INFO 7390 Final Project Report 08-17-2018

Advances in Data Science and Architecture

Emblem Detection for Commercial Research

Dinesh Maria Anthony, Kanika Nama, Inchara Niyanta

(packianathanjerome.d **,** nama.k, niyanta.i) @husky.neu.edu

INFO 7390, Summer 2018, Northeastern University

TABLE OF CONTENTS

Abstract-

In this paper, a complete logo detection/ recognition system for document images are proposed. In the proposed system, first, a logo detection method is employed to detect a few regions of interest (logo-patches), which likely contain the logo(s), in a document image. The detection method is based on the Convolutional neural network(CNN). For the logo recognition, a template-based recognition approach is proposed to recognize the logo which may present in every detected logo-patch. The proposed logo recognition strategy uses a search space reduction technique to decrease the number of template logo-models needed for the recognition of a logo in a detected logo-patch. The features used for search space reduction are based on the geometric properties of a detected logo-patch. Based on our experimentations on 16000 document images of Flickr dataset, 99.31% of the logos were detected as logo-patches. Among the detected logo 97.90% of logos were recognized. Considering both logo detection and recognition results, 97.22% of the logos in the document images could truly be detected/recognized as the overall performance of the proposed system. Later the project was deployed in EC2 instance to create web application with user interface.

Introduction-

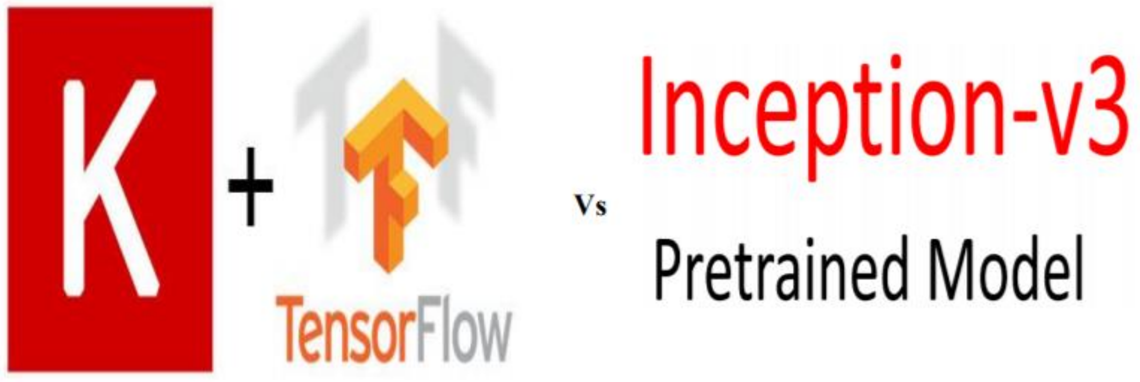
The main concept behind the project is Image Classification using Convolutional neural network (CNN). Image Classification is one of the core problems in Computer Vision that, despite its simplicity, has a large variety of practical applications. A CNN consists of one or more convolutional layers, often 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 preprocessing, 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.

[1] Propose an effective and scalable framework for recognizing logos in images which is based on a method for encoding and indexing the relative spatial layout of local features detected in the images with logos. Based on the analysis of the local features and other spatial structures we use an automatic method for constructing a model for each logo-class out of multiple training images. This allows them to detect logos under varying conditions such as perspective tilt.

[2] View the problem of logo recognition as an instantiation of the broader problem of object recognition. We divide the overall problem into four sub-problems: logo classification, logo localization, logo detection without localization, and logo detection with localization. Using this we are able to achieve 90% accuracy and establish the state of the art in logo recognition. This problem is also very similar to reading text in the wild like house numbers, for example. Thus, we can expect that approaches from that domain with a little adaptation would work very well for the task of logo recognition

[3] Use an approach as follows: word bounding box proposal generation, proposal filtering and adjustments, text recognition and final merging. The detection stage is done using the CNN object detection framework in which a region proposal is converted into a fixed size to which we can apply a CNN. This helps avoid the computationally expensive task of scanning the full image for text. Before applying the CNN, we use various classifiers to filter out the large number of false-positive region proposals. Once we have filtered out some of the region proposals and only retain ones that are highly likely to contain text, we apply whole-word approach for recognition by providing the entire region proposal to a deep convolutional neural network.

Dataset

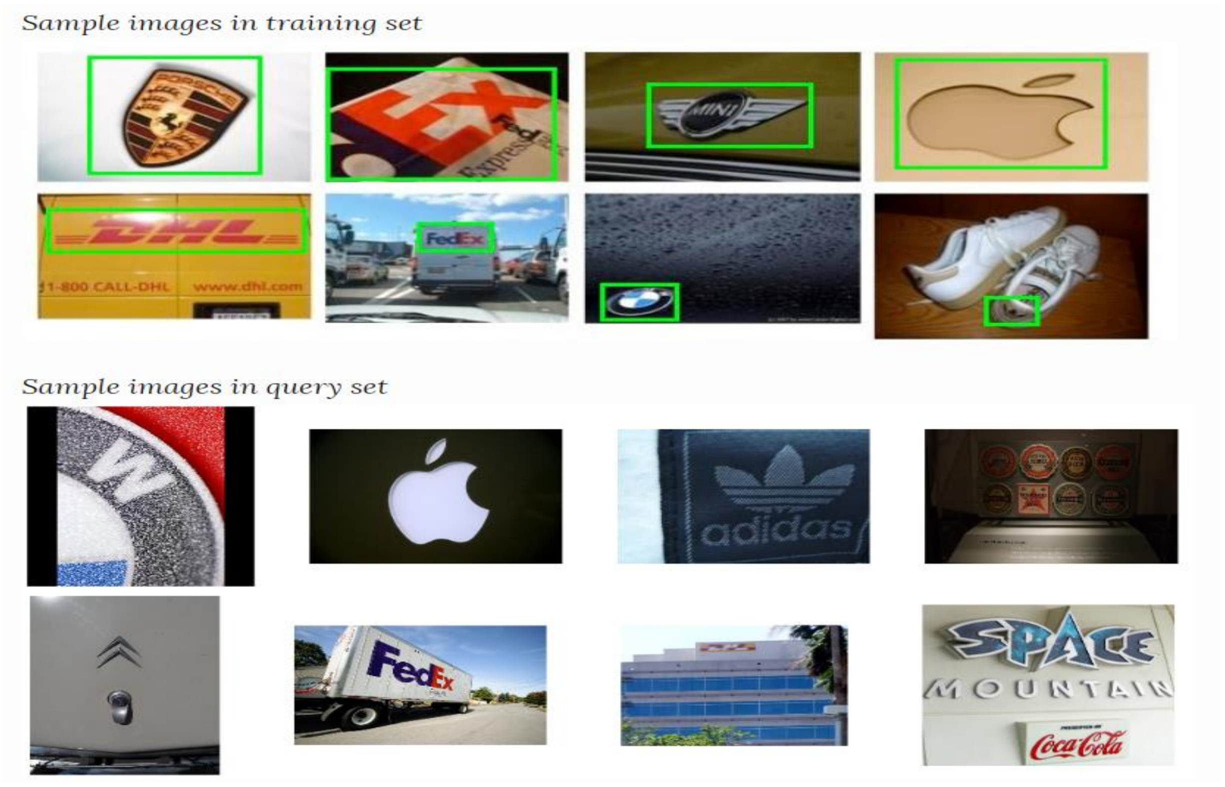
The Flickr Logos 27 dataset is an annotated logo dataset downloaded from Flickr and contains more than four thousand classes in total. It consists of three image collections/sets.

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.

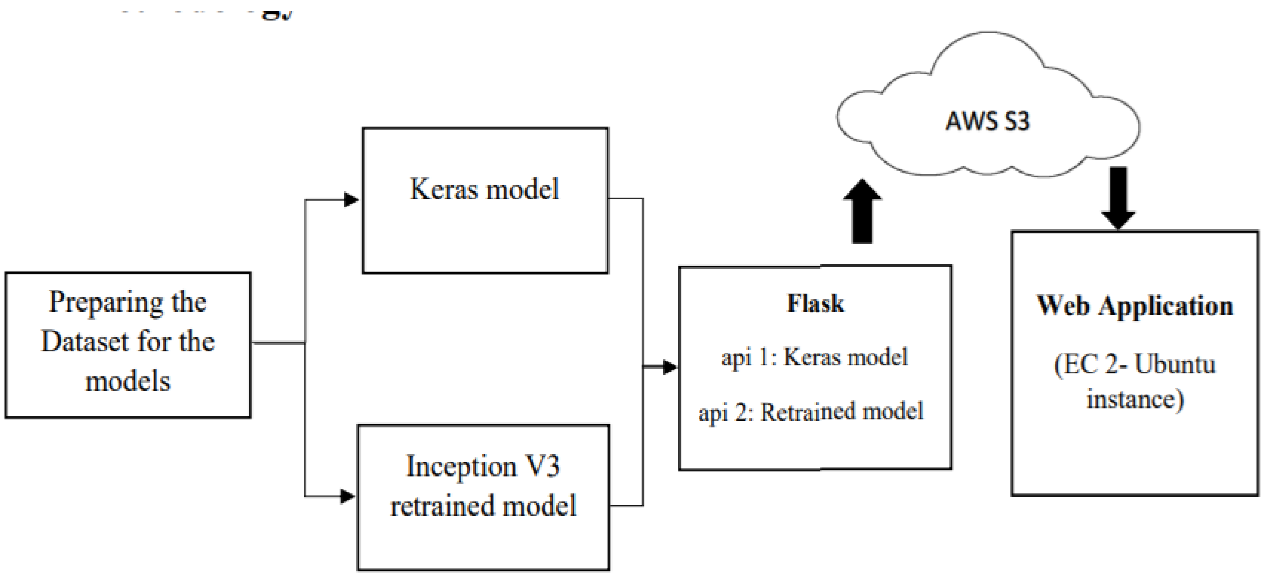
The distractor set contains 4207 logo images/classes, that depict, in most cases, clean logos. All images come from the Flickr group Identity + Logo Design. Each one of the distractor set images defines its own logo class and we regard the whole image as bounding box.

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, UNICEF, 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. 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 16000 images.



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”.

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

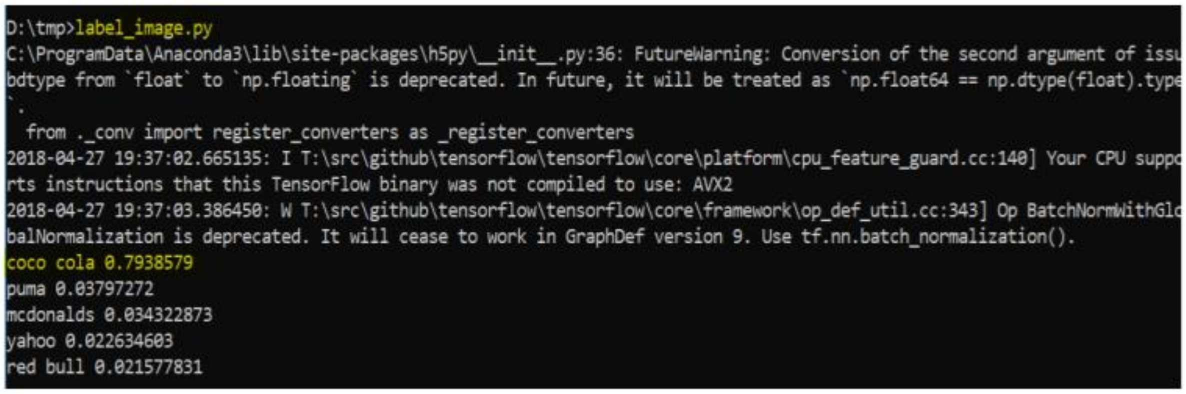
This script loads the pre-trained module and trains a new classifier on top for the logo photos that was pre-processed. The transfer learning is that the lower layers that have been trained to distinguish between some objects can be reused for many recognition tasks without any alteration.

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

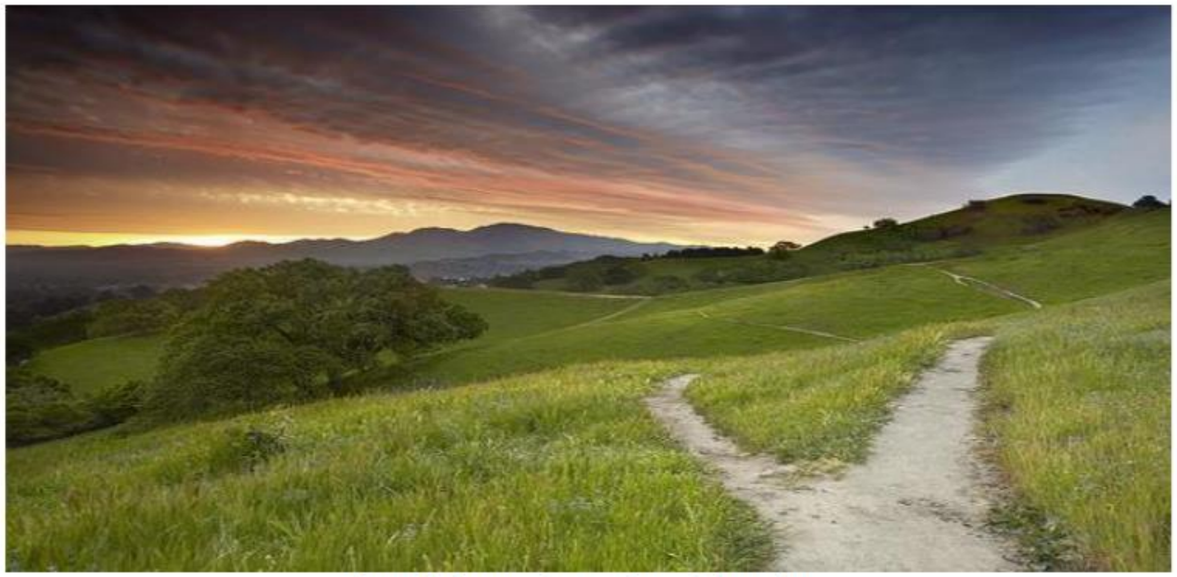
The model read the image and produced below output:

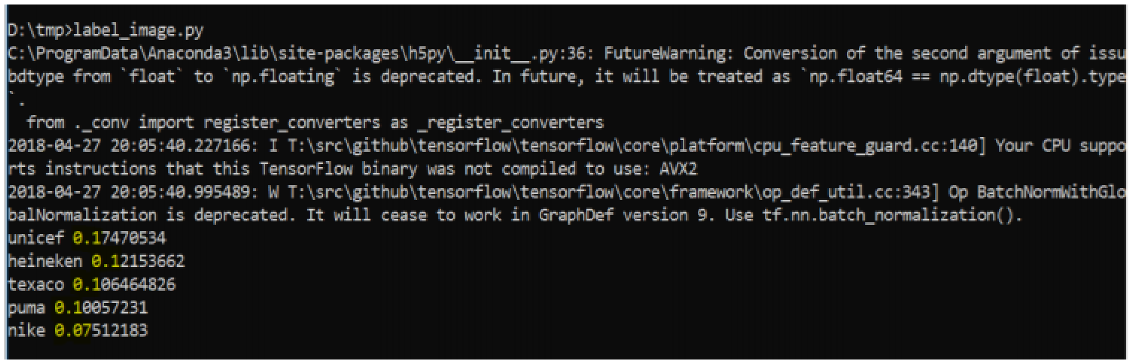


```
D:\tmp>label_image.py
C:\ProgramData\Anaconda3\lib\site-packages\h5py\__init__.py:36: FutureWarning: Conversion of the second argument of issu
bdtype from `float` to `np.floating` is deprecated. In future, it will be treated as `np.float64 == np.dtype(float).type
`.
  from ._conv import register_converters 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

Model development

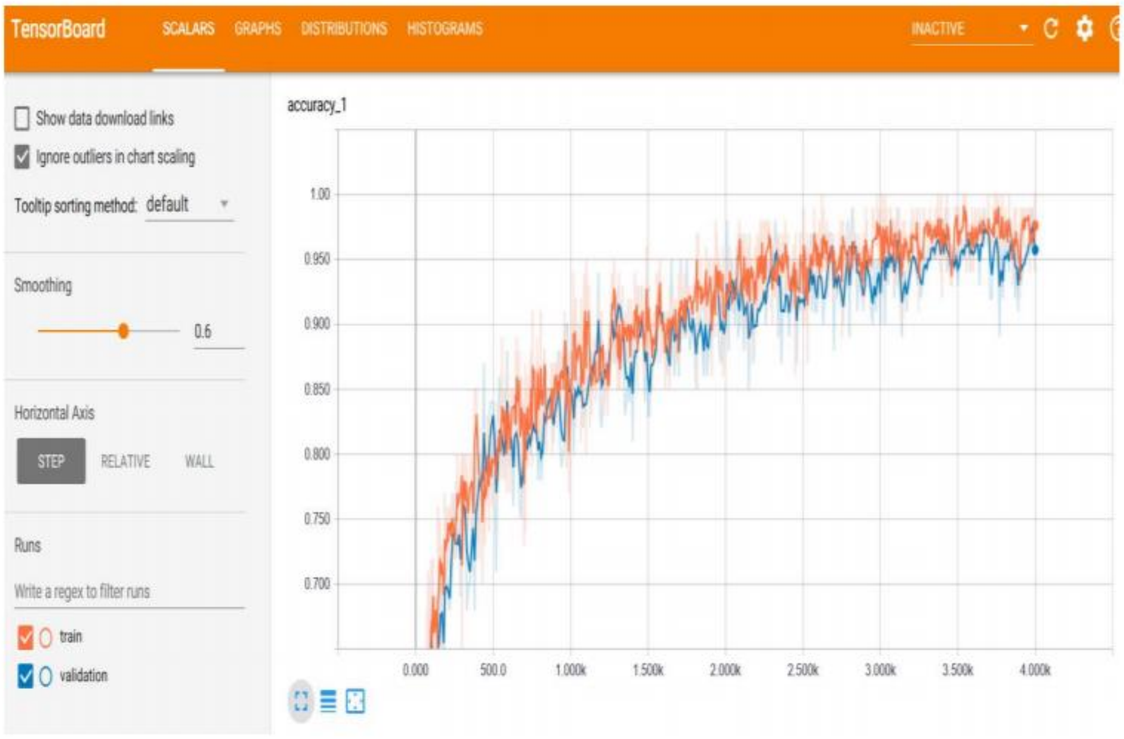
Most of my time was spent iterating on developing the logo detection model. The main steps included modifying the out-of-the-box Inception v3 model and associated retraining script in TensorFlow, retraining the modified model, analyzing model performance, and tuning model parameters accordingly.

Some of the major modifications to the out-of-the-box Inception v3 model and associated retraining script in TensorFlow I made were:

- **Retrained the final model layer** to categorize `logo` vs. `no-logo`
- **Added a dropout layer** to minimize overfitting.
- **Up-sampled** `logo` class of images to improve precision and recall for the unbalanced class.
- **Removed random sampling for validation and test sets** from the script which originally distorted the results.
- **Added Tensor Board summaries** to be able to visualize model training.

Evaluation of model

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 visualize the accuracy.

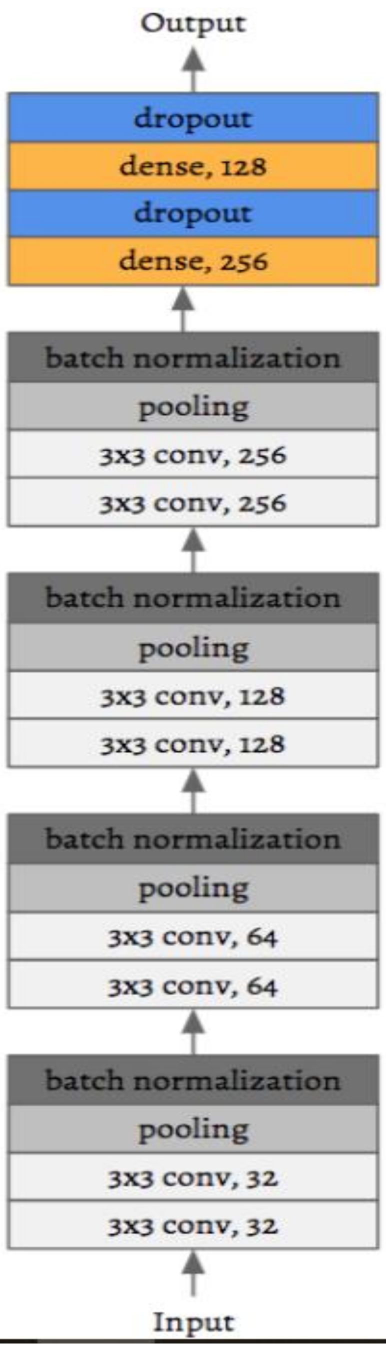


Tensor Board graph for flickr-27 dataset after training

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

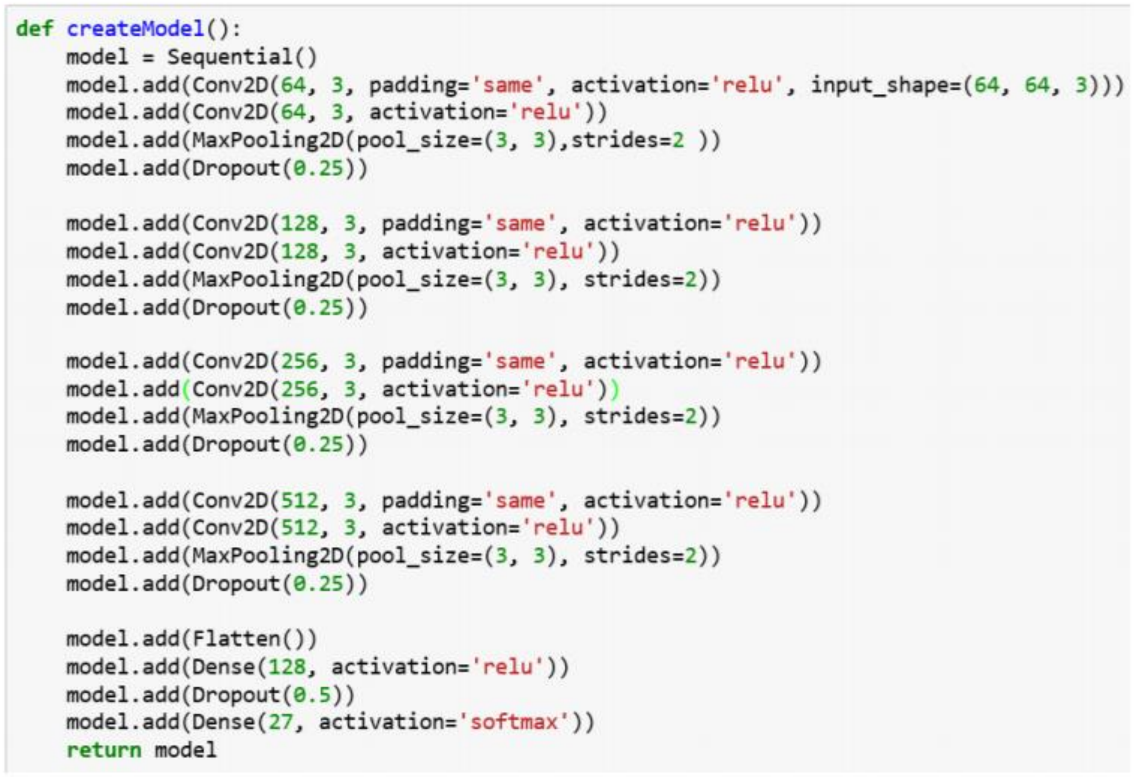
Keras model for Image Classification

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



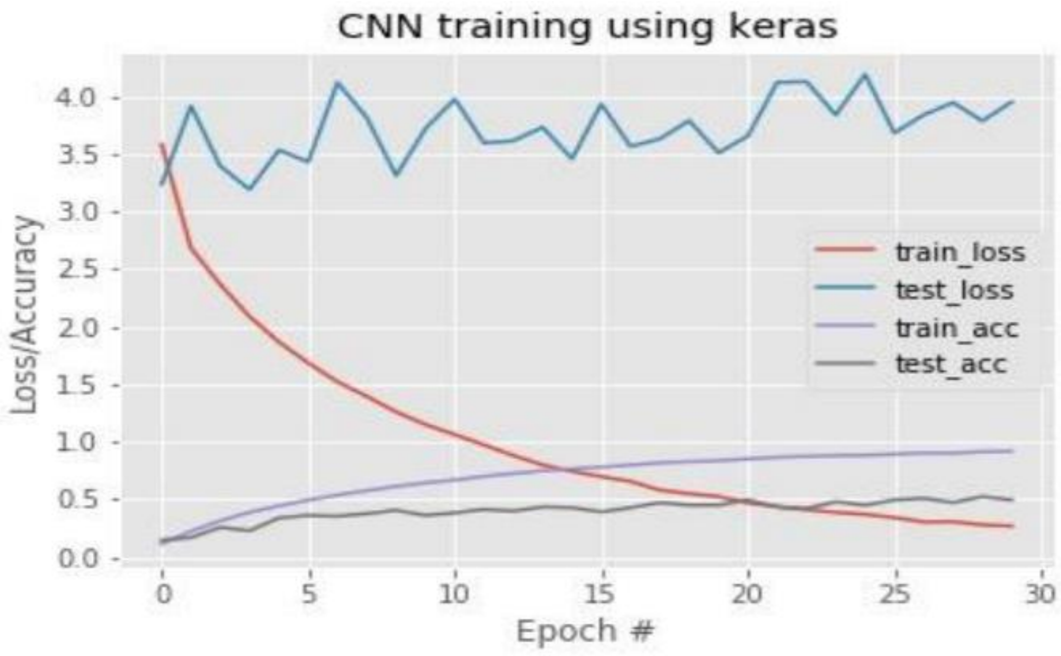
Architecture of keras model build for Brand logo visibility

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 16000 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. 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 usage 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.