Machine Learning Engineer Nanodegree

Capstone Project

Khalid Aljaghthami

February 1st, 2019

Definition

Project Overview

One of the most significant research areas in machine learning is computer vision which is primarily based on the recognition of objects in images and videos. This opens an endless possibility of different kind of objects to be recognized. Detecting those objects could generate an enormous number of different applications leading to improving quality of life (Foo et al., 2018).

Logo detection is a subset of the object detection. Logo detection has been there for a long time and has been used in many multiple industries. For example, the transportation services have been interested in object recognition of vehicle parts such as plate and logo (Foo et al., 2018, Hang S., Hang et al., 2018). Similarly, commercial research is also in logo recognition in help researchers predict brand trend (Hang et al., 2018).

There are many attempts to address such area of problems. Datasets such as "**FlickrLogos**" used Ficker as a data input and "**WebLogo-2M**" used a combination of logo images and icons from the web through a crawler (Hang et al., 2018). However, the **QMUL-OpenLogo** has taken a further step in which it has created **Open Logo Detection Challenge**. The challenge aims to create a platform to tackle the problem of logo detection using a combined dataset from all existing and open datasets (Hang et al., 2018). In this paper, we will participate in the challenge with simplicity in mind given the project goal and the computing resources we have and use their proposed dataset as our data input.

Problem Statement

There are millions of brand logs across the world. The current practice in logo detection system assumes the large training data for every logo class. The Open Logo Detection Challenge took a different approach since this is practically impossible given the introduction of logos every day (Hang et al., 2018). A more reasonable way to try to create a model which could be trained on data where few classes have large training data while many classes has few training data. Therefore, the goal is to create a model which will be trained on varying

training per class and find the mean average precision. The solution should be better or close to the benchmark in the **Open Logo Detection Challenge** which stands at 48.2. Convolution neural networks is well suited for this kind of problems. **In this paper, we will train a model using CNN with different transfer learning techniques and evaluate how the accuracy of the model with dataset of the images without cropping and with cropping given the annotations.**

Metrics

Due to the dataset nature, the metrics selected to measure the performance of the model is Categorical Accuracy. It calculates the mean accuracy rate across all predictions for multiclass classification problems. To optimize the model learning, the training uses log score function as its cost functions.

```
K.cast(K.equal(K.argmax(y_true, axis=-1),K.argmax(y_pred, axis=-1)),K.floatx())
```

Figure 1: Keras implementation of Categorical Accuracy

Analysis

Data Exploration

The **QMUL-OpenLogo** is a challenge platform with 27,083 images and 352 classes. The platform recommends two settings of splitting the data. The first split is Semi-Supervised Split while the other one is Fully Supervised Split. In this paper, the Fully Supervised Split will be used. The dataset characteristics includes that for every class, there is 70% training data and 30% evaluation data. Thus, it can be summarized as follows:

Classes: 352
Training Images: 15,975
Validation images: 2,777
Test Images: 8,331

The challenge tries to emulate a real-world environment in which the images are coloured, arbitrary, existing in varying sizes, dimensions, resolutions and availability per class. The dataset consists of Annotations, Images and Class Set.

Interesting Properties

- Many Classes with low training images

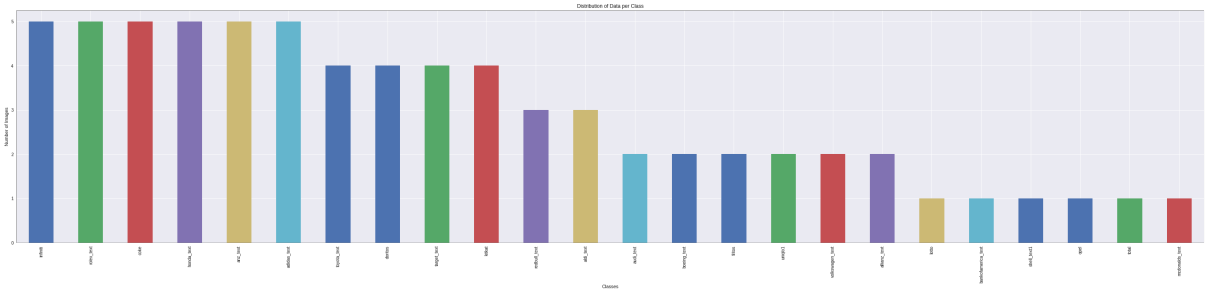


Figure 2: classes with less than 10 training images

The dataset comes with the challenges in which not all logo has a large number of training images. Indeed, there are around 289 logo classes with 10 images or below.

- Logos in Images appear in different sizes

Images comes in different sizes and background. While some the logos can be easily recognized, other are hard to be recognized to its size.

- Statistical descriptive of the original data sources shows the diversity of image sizes and numbers

The dataset is provided a combined dataset. Therefore, it is not possible to identify which image is coming from which dataset. However, the competition paper provide a statistical descriptive about the images.

Dataset	Logos	Images	min∼max (mean) Instances / Class	min∼max (mean) Scale (%)
FlickrLogos-27 [15]	27	810	35∼213 (80.52)	0.0160∼100.0 (19.56)
FlickrLogos-32 [33]	32	2,240	73∼204 (106.38)	0.0200∼99.09 (9.16)
Logo32plus [1]	32	7,830	132∼576 (338.06)	0.0190∼100.0 (4.51)
BelgaLogos [14]	37	1,321	2∼223 (57.08)	0.0230∼69.04 (0.91)
WebLogo-2M(Test) [36]	194	4,318	18∼204 (40.63)	0.0180∼99.67 (7.69)
Logo-In-The-Wild [40]	1196	9,393	1∼1080 (23.49)	0.0007∼95.91 (1.80)
SportsLogo [20]	20	1,978	108∼292 (152.25)	0.0100∼99.41 (9.89)
QMUL-OpenLogo	352	27,083	10∼1,902 (88.25)	0.0014∼100.0 (6.09)

Figure 3: Statistics of the images the original data sources from Hang et al., 2018

However, due to the complexity of this challenge, the lack of computing resources by Colab, which only support 10GB of RAM and for the sake of simplifying the capstone project, the dataset will be limited to only classes which has more than 250 training images. Thus, we will have the following characteristics in our experiment:

Classes: 16
Training Images: 4,443
Validation images: 1,111
Test Images: 1,010

Images

The dataset provides around 27,083 images. All of these images are collected from different existing logo datasets. The images can used for training, validating and testing purposes and all are located under 'JPEGImages' folder. Below are random images selected.

Annotations

The dataset provides additional information about each image. The information includes owner, image size, name and bounding box are shown in the image.

Class Set

The dataset comes with pre-defined split of the data according the competition requirement. It contains information about how data is split in Fully Supervised method including the details about the training and testing data as well as the classes and their associated images.

Exploratory Visualization

The data set comes with annotations which determine where the logo is in the picture. Sometimes, there are multiple annotations per picture as shown above in the images. The dataset comes at varying images per logo class. The plot below shows distribution of the data per class in the training set. It is very clear that the number of images per class are different. This indicates imbalance in the dataset which may have significant impact on the model and its performance.

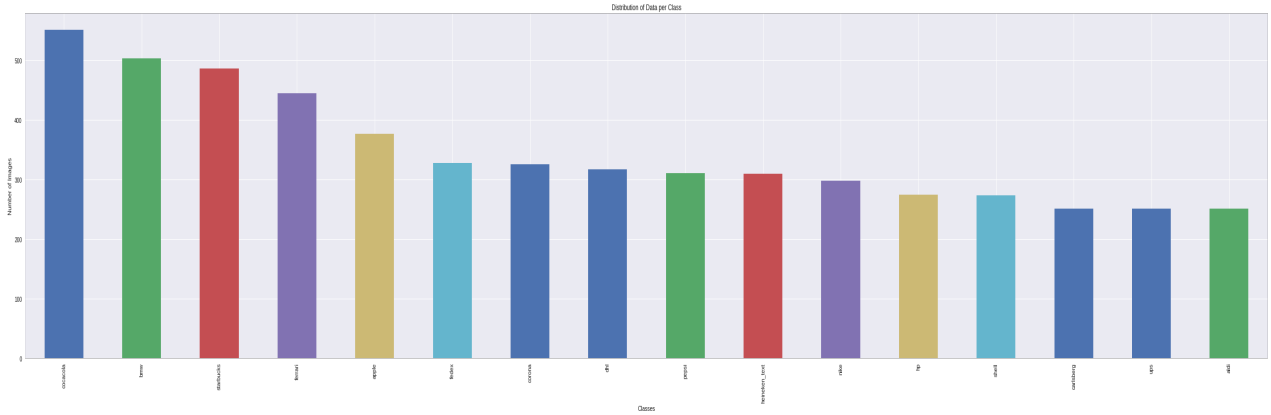


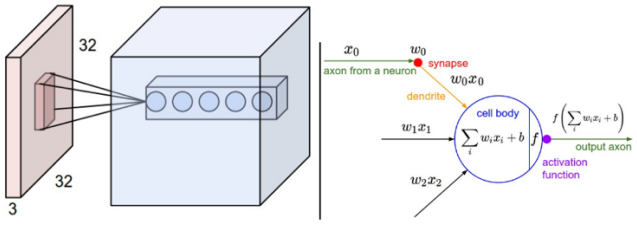
Figure 4: distribution of data per class

Algorithms and Techniques

CNN Model

Convolution Neural Network (CNN) is a class of neural networks and commonly used in computer vision. Similar to any neural networks, CNN consists of neurons with weights and biases. Each neuron receives inputs where they go through operations such as Softmax or Rectified Linear Unit to produce outputs. CNN has a unique assumption where the CNN expects the inputs as image pixels. So that, CNN implement certain operations to reduce the number of parameters in the neural network which has a considerable impact on the complexity. For example, a regular neural network with colored images of size 32x32 as in CIFAR-10 has 3072 weights (32 * 32 * 3) in the first hidden layer. In large image size such as 200x200, it becomes very hard to scale and tends to easily overfit when the number of parameters becomes large.

On the other hand, the CNN has an advantage where each neuron is 3-dimensional including width, height and depth. Each output of a neuron is controlled by depth, stride and padding which are parameters help shape the output. Also, CNN provide a pool layer where it provides a mechanism to downsampling such as reducing 32x32 to 16x16. these a considerable effect on the number of the parameters and overall learning process. The last layer is a fully connected layer which represents the final set of classes to be predicted. An example of CNN network.



Left: An example input volume in red (e.g. a 32x32x3 CIFAR-10 image), and an example volume of neurons in the first Convolutional layer. Each neuron in the convolutional layer is connected only to a local region in the input volume spatially, but to the full depth (i.e. all color channels). Note, there are multiple neurons (5 in this example) along the depth, all looking at the same region in the input - see discussion of depth columns in text below. **Right:** The neurons from the Neural Network chapter remain unchanged: They still compute a dot product of their weights with the input followed by a non-linearity, but their connectivity is now restricted to be local spatially.

Figure 5: Insight into the Convolutional Neural Network from Stanford University
Source: http://cs231n.github.io/convolutional-networks/>

CNN has a weakness in which it performs very well on image classification while its performance is not good for other problems such as NLP related problems. Similarly, CNN only accept images of fixed size. That is, each image has to be of fixed size defined in the network configuration.

CNN in Logo Detection

Convolution Neural Network (CNN) is used as the selected classifier. It is the most commonly used algorithms in computer vision. The **Open Logo Detection Challenge** has relatively large dataset making it a potential fit for deep neural network. The output of the CNN model a predicated class of logo.

In the challenge, all papers used algorithms including YOLO and Faster R-CNN. For the sake of this project, we will explore the possibility of using pure CNN as well as transfer learning technique and its impact on the model result.

To achieve this, there are a selected number of parameters to be tuned to achieve an acceptable level of performance

- Threshold to help manage false positive results
- Training Length through the number of epochs
- Batch Size
- Learning rate
- Image augmentation to double the size of the data
- Transfer learning to benefit from an existing pre-train model
- One-hot encoder for target classification

Benchmark

We will start standard CNN to create a naive accuracy. Based on the first model result, we will work on improving the results to the highest one possible. In addition, as per the challenge leaderboard, the mAP benchmark for the fully supervised dataset is 48.3%. Due to the change in the dataset for this project, we may not be able to achieve similar result, thus it will be replaced with the accuracy of the naive model.

Methodology

Data Preprocessing

There are a number of preprocessing steps conducted in order to explore the data and to prepare the data for training.

- The original dataset comes as a raw data. This means that there is a coding required to connect to different pieces together such as linking between the annotations and the images which they reside in two different folders.

  Preprocessing Steps:

  1. Create a function to load annotation folder's files and JPEGImage folder's file name and construct a mapping between the class name, actual image and its annotation.

- The dataset comes pre-split for the training set and the testing set. However, a validation set from the training set should generated.

  Preprocessing Steps:

  1. Create a function to load the image names of training set and the testing set.
  2. Of the training set, create a validation set which consists of 20%

- Due to the memory limit of 10GB on Colab, the platform selected to run the training, the dataset to be reduced to only include classes which contain at least 250 training images.

  Preprocessing Steps:

  1. For all logo classes, exclude classes which have a training image below than 250 images
  2. Verify the testing does not images for the excluded classes

- This project will create a model for images without cropping and with cropping and explore its results. Thus, a code is needed to crop all training and testing data using the given annotations per images.

  Preprocessing Steps:

  1. Create a function which will load each image in the JPEGImage folder and exclude the logo from the image based on its corresponding file in the Annotation folder.

- Prepare the necessary steps to import the pre-trained models selected for this experiment. The selected models are ResNet50 and VGG-19.

  Preprocessing Steps:

  1. Download the ResNet50 and VGG-19 pre-trained models manually from their Keras website or use Keras code to download them.

- Prepare images input into the CNN model

  Preprocessing Steps:

  1. The RGB images are to be converted to BGR.
  2. Resize images to 28x28 pixels and create 4D tensor of (1,28,28,3)
  3. For the standard CNN, re-scale the images to 255.
  4. Images to be converted to forms which pre-trained model accept.
  5. The target classes care categorical classes, thus, hot-coded is required for the target classes.

Implementation

The process of generating the model went through a couple of challenges. While some of them are considerably challenging, others are trial and errors such as tuning parameters.

RAM Limit Challenge

The implementation of the code was done on Google Colab, which is a free jupyter like notebook running in the cloud. Colab has a challenge of only allowing 10GB of RAM. This was a nightmare given it is time-consuming to prepare the data and training the model and when it reaches the limit, it crashes and loses everything.

The RAM limit challenge was in loading the data which accounts for around 27,000 images. **The solution** was to reduce the number of images to maximum number which could be supported by the Colab. After trail and errors attempt, it around 7000 images can be used in training the model accounting for 16 logo classes. Therefore, the dataset was reduced, and further checks were applied the dataset to ensure correctness of the data between the training and testing data given that the testing and training data already given by the dataset.

Pure CNN Architecture

The first attempt to address the problem is through training a pure CNN architecture. The model will be used as a bias for benchmarking our refined models. The challenge was tuning our parameters to balance between the training time and the accuracy. The challenges faced were as follow:

1. Convolutional layers
   - Adding more layers enhance the accuracy but increases the possibility of overfitting and also may require more computing time and thus increase the training time.
   - The input shape of images given the various sizes of the images where logo may a small part of it.
   - Dropout percentage for the Conv layers
2. The accuracy metrics

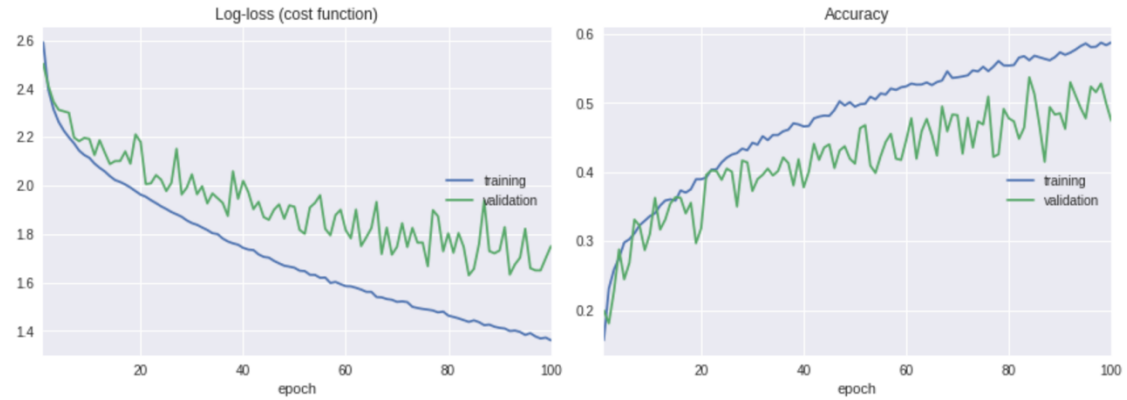
After a number of trails and errors attempts, the final solution for the benchmark is as follow:

1. Loading data
2. Preprocess the data including:
   a. limit the dataset
   b. hot-encoded the logo classes
   c. rescaling the images
3. Define the network architecture as below
   a. Epochs: *100*
   b. Batch Size: *20*
   c. Dropout: *0.3, 0.5 respectively for all ConvNet layers*
   d. Filter: *32 , 128 ,64 respectively for all ConvNet layers*
   e. MaxPooling : *2 for all ConvNet layers*
   f. Activation : *relu for ConvNet, softmax for FC*

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 228, 228, 32)      416
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 114, 114, 32)      0
_________________________________________________________________
dropout_1 (Dropout)          (None, 114, 114, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 114, 114, 128)     16512
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 57, 57, 128)       0
_________________________________________________________________
dropout_2 (Dropout)          (None, 57, 57, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 57, 57, 64)        32832
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 28, 28, 64)        0
_________________________________________________________________
global_average_pooling2d_1 ( (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 16)                1040
=================================================================
Total params: 50,800
Trainable params: 50,800
Non-trainable params: 0
```

4. Compile the model with following characteristcs
    a. Loss function: *categorical_crossentropy*
    b. Metric: *categorical_ accuracy*
5. Train the model.
6. Plot the progress and store the best weight.
7. Test the model against testing data.

Overall, the model has been learning as indicated in the below plot. **The accuracy on testing data is 50.19% and this will be our benchmark.**



Refinement

Following our bias model, we implement further refinement to the improve the learning model. Firstly, transfer learning will be applied using Resnet50 and VGG19, followed by repeating the same steps but with logo images cropped from original images.

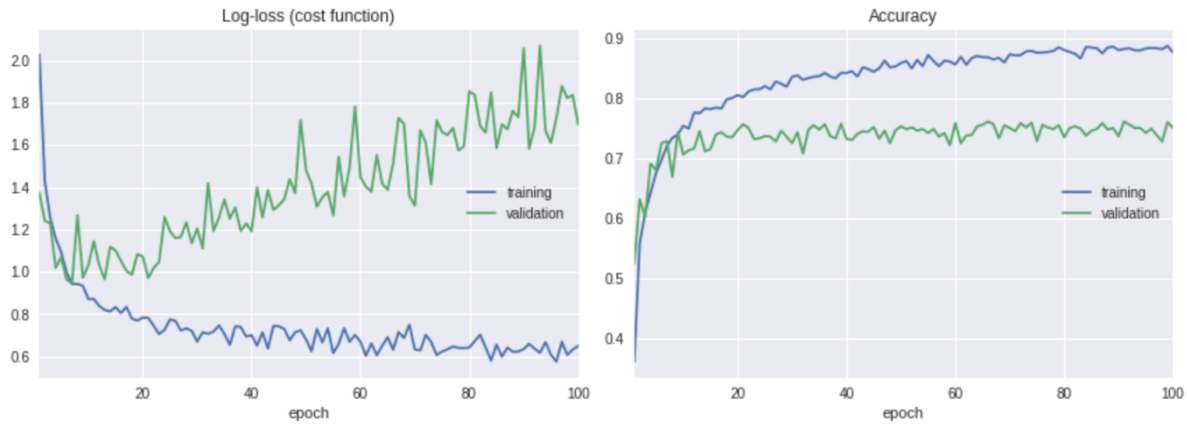
CNN Model Using Resnet 50

With a target to improve the accuracy of the model, the logo images were attached to the Restnet50 pre-trained features and trained on it. The steps implemented to execute the model is similar to the CNN model with following differences:

- Resnet50 pre-trained model is downloaded and features of logo images are extracted.
- The network architecture is as designed as the following:
    - Epochs: *100*
    - Batch Size: *20*
    - Dropout: *0.5 for all hidden layers*
    - Dense Units: *128 , 1025 ,512 respectively for all FC layers*
    - Activation : *relu, for hidden FC layers and softmax for last FC* Layer

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
global_average_pooling2d_1 ( (None, 2048)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               262272
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1025)              132225
_________________________________________________________________
dropout_2 (Dropout)          (None, 1025)              0
_________________________________________________________________
dense_3 (Dense)              (None, 512)               525312
_________________________________________________________________
dense_4 (Dense)              (None, 16)                8208
=================================================================
Total params: 928,017
Trainable params: 928,017
Non-trainable params: 0
```

Overall, the model has been learning as indicated in the below plot. **The accuracy on testing data is 65.24%.**



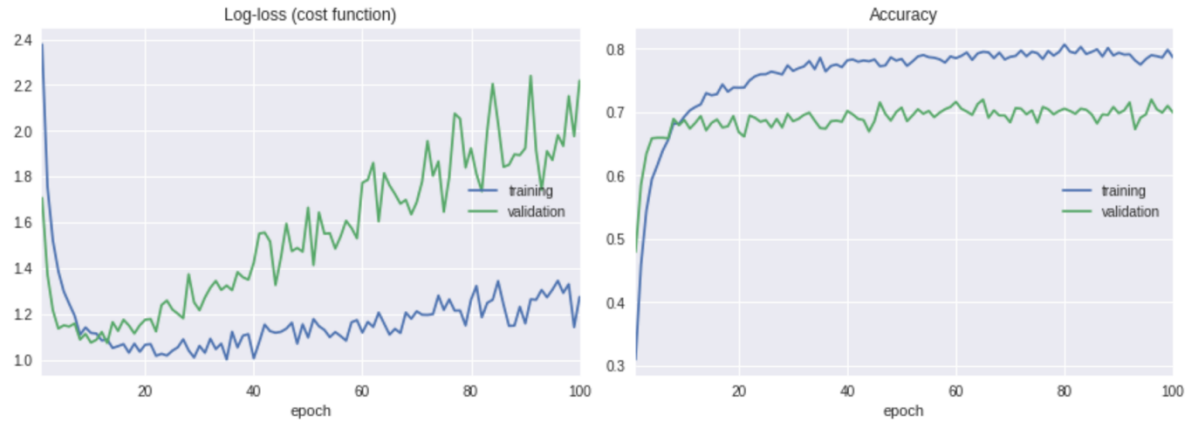
CNN Model Using VGG19

Similar to the previous refinement, the logo images were attached to the VGG19 pre-trained features and trained on it. The steps implemented to execute the model is very similar to the Resnet50 model with following differences:

- Instead of Resnet50, VGG19 pre-trained model is downloaded and features of logo images are extracted.
- The network architecture is as designed as the following:
    - Epochs: *100*
    - Batch Size: *20*
    - Dropout: *0.5 for all hidden layers*
    - Dense Units: *128 , 1025 ,512 respectively for all FC layers*
    - Activation : *relu, for hidden FC layers and softmax for last FC* Layer
-

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
global_average_pooling2d_2 ( (None, 512)               0
_________________________________________________________________
dense_5 (Dense)              (None, 128)               65664
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_6 (Dense)              (None, 1025)              132225
_________________________________________________________________
dropout_4 (Dropout)          (None, 1025)              0
_________________________________________________________________
dense_7 (Dense)              (None, 512)               525312
_________________________________________________________________
dense_8 (Dense)              (None, 16)                8208
=================================================================
Total params: 731,409
Trainable params: 731,409
Non-trainable params: 0
```

Overall, the model has been learning as indicated in the below plot. **The accuracy on testing data is 67.42%.**



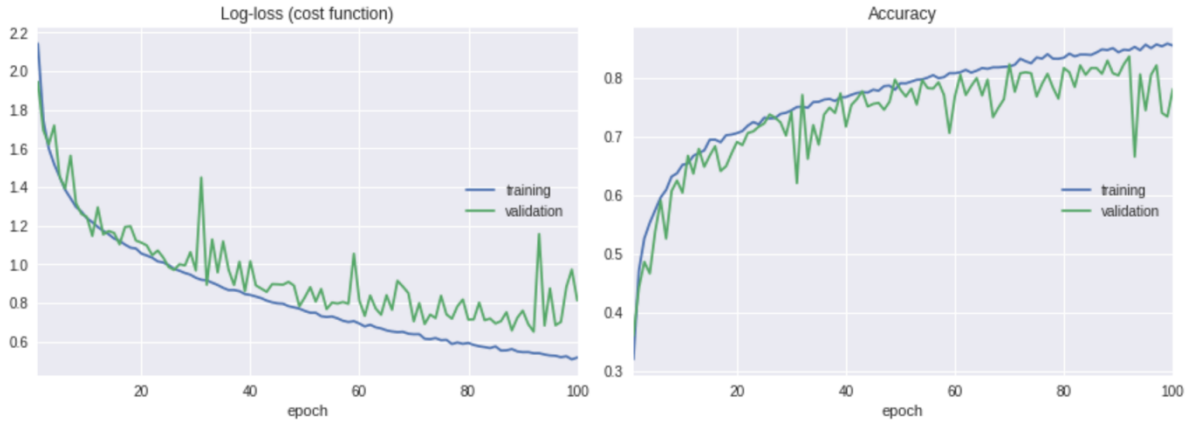
CNN Model Using Cropped Images

The second approach is to crop logos from the images and train the models using those cropped images. The idea is to show how much improvement of logo features will have model if possibly irrelevant pixels are eliminated.

As result of this, additional preprocessing step for the data is required to crop all images in the dataset.

Therefore, using the same pure CNN architecture design with just replacing dataset with cropped images will have the following impact on the model.

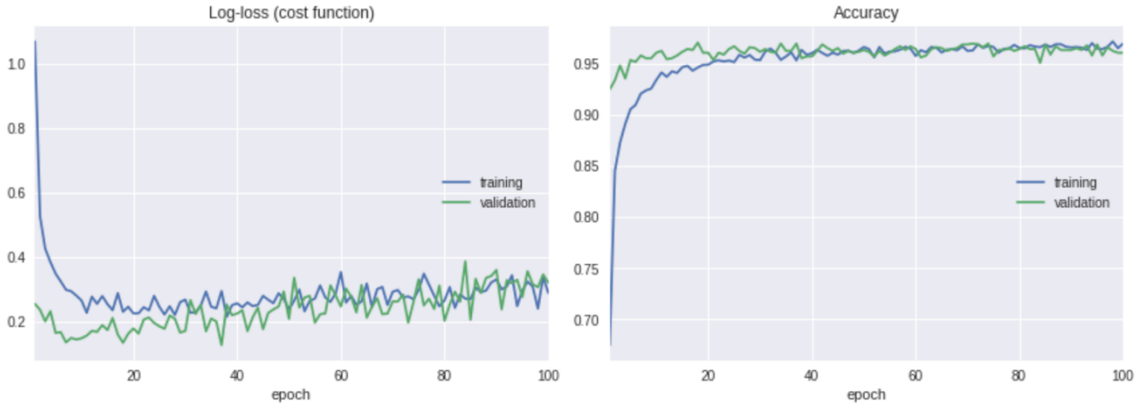
Overall, the model has been learning as indicated in the below plot. **The accuracy on testing data is 78.81%.**



CNN Model Using Cropped Images and Resnet 50

Using the same Resnet50 implementation used in images without cropping, the model will have the following impact on the data.

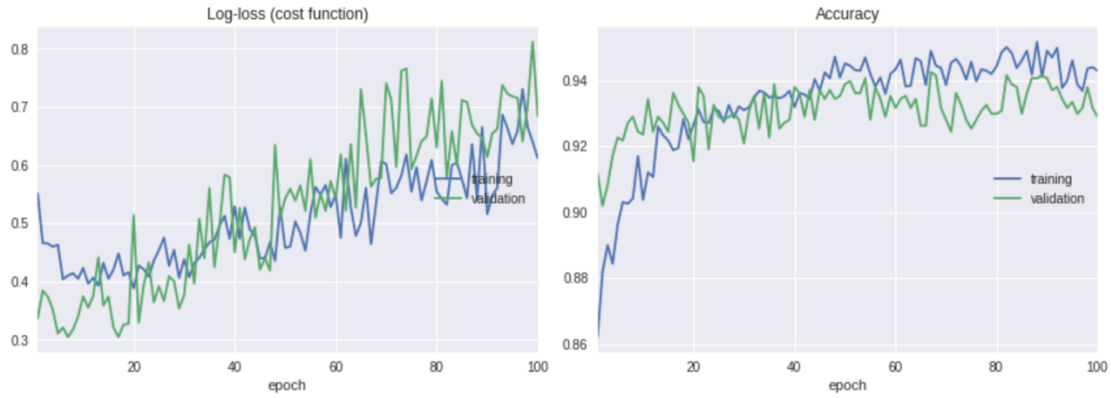
Overall, the model has been learning as indicated in the below plot. **The accuracy on testing data is 94.55%.**



CNN Model Using Cropped Images and VGG19

Using the same VGG19 implementation used in images without cropping, the model will have the following impact on the data.

Overall, the model has been learning as indicated in the below plot. **The accuracy on testing data is 90.79%.**



Result

Model Evaluation and Validation

In typical object detection studies, it is always best to detect objects and then classify them. That is why, there are a number of algorithms which provide a combination of detecting and classifying objects. Faster RCNN is considered to be one of the best algorithms for such scenario. Our experiment using pure CNN with transfer learning supports the above statement where cropped images outperform non-cropped images significantly.

There are two methods implemented including cropped images and non-cropped images. Each method was implemented in different CNN architectures including Pure CNN, CNN with Resnet50 and CNN with VGG19.

CNN model with transfer learning is proved to provide better performance. The architecture with transfer learning provides two main characteristics. The first one is that training time for the model is significantly faster than the standard CNN. The second one is the performance of the model is way better than standard CNN.

The **Open Logo Detection Challenge** is providing test sets to validate the model created along with the annotation of each logo in the image.

Performance of Different Models		
	Cropped Images	Non-cropped Images
Pure CNN	**80.49%**	**50.20%**
CNN with Resnet 50	**95.84%**	**65.24%**
CNN with VGG19	**90.79%**	**59.10%**

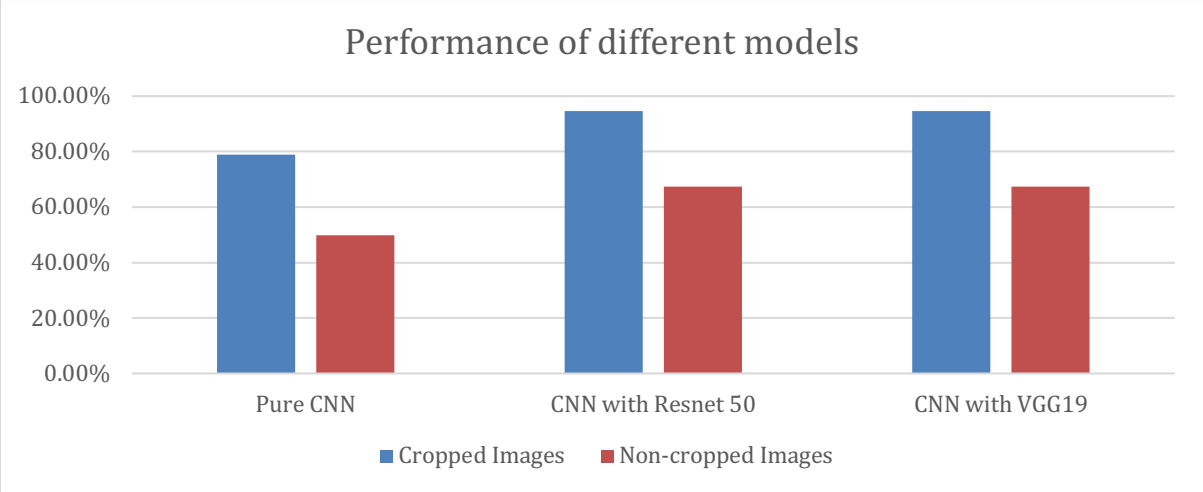
On the other hand, a k-fold cross validation has been performed over different validation fold. From the below table, it is very clear that the model is robust and stable across the entire dataset.

5-Fold Cross Validation					
Method	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**
Cropped Images	92.28%	91.88%	91.78%	90.79%	90.10%
Non-Cropped Images	55.45%	50.20%	57.92%	58.51%	59.21%

Therefore, **the model with transfer learning such as CNN with Resnet50 or VGG19 is selected one**. It is very clear that from the different models and various methods implemented that transfer learning has a significant improvement on the model. The boost in the performance is always when learning the actual object and classifying an object with no noise pixels as in cropped-images.

Justification

Due to the change in the dataset, it will not make sense to compare against the benchmark used in the leaderboard in the **Open Logo Detection Challenge.** However, our naive benchmark was 49.80% based on a pure CNN over non-cropped images with the best possible parameters. The selected model is, on the other hand, CNN with Resnet50 or VGG19 which both outperforms the naive benchmark both on cropped images and non-cropped images.



While the current model is performing very well on the dataset, it still did not solve the challenge for Open Logo Detection is open due to the following reasons:

- The reality is that there are many logos in the world and a logo never be trained the model will not recognize it.
- Images are always taken in the wild. Therefore, it will not be always cropped.
- Our solution was implemented in pure CNN while the best practice shows that Faster RCNN and YOLO have outstanding results than just CNN.

Conclusion

Free-Form Visualization

In object detections in general and in logo specifically, images are always taken in the wild. Therefore, any problem of this kind has some characteristics be taken care of.

1. Image Noise

From the above image, the logo part constitutes a small part of the total pixels in the image. This has a significant impact on the model when trying learns patterns.

2. Multiple objects

Image taken in the wild may possibly have multiple logos in the image. Our model was assuming that each image represents one logo and not considering other objects in the image.

Reflection

Throughout the course, we have been learning different aspects in machine learning. The common aspects across all these projects which I completed in the course. This project makes me think analytically and critically from the beginning starting the initial thought of the project idea.

The process I followed to reach to complete the projects as follows:

1. I thought about image classification. During the search, I found a great competition in logo detection and decided to pick it up after reading a couple of related paper in this competition.
2. Downloaded the datasets and did all necessary pre-processing steps for training the model including challenges resulted in diverting from the original competition.
3. A benchmark was created to help compare our model progress.
4. A couple of models created with different characteristics, tuned and trained on the data until found the best parameters each.
5. Tested all these against test data to verify the accuracy of each model.

Out of all these steps, I found pre-processing steps is a very tough and sometimes boring step. The issue with the preprocessing is that there are not pre-defined steps to pre-process the data. Each project and more specifically each implementation may require a set of pre-processing steps to progress into the next phase. Another challenge is computing resources. I found it very difficult to estimate the computing resources required for some activities such as load images and re-scaling them. This causes me a problem when using Colab and pushes me to divert from the original competition. Overall, the project is very interesting and touches a really nice area in image classification. I will proceed to understand the computing resources tricks and how I could manipulate the data with the limited resources available.

Improvement

The implementation used in project is based purely on CNN and CNN with transfer learning. It is believed that well-known Faster RCNN and YOLO may have considerable improvement on the dataset. Indeed, all papers in the coemption is using these algorithms. This indicates that such algorithms will have better performance. In fact, I tried to implement but seemed that it requires a deeper understanding of Computer Vision which I will be considering the following goal in my learning path.

References

- Foo C., Hui Y., Joon H. & Jeevan K., 2018, *"Vehicle logo recognition using whitening transformation and deep learning"*, Springer

- Hang S., Xiatian Z. & Shaogang G., 2018, *"Open Logo Detection Challenge"*, In Proc. British Machine Vision Conference (BMVC), Newcastle, UK