model, multiple convolutions layers are used, each convolutional layer is similar to the idea of human brain, it filters out the feature of the model using dot product calculation by taking number of partitions of the image to construct a feature map. In a typical model, there are two sets of convolutions layers with max-pooling to learn the features. The convolutional layers are used to construct the feature maps from the input, then max-pooling takes the maximum within the pool size, to exaggerate each possible feature within the image. After two sets of convolutional layers and max-pooling. The pooled feature maps are now flattened into 1d array by a flatten layer, then it fully connects to a output layer, usually dense layer to dense all the neurons back to the number of categories, in this model is 101.

Activation function For the convolutional and dense layer, activation function is needed for non-linear transformation of the data. For our models, we choose our activation function as the rectified linear unit, which abbreviates to Relu. Relu is as simple function that takes x as the input and output x if it is positive and 0 if it is negative. Therefore, with the active function, all the negative values cluster to 0, and all positive values stay the same. Using Relu as the activation function provides several benefits in the model training. It is efficient by the simplicity of the function, so it requires less computational power than other activation functions such as Sigmoid, which does log operation. Sigmoid is also known for the problem of vanishing gradients because all output value is between 0 and 1, the derivative of the function get very close to 0 when the input is very large or small. When dense layer is the output, it uses softmax function to find the max value in the result array. The category with the max value indicates that the model classifies the input as that category.

Models Five models with different structures are constructed for the comparison of different layers.

Model 1: Convolution -> max-pooling -> flatten - > dense -> output

The first model is a simplistic CNN approach where there is one layer for each type of layer. This model trains relative fast because it has the least number of neurons out of the five. When training finished, the accuracy for training is almost 1, and loss is close to 0. However, the validation loss

remains very large, about 10 times of training loss. The validation accuracy is about 0.52, which has much space for improving. The difference between the validation and training loss indicates that there is over-fitting of the training data.

Model 2: Convolution -> Convolution -> max-pooling -> flatten - > dense -> output

To improve from the first model, one more convolutional layer is added after the first one. With the additional convolutional layer, it provides more depth to the feature learning process. The neurons in the network doubled, as well as the training time for each epoch. The validation loss and accuracy improved from the first one. Loss improved from 3.45 to 3.15, accuracy from 0.52 to 0.55. Therefore, by adding convolutional layer, the model is more accurate and less over-fitting.

Model 3: Convolution -> Convolution -> max-pooling -> flatten - > dense ->dropout -> dense -> output

Over-fitting is reduced but still persist in the last model. To further improve the over-fitting, a dense and a dropout layer is added before the output layer. Dropout layer is where it randomly deactivate neurons in the previous dense layer in each epoch. By deactivating random neurons in the network, the model is trained less specific for the training data. That is, no certain neurons would have dominating impact on the prediction of the data input. With the additional dropout process, the difference between the training and validation loss has lowered, meaning reduced over-fitting. However, the validation accuracy is not improved from the last model.

Model 4: Convolution -> Convolution -> max-pooling -> Convolution -> Convolution -> max-pooling -> flatten - > dense -> output

To further improve the accuracy of the model, one more set of convolutions max-pooling is added to the first set of layers. For this model, dropout is not used. With two cycles of convolutional layers and max-pooling, the model can recognize more complex and subtle feature where one cycle of layers would not able to classify. The validation accuracy of this model is better than model 3: one set of convolutions without dropout and similar to model 4: one set with dropout. The validation loss of this model has over 4, which is the highest among all models. The reason is that with multiple layers of convolutional layers without dropout, some neurons in the network would have more weight than all others after some training, which causes the over-fitting of the model.

Model 5: Convolution -> Convolution -> max-pooling -> Convolution -> Convolution -> max-pooling -> flatten - > dense ->dropout -> dense -> output

Bringing back the dropout process to two sets of convolutional layers. A dense layer and a dropout layer is added before the output. This model is the most accurate out of the five model, it has the highest validation accuracy about 0.56, and validation loss around 2.8. Dropout layer works well in pair with multiple convolutional layers.

Performance To evaluate the performance of different CNN models, all 5 models are trained until their training accuracy is close to 1 or their validation accuracy stops improving. For each model, experimental data recorded are training epochs, epoch time, training accuracy, training loss, validation accuracy and validation loss. After constructing the models, the average category accurate can be calculated by taking the dataset as the input to find all accurate of each category. Then sum up all categories and find the average of the accuracy.

Result [1] [2]

model	epchos	epcho time	loss(train)	acc.(train)	loss(val.[1])	acc.(val.)	avg category acc.
1	5	173	0.0318	0.9973	3.4671	0.5235	0.8472
2	5	602	0.0135	0.9987	3.1584	0.5484	0.8362
3	16	945	0.2923	0.9176	3.1858	0.4986	0.8164
4	5	1147	0.0166	0.9972	4.2347	0.545	0.8347
5	17	1784	0.0333	0.9919	2.8196	0.5576	0.8472

The result of all models shows that the most complex model 5 has the highest validation accuracy and average category accuracy while validation loss being the lowest. This indicates that two or more

[1] val. -> validation

[2] acc. -> accuracy

cycles of convolutional layers are superior than using only one cycle or one layer of convolution. The depth of layers of the model 5 enables it to classify images that have complex features, and provide a better overall performance. This experiment evaluated CNN models with one or two cycles of convolutional layers. For the future testing and improvement, more complex layers of convolutions as well as the combination between CNN and other neural networks can be considered and explored.

4 Nearest Neighbor Approach

4.1 Abstract of Nearest Neighbor Approach

In this section, we are going to solve the original object classification problem using approximate nearest neighbor search technique. Given a query image of unknown category, we would search the entire dataset and try to find the image that has the shortest distance between them. We then predict the category of the query image by its nearest image's category. This process is called nearest neighbor search and it would normally take a long time due to the so called curse of dimensionality. There are a number of different algorithms that can perform nearest neighbor searching, but in this experiment, we use Locality-Sensitive Hashing.

4.2 Introduction to LSH

Locality-Sensitive Hashing (LSH) is a hashing technique that aimsg to preserve the local relations of the data while significantly reducing the dimensionality of the dataset. LSH works on the principle that if there are two points in feature space closer to each other, they are very likely to have same hash.

4.3 Dimension Reduction

Locality-sensitive hashing method takes advantage of some mechanism of dimension reduction. In our implementation, we use Random Projection Method to achieve this reduction.
Consider a image dataset matrix D with n vectors of size d. This database D can be projected onto a lower dimensional space with n vectors of size k using a random projection matrix.

4.4 Steps to Design the LSH table

1. Create k random vectors of length d each, where k is the size of bit-wise hash value and d is the dimension of the feature vector.
2. For each random vector, compute the dot product of the random vector and the observation. If the result of the dot product is positive, assign the bit value as 1 else 0
3. Concatenate all the bit values computed for k dot products
4. Repeat the above two steps for all observations to compute hash values for all observations
5. Group observations with same hash values together to create a LSH table

Limitations Because of the randomness, it is not likely that all similar items are grouped correctly. To overcome this limitation, a common practice is to create multiple hash tables and consider an image a to be similar to image b, if they are in same bin in at least one of the tables. It is also worth noting that multiple tables generalize the high dimensional space better and amortize the contribution of bad random vectors.
In practice, the number of hash tables and size of the hash value k are tuned to adjust the trade-off between recall and precision.
In our code, we have the number of hash tables be 3 and the bit length of the hash values be 50(a rather large size for a hash value). We tried using a small size of hash value, but it turns out that 50 is good enough for this small dataset.

Another limitation induced by the randomness is that the accuracy of this approach of solving object classification is not stable.

5 Conclusion

In conclusion, our project examined two major methods of predicting the classification of a image input. With different modification of CNN models, the best performing model out of the five constructed is the last model with two cycles of convolutional layers with dropouts.

Notice that in the second method, LSH method, as analyzed at the end of the last section, the accuracy of solving object classification by LSH is not stable because of the randomness imposed by the random projection matrix. The accuracy of this method range from 30 percent to 60 percent. Therefore, we would not include the exact accuracy of this method here.

By comparing this CNN model with the model using Nearest Neighbor Algorithm with LSH, the CNN model is superior in terms of accuracy, where the average category accuracy is over 80 percent. But in terms of time complexity, the CNN model need hours to train while LSH computes in a much shorter time. Considering the advantages and disadvantages of both models , if memory and processing power are sufficient, CNN is more accurate. If the algorithm is hosting on a more portable and less powerful machine, LSH is the more feasible approach.

6 Reference

https://santhoshhari.github.io/Locality-Sensitive-Hashing/
https://necromuralist.github.io/neural_networks/posts/image-to-vector/
https://docs.python.org/3/tutorial/venv.html
http://www.vision.caltech.edu/Image_Datasets/Caltech101/
https://stackoverflow.com/questions/48121916/numpy-resize-rescale-image
https://github.com/bhavul/Caltech-101-Object-Classification