Advances in Data Science and Architecture

Brand Logo Visibility using Convolutional Neural Network

Niranjhani Vasudevan, Gaurang Davda, Divya Priya Emmanuel

{vasudevan.n, davda.g, emmanuel.di}@husky.neu.edu

Table of Contents

Abstract

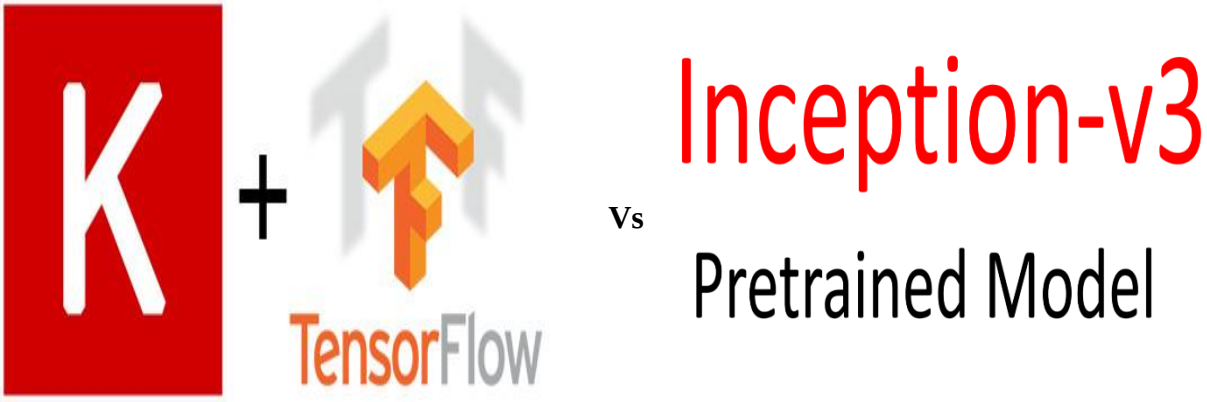
The advertising and branding industry has become more crowded than it previously was and it is not just increasing but increasing exponentially which is a major cause of concern for all those who also want their branding and advertising campaigns to reap expected benefits for them. Every company logo is the first and most important branding tool and it has the power to effectively register an overhaul in your company's recognition and success. Logo is what defines brand and conveys the ideology and purpose of the brand to the consumer. If it's instantly recognizable, then rest assured, the sales will look north. Our project recognises logo present in any given image. The image logo classification was built in two different models of neural network for comparison. Later, the project was deployed in EC2 instance to create web application with user interface.

Introduction

The main concept behind the project is Image Classification using Convolutional neural network (CNN). Image Classification is one of the core problems in Computer Vision that, despite its simplicity, has a large variety of practical applications. A CNN consists of one or more convolutional layers, often with a subsampling layer, which are followed by one or more fully connected layers as in a standard neural network. Why are we choosing CNN?

- *Ruggedness to shifts and distortion in the image*: Detection using CNN is rugged to distortions such as change in shape due to camera lens, different lighting conditions, different poses, presence of partial occlusions, horizontal and vertical shifts, etc. However, CNNs are shift invariant since the same weight configuration is used across space.
- *Fewer memory requirements*: In the convolutional layer, the coefficients are used across different locations in the space, so the memory requirement is drastically reduced.
- *Easier and better training*: Assuming perfect training, we can design a standard neural network whose performance would be same as a CNN. But in practical training, a standard neural network equivalent to CNN would have more parameters, which would lead to more noise addition during the training process. Hence, the performance of a standard neural network equivalent to a CNN will always be poorer.

So, we use CNN for image classification training models with 27 different logos. We compare the result from different model which was trained for same number data using two different approach. One model consists of Keras with tensor flow backend and the other is implementing transfer learning using Google's popular Inception v3 model.



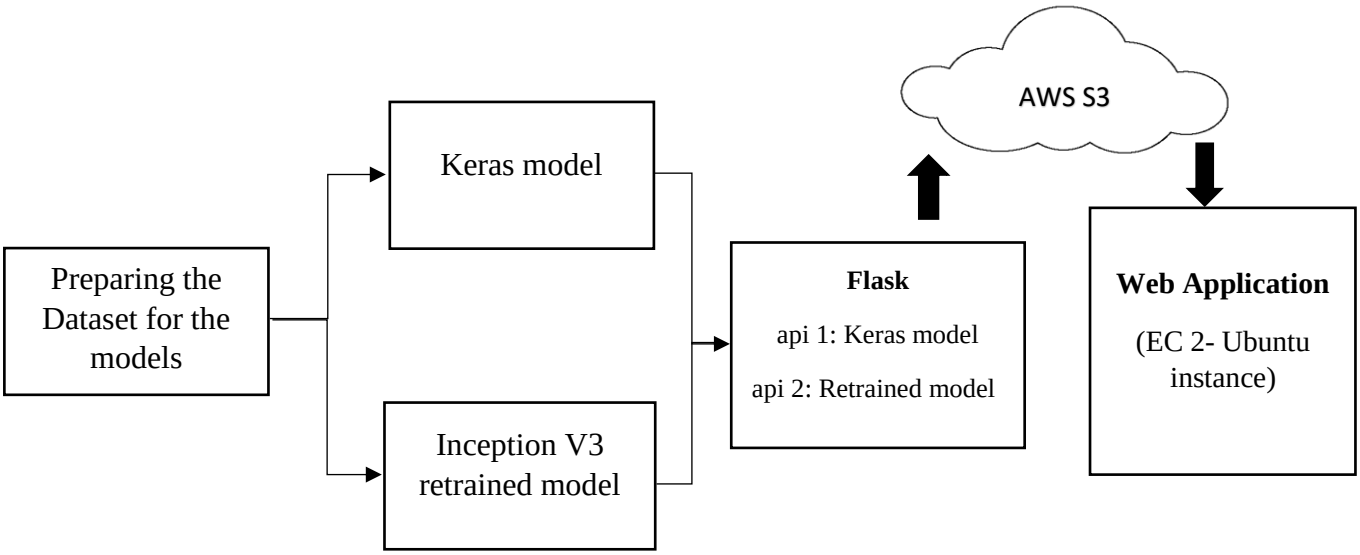
Two models used in Brand logo visibility

Related work

Image classification plays an important role in computer vision, it has a very important significance in our study, work and life. Image classification is process including image pre-processing, image segmentation, key feature extraction and matching identification. With the latest figures image classification techniques, we not only get the picture information faster than before, we apply it to scientific experiments, traffic identification, security, medical equipment, face recognition and other fields. During the rise of deep learning, feature extraction and classifier has been integrated to a learning framework which overcomes the traditional method of feature selection difficulties. The idea of deep learning is to discover multiple levels of representation, with the hope that high-level features represent more abstract semantics of the data. One key ingredient of deep learning in image classification is the use of Convolutional architectures.

Convolutional neural network design inspiration comes from the mammalian visual system structure [1]. Visual structure model based on the cat visual cortex was proposed by Hubel and Wiesel in 1962. The concept of receptive field has been proposed for the first time. The first hierarchical structure Neocognition used to process images was proposed by Fukushima in 1980. The Neocognition adopted the local connection between neurons, can make the network translation invariance. Convolutional neural network is first introduced by LeCun in [1] and improved in [2]. They developed a multi-layer artificial neural network called LeNet-5 which can classify handwriting number. Like other neural network, LeNet-5 has multiple layers and can be trained with the backpropagation algorithm [3]. However, due to the lack of large training data and computing power at that time. LeNet-5 cannot perform well on more complex problems, such as large-scale image and video classification. Since 2006, many methods have been developed to overcome the difficulties encountered in training deep neural networks. Krizhevsky propose a classic CNN architecture Alexnet [4] and show significant improvement upon previous methods on the image classification task. With the success of Alexnet [4], several works are proposed to improve its performance. ZFNet [5],VGGNet [6] and GoogleNet [7] are proposed. In recent years, the optimization of Convolutional neural network is mainly concentrated in the following aspects: the design of Convolutional layer and pooling layer, the activation function, loss function, regularization and Convolutional neural network can be applied to practical problems.

Methodology



Methodology of Brand logo visibility project Implementation

The first phase is to create the dataset in which the model can be trained with. It needs to undergo pre-processing before reaching the input layer of the models. In the second phase, we run the same dataset in two different models. For keras model, the model is saved as .h5 file and for the retrained model its saved as .pb file. Both the saved models are loaded in flask as the third phase. At last phase, we implement a web application which uses all the models from cloud storage "Amazon S3".

Dataset

The *Flickr Logos 27* dataset is an annotated logo dataset downloaded from Flickr and contains more than four thousand classes in total. It consists of three image collections/sets.

The *training set* contains 810 annotated images, corresponding to 27 logo classes/brands (30 images for each class). All images are annotated with bounding boxes of the logo instances in the image. We allow multiple logo instances per class image. The training set is randomly split in six subsets, each one containing five images per class.

The *distractor set* contains 4207 logo images/classes, that depict, in most cases, clean logos. All images come from the Flickr group Identity + Logo Design. Each one of the distractor set images defines its own logo class and we regard the whole image as bounding box.

Finally, the *query set* consists of 270 images. There are five images for each of the 27 annotated classes, summing up to 135 images that contain logos. Furthermore, the query set contains 135 Flickr images that do not depict any logo class, giving 270 test images in total.

The brands included in the dataset are: Adidas, Apple, BMW, Citroen, Coca Cola, DHL, Fedex, Ferrari, Ford, Google, Heineken, HP, McDonalds, Mini, Nbc, Nike, Pepsi, Porsche, Puma, Red Bull, Sprite, Starbucks, Intel, Texaco, Unisef, Vodafone and Yahoo. Flickr 27 is far from enough. Another major challenge we face is that among those 944 images (809 training + 135 query), we have 313 different image sizes, ranging from as small as 22x90 pixels, to as large as 500x500 pixels. Some other challenges include partial logo (missing part), tiny logo, various background (real-world setting vs. white background) glares, etc., which all add more difficulties for our logo recognition system. Hence, we resize all images to 64 x 64 images. Inorder to train model with huge dataset data augmentation was performed since the we had was a tiny dataset of image logos.

After Augmentation, each logo consists of ~500 images. Totally the dataset consists of 13500 images.

Transfer learning – Inception V3

Modern image recognition models have millions of parameters. Training them from scratch requires a lot of labelled training data and a lot of computing power (hundreds of GPU-hours or more). Transfer learning is a technique that shortcuts much of this by taking a piece of a model that has already been trained on a related task and reusing it in a new model. We are using **retrain.py** from below github:

https://github.com/tensorflow/hub/raw/master/examples/image_retraining/retrain.py

This script loads the pre-trained module and trains a new classifier on top for the logo photos that was pre-processed. The transfer learning is that the lower layers that have been trained to distinguish between some objects can be reused for many recognition tasks without any alteration. The first phase analyses all the images on disk and calculates and caches the bottleneck values for each of them. 'Bottleneck' is an informal term we often use for the layer just before the final output layer that does the classification. Because every image is reused multiple times during training and calculating each bottleneck takes a significant amount of

time, it speeds things up to cache these bottleneck values on disk so they don't have to be repeatedly recalculated. The script will write out the new model trained on your categories to /tmp/output_graph.pb, and a text file containing the labels to /tmp/output_lables.txt.

In-order to test our retrained model, we used *lable_image.py* provided by Inception V3. But the code required lot of changes before it started running. Below the sample unseen image passed to the model to tests its prediction :

Sample unseen image passed to model

The model read the image and produced below output:

```
D:\tmp>label_image.py
C:\ProgramData\Anaconda3\lib\site-packages\h5py\__init__.py:36: FutureWarning: Conversion of the second argument of issu
bdtype from `float` to `np.floating` is deprecated. In future, it will be treated as `np.float64 == np.dtype(float).type
`.
  from ._conv import register_converters as _register_converters
2018-04-27 19:37:02.665135: I T:\src\github\tensorflow\tensorflow\core\platform\cpu_feature_guard.cc:140] Your CPU suppo
rts instructions that this TensorFlow binary was not compiled to use: AVX2
2018-04-27 19:37:03.386450: W T:\src\github\tensorflow\tensorflow\core\framework\op_def_util.cc:343] Op BatchNormWithGlo
balNormalization is deprecated. It will cease to work in GraphDef version 9. Use tf.nn.batch_normalization().
coco cola 0.7938579
puma 0.03797272
mcdonalds 0.034322873
yahoo 0.022634603
red bull 0.021577831
```

Classification result

The model shows result of the top-five classification for a given image. Our sample image was classified as **"coca cola"** logo with **79%** accuracy. But at the same time when given an image without a logo, below was the result:

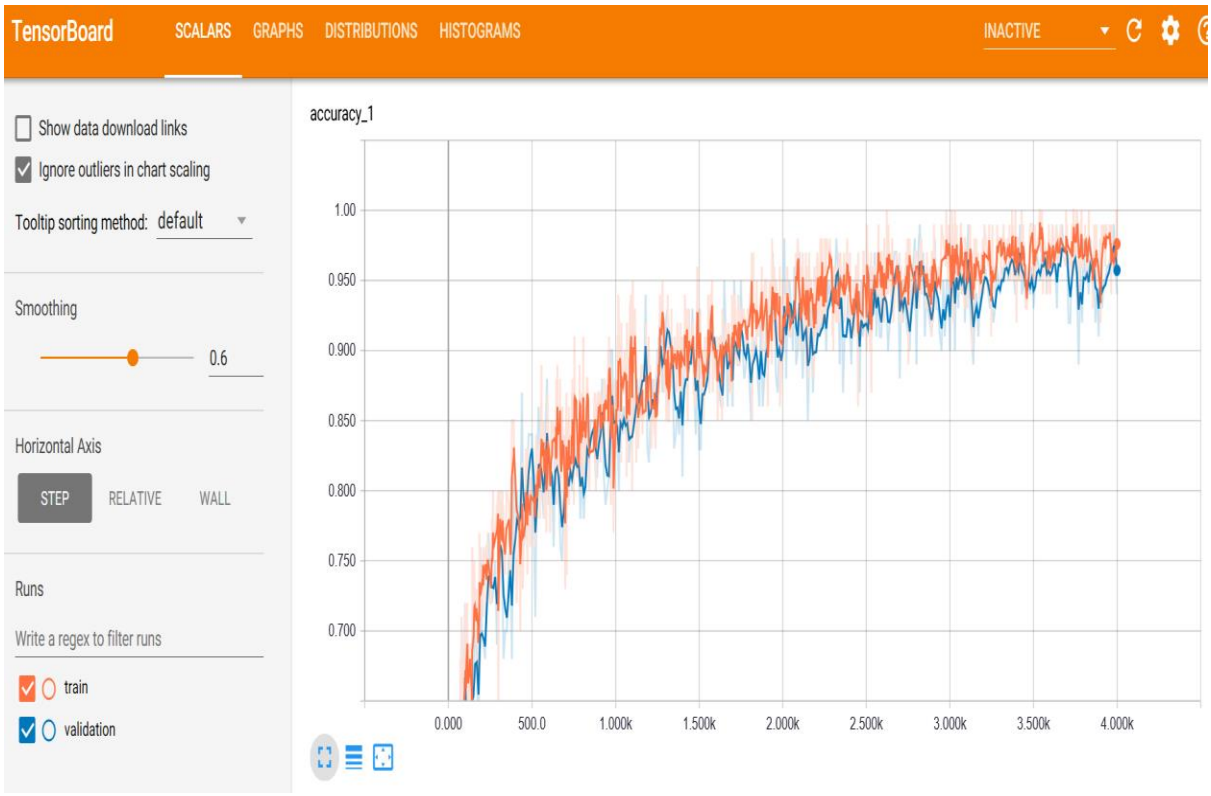
Sample image for testing without logo

```
D:\tmp>label_image.py
C:\ProgramData\Anaconda3\lib\site-packages\h5py\__init__.py:36: FutureWarning: Conversion of the second argument of issu
bdtype from `float` to `np.floating` is deprecated. In future, it will be treated as `np.float64 == np.dtype(float).type
`.
  from ._conv import register_converters as _register_converters
2018-04-27 20:05:40.227166: I T:\src\github\tensorflow\tensorflow\core\platform\cpu_feature_guard.cc:140] Your CPU suppo
rts instructions that this TensorFlow binary was not compiled to use: AVX2
2018-04-27 20:05:40.995489: W T:\src\github\tensorflow\tensorflow\core\framework\op_def_util.cc:343] Op BatchNormWithGlo
balNormalization is deprecated. It will cease to work in GraphDef version 9. Use tf.nn.batch_normalization().
unicef 0.17470534
heineken 0.12153662
texaco 0.106464826
puma 0.10057231
nike 0.07512183
```

Result for unseen data without logo

There is very bad accuracy since the model was not trained with data without logos. But still it classifies with some classes because of the similarity of trained picture with this with unseen logo image. In this case, the "cloud" part of the image was trained when training images with logo. The model has our newly retrained layer on the top of the convolutional neural network. The model can be well trained if we have sufficient number of images.

The model created a “tmp” on the root directory where the saved model and required files for supporting the training of the images generated. From the “retrain_logs”, we were able to visualise the accuracy.

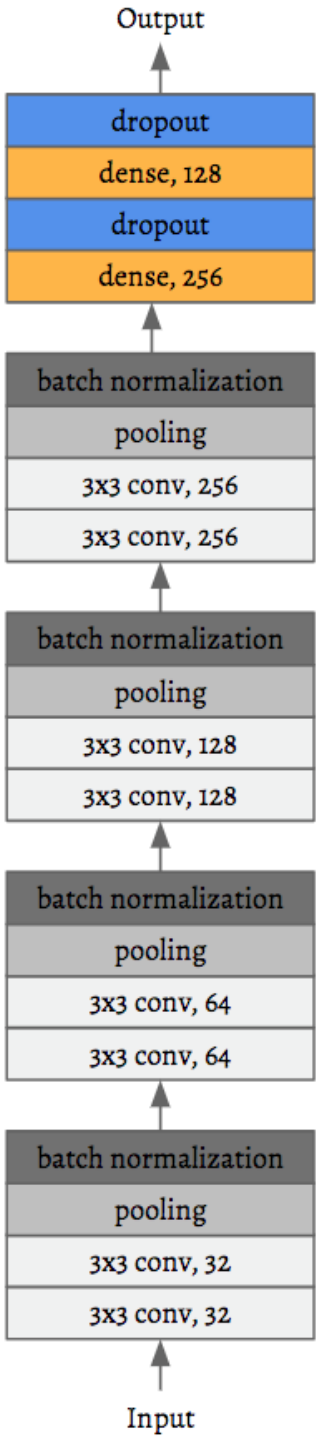


TensorBoard graph for flickr-27 dataset after training

This is used to understand, debug, and optimize the retraining.

Keras model for Image Classification

This is the model designed by us to compare Image classification accuracy with Inception V3. Below is the architecture of our model:

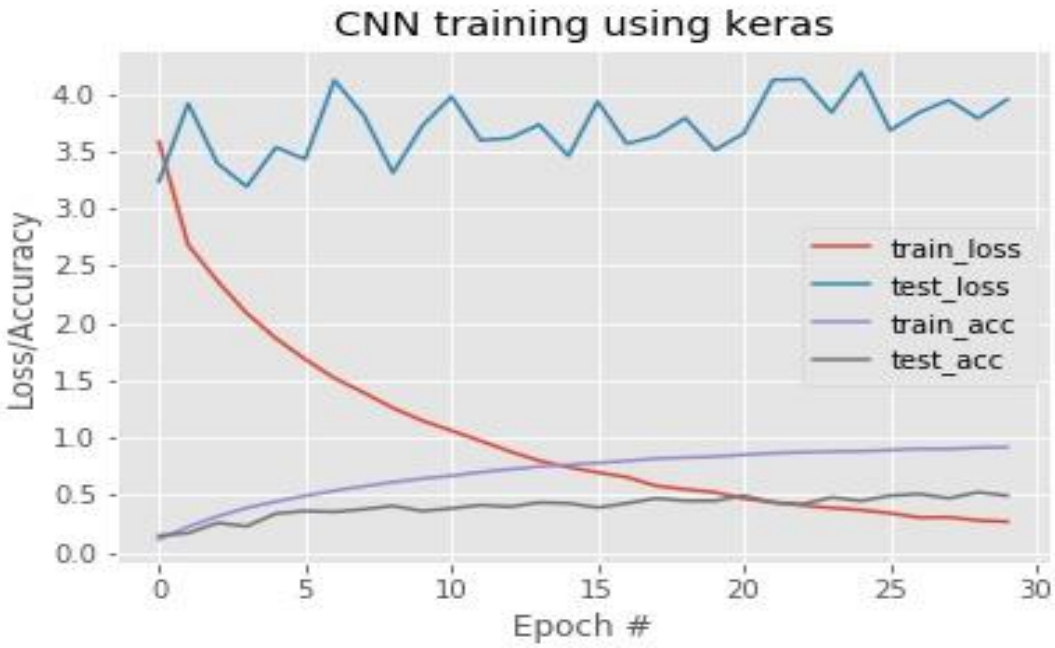


Architecture of keras model build for Brand logo visibility

We used 50 epochs and trained the model in Northeastern Discovery cluster since GPU resource were not available at within a week.

```
def createModel():
    model = Sequential()
    model.add(Conv2D(64, 3, padding='same', activation='relu', input_shape=(64, 64, 3)))
    model.add(Conv2D(64, 3, activation='relu'))
    model.add(MaxPooling2D(pool_size=(3, 3),strides=2 ))
    model.add(Dropout(0.25))

    model.add(Conv2D(128, 3, padding='same', activation='relu'))
    model.add(Conv2D(128, 3, activation='relu'))
    model.add(MaxPooling2D(pool_size=(3, 3), strides=2))
    model.add(Dropout(0.25))

    model.add(Conv2D(256, 3, padding='same', activation='relu'))
    model.add(Conv2D(256, 3, activation='relu'))
    model.add(MaxPooling2D(pool_size=(3, 3), strides=2))
    model.add(Dropout(0.25))

    model.add(Conv2D(512, 3, padding='same', activation='relu'))
    model.add(Conv2D(512, 3, activation='relu'))
    model.add(MaxPooling2D(pool_size=(3, 3), strides=2))
    model.add(Dropout(0.25))

    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(27, activation='softmax'))
    return model
```

Training all 13500 images the result produced by the model is below:



Accuracy and loss graph

The graph clearly shows the overfitting of out model when observing the training and testing accuracy plot. In order to overcome this, we should have trained the model with more number of images which parallelly requires a lot of GPU resource to train.

Deployment

Both the trained models where deployed in EC2 instance with required user Interface. The web application was secured with different logins and each login had its own functionality.

Conclusion

We conclude the research by stating, Inception V3 classifies better than Keras model build by us. The keras model requires more data of images which might in turn require more GPU than the GPU we used to train in Discovery cluster.

Future Scope

The deployment can be performed for multiple images for company logo analytics used by consumers in social media by downloading images from their public post. We can even perform analysis based geolocation based consumer usuage from the location shared in their post.

References

[1] LecunY, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11):2278-2324.

[2] Cun Y L, Boser B, Denker J S, et al. Handwritten digit recognition with a back-propagation network[C]// Advances in Neural Information Processing Systems. Morgan Kaufmann Publishers Inc. 1990:465.

[3] Hecht-Nielsen R. Theory of the backpropagation neural network[M]// Neural networks for perception (Vol. 2). Harcourt Brace & Co. 1992:593-605 vol.1.

[4] Krizhevsky A, Sutskever I, Hinton G E. ImageNet Classification with Deep Convolutional Neural Networks[J]. Advances in Neural Information Processing Systems, 2012, 25(2):2012.

[5] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in ECCV, 2014.

[6] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in ICLR, 2015. [7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with Convolutionals,” Co RR, vol. abs/1409.4842, 2014.