National College of Ireland

Comparing Convolution Neural Network For Single Object Vs Multiple Object Classification

MSc Research Project
Cloud Computing

Rahul Raghava
x17108799

School of Computing
National College of Ireland

Supervisor: Manuel Tova-Izquierdo

National College of Ireland
Project Submission Sheet – 2017/2018
School of Computing

Student Name:	Rahul Raghava
Student ID:	x17108799
Programme:	Cloud Computing
Year:	2018
Module:	MSc Research Project
Lecturer:	Manuel Tova-Izquierdo
Submission Due Date:	20/12/2018
Project Title:	Comparing Convolution Neural Network For Single Object Vs Multiple Object Classification
Word Count:	5215

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

Signature:	
Date:	27th January 2019

PLEASE READ THE FOLLOWING INSTRUCTIONS:

1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.
3. Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Comparing Convolution Neural Network For Single Object Vs Multiple Object Classification

Rahul Raghava
x17108799
MSc Research Project in Cloud Computing

27th January 2019

Abstract

Convolution Neural Network of huge network size can classify several objects in the image ranging up to 1000 objects. The accuracy of the network might be high, but the accuracy of each individual object varies. The accuracy of a few objects would be less than the average accuracy of the network. But few applications used for image classification would just require to classify single objects in an image. If the application requires a single object to classify, the network could be trained for it. The research paper compares the accuracy of the convolution neural network for a single object and multi object, which could provide better understanding of the network to select a single object or multiple objects for training the network.

1 Introduction

Convolution Neural Network is a go-to solution for object detection in an image due to the accuracy these networks can achieve. The CNN can achieve object detection without much effects from brightness, position, colour or orientation of the object in the image. The run time of the CNN is fixed, making it easier to predict the completion of the task. Also the number of layers in CNN has increased rapidly to classify a huge number of an object. With all the advantages CNN was able to grow faster, which makes it the best solution for object recognition in an image.

Increasing the network size has also resulted in better accuracy and lower error rate. This indicates that increasing the network size would provide better performance. In contrast, a sudden increase in the network size may also lead to very minimal improvement or it may even reduce the accuracy. Using a larger network in CNN requires higher computation power, making it less efficient to use. Though several other techniques such as inception module by Szegedy et al. (2015) or residual block from He et al. (2016) could be used, but this requires high computation power. Implementing a solution for better performance and efficiency is a challenging task.

Most of the applications that use CNN would require to classify few objects from an image, the applications such as classifying agriculture yield to a different class or tag a motion in surveillance etc. The main task of the research is to provide a solution to improve the accuracy for applications with few objects to classify. Since most of the network is designed to classify huge numbers of objects, these networks would not be

efficient for classifying few objects. The smaller networks would be much suitable for classifying a few objects since they have similar or less accuracy and minimal execution time. Since smaller networks require less computation power, they could also be executed multiple times, making it easier for localization network such as Ren et al. (2015) and Redmon et al. (2015) to execute.

When network is trained to classify certain number of objects, accuracy for a object varies by itself. By using single object to classify, CNN could increase the accuracy for the certain object. This could be due to more number of layer weights or filters which could be just used for classifying the single object. So, this papers tries to compare the performance of the CNN for single object classification and multi classification.

2 Convolution Neural Network

Convolution Neural Network (CNN) is inspired by the Multi-Layer Perceptron described in Lippmann (1987) which used a similar algorithm for feedforward and back propagating through the network. Initially, this network was used for single array inputs and later the implementation was used for handwritten character recognition, which had image or two-dimensional array of values. The implemented network could perform better than other algorithms, but training the network was a huge task, because loss would vary for each input resuting in rapid changes of the weights. To avoid it Lecun et al. (1998) proposed to change of the weights for a batch of input images combined resulted faster learning time.

Due to lack of parallel computation or even huge amount of data for training, CNN was not practical to be used widely for a while. This changed when programmable GPU was introduced, which could perform high amount of parallel computation. Alexnet by Krizhevsky et al. (2012) proposed a CNN for classifying huge number of objects in the image. This network used Relu for activation layer and Dropout layers for better results. Several other CNN architectures such as VGGnet by Simonyan and Zisserman (2014), ZF net by Zeiler and Fergus (2014), Googlenet by Szegedy et al. (2015) and Resnet by He et al. (2016) have transformed the capability of computer imaging to another level. Using CNN, it is possible to identify, localize or describe an object in the image.

Advantages of Convolution Neural Network: Three major advantages of the CNN which are sparse interaction, parameter sharing and equivariant, explained by Goodfellow et al. (2016). With higher number of hidden layers CNN can reduce the number interconnection between the layers, which was called as sparse network. This is due to convolution and connection between nodes are achieved in later stages as the network spreads. Sparse interaction reduces the number of weight parameter and computation. Parameter sharing is done when CNN uses same parameter for different node in the layer. Equivariant would result in similar output after interchanging the filters between layers.

2.1 Working of CNN

Convolution Neural Network has several hidden layers which are computed in parallel with respect to each node in the layer and continues a similar pattern for the entire network. The colour images are generally taken for input as three channels of input,

which consists of RGB data for each channel. Mostly, the input of the CNN is a layer of convolution nodes. This layer performs convolution for the input image with a filter of size less than image. The filter consists 2D array of weights different for each connection between the nodes. The convolution operation equation from Goodfellow et al. (2016) for an image I of size i × j and a filter K of size m × n with Bias B is shown in 1. All of the pixel from the output of a node, form a 2D array of values which is referred to as feature map.

$$C(i,j) = B_{ij} + \sum_m \sum_n I(i+m, j+n)K(m,n) \tag{1}$$

After performing convolution, activation function is used on each value from a feature map. This layer is generally considered within the convolution layer. The activation functions used are nonlinear that makes output nonlinear. Out of several activation function Rectified Liner unit (Relu), is widely used for CNN. Relu function is same as input when input is above 0, else the output is 0. Equation of relu is $f(x) = max(0, x)$. Some other activation functions are a sigmoid function, soft-max function, and tanh function. These layers are generally used in the output layer, which concludes the output.

After an activation layer, few of the features map data could be redundant. To make computation efficient and minimize the data, max pooling layer is used on the feature map. The window of specific size is used, where the maximum value within the window is taken as the output. Window size could vary from 2 × 2 to 5 × 5 or higher. For a feature map of size i × j and a window size of x, the output would be of size i/x × j/x. The size of output feature map would be reduced by a huge amount making further less computation. With three layers in order of convolution layer, activation layer and max pool layer, hidden layers of the CNN is designed as shown in 1. The number of layers in the hidden layer varies based on application, computation power etc. The output shown

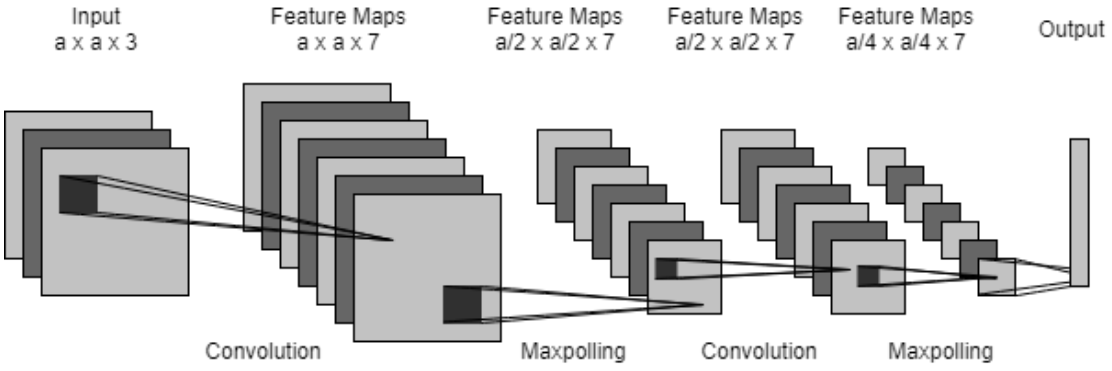


Figure 1: General Architecture of Convolution Layer

in 1 could have different implementation based on the network. One implementation converts all the feature map data into single array of data. Using the single array data as input for a fully connected network, output is taken for required number of objects. Implementation of the fully connected network which is also called as Artificial neural network(ANN) is shown in 2. Another implementation is Global Average Pooling, which applies average pooling until all of spatial dimensions are equal to zero. This avoids over fitting in output layer and provides better conversion from feature map to objects.

Training the network: Convolution neural network requires to train the weights so that the network can predict when an image is given as an input. Training the

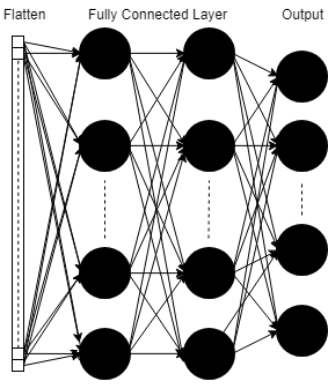


Figure 2: Popular implementation of output layer of Convolution Neural Network

network requires a set of images labeled to a class. When an image is provided as input to the network, error from expected value is calculated. Some of the error functions used are mean squared error, cross entropy error etc. Using error value, weights of the network are adjusted using back propagation. To optimize training and to reduce the training time, gradient decent is used. Used by Lecun et al. (1998), it accepts error value of a batch of images and alters the weights using back propagation. Several other techniques to optimize the network are Dropout used in Krizhevsky et al. (2012) and Batch normalization layers used in Ioffe and Szegedy (2015). Dropout layer is used to create a sparse network by randomly removing the connection between certain nodes while training. Batch normalization is used to normalize all the output from previous layer to avoid internal covariate shift.

2.2 Prominent implementation of CNN

Early implementation of the of CNN was by Lecun et al. (1998), which was called Lenet. This network was designed to classify handwritten numbers. Another breakthrough in CNN network was alexnet from Krizhevsky et al. (2012). This network was the first to classify several number of objects, which used dropout layer and Relu activation layer for the first time. Based on alexnet architecture, bigger network was created by Zeiler and Fergus (2014). Though this network had very less improvement, this paper visualized the filters used the network. This gave much deeper understanding to the CNN. Similar architecture CNN by Simonyan and Zisserman (2014), which used 3×3 filter size for entire network. This network achieved a much lesser error rate. On aim of building better CNN, deeper network was introduced. Inception Szegedy et al. (2015) was deep network with 22 layers. It introduced inception module, which was parallel micro network. The output from all parallel network were concatenated and provided as a single layer output. A deeper network was Resnet He et al. (2016). This network had 152 layer in the network, which mainly consisted of residual block. Residual block was a series of convolution, relu and convolution layer, with output values are added with input values. So the changes are added to the input. This had lowest error rate, when compared any other models.

CNN are also used in localization of the object in an image. First implementation of CNN for localization was achieved by R-CNN by Girshick et al. (2014). It used selective search to identify the possible region for an object. This region is fed to CNN to classify it to an object. But execution time of the R-CNN was very huge and increased as number of selective search increased. So Fast R-CNN was introduced by Girshick (2015), which

performed CNN on the entire image at first and later used selective search to localize the object. Though this was drastically faster, real time implementation was not possible. Faster R-CNN used Region Proposal Network at the output of the CNN for entire image to localize the object. Another localization technique mask R-CNN He et al. (2017) is able to localize an object to a pixel level. Real time localization CNN is YOLO by Redmon et al. (2015), which splits image to 16 $\times$ 16 grid uses two parallel steps. In one step CNN is used to classify class probability of the grid and another step creates a bounding box for possible objects. Combining the results, localized object is given as output. A improved YOLO 2 Redmon and Farhadi (2016) and YOLO 3 Redmon and Farhadi (2018) were implemented with fewer improvement to the previous implementation.

3 Related Work

Aim of the paper is to compare single object CNN and multi object CNN, to find out which network can provide better results. This comparison is done to get better accuracy from a CNN for application which use few objects for classification. Some of popular object recognition used before CNN where, viola jones algorithm by Viola and Jones (2001) and histogram algorithm by Dalal and Triggs (2005) . These algorithms where used widely for objects recognition such as face recognition, human etc. Though the flexibility of the objects where limited, they could classify much faster. As processing power was improved, the use of CNN grew making it popular image classifier. Major CNN implementation are described in section 2.2. Smaller CNN are trying to achieve higher accuracy with low processing requirement. Low computation network have achieved by reducing size of the network or reducing the weights of the network. Previous implementation of this networks explained in 3.1 and 3.2

3.1 Minimized Network

Based on the popular CNN Alexnet by Krizhevsky et al. (2012), a smaller CNN was designed. This network was called as squeezenet Iandola et al. (2016). It used a fire module, which could achieve better accuracy for a smaller network. Fire module consists of squeeze layer, with 1 $\times$ 1 filter size convolution layer. This is feed to expand layer which has convolution layer of 1 $\times$ 1 and a 3 $\times$ 3 parallel to each other. Expand layer has 4 times the number of nodes compared to squeeze layer for each of its convolution layer. Both layers output are concatenated as a single layer output. Instead of using fully connected layer for output, squeezenet uses last layer of convolution layer for output. Number of nodes in this layer is same as number of classification. Using average pooling for output of the nodes, it converts the feature map to 1 $\times$ 1 output. Classification is done by last activation layer.

Another implementation of smaller CNN is mobilenet Howard et al. (2017). It uses a depth wise layer instead of a convolution layer. This layer has depth wise convolution which uses single filter for each input, followed by 1 $\times$ 1 convolution layer. This reduces the computation of the network, because of single filter for each input. Two other variables width multiplier and resolution multiplier ranging from 0 to 1 are used. Width multiplier reduces the number of inputs towards a convolution layer and resolution multiplier reduces the output resolution layer from the a convolution layer. This values reduces the computation of the network by width multiplier $\times$ resolution multiplier times.

Densenet proposed by Huang et al. (2017), used a dense block to improve the connectivity between the layers. Dense block had several layers of convolution layer, where each output is provided to upcoming layer within the dense block to concatenate with the input of the dense block. Dense blocks where combined with pooling layer, batch normalization and dropout layers. Shuffle net Zhang et al. (2017) consists of group convolution layer and channel shuffle, which shuffles the inputs to outputs between different grouped channels. Shuffle unit similar to residual block which uses grouped convolution layer followed by channel shuffle. Tiny YOLO is a smaller implementation of the YOLO network Redmon et al. (2015). It was just a smaller network compared to YOLO. This was implemented on Darknet libraries Redmon (2013–2016). It used a CNN network to classify the object and selective search to localize the object in image.

3.2 Minimized Weights

Based on binary weights from Hubara et al. (2016), binarized neural network by Courbariaux et al. (2016) was created. 32 bit float values were replaced by binary weights, which represents ± 1. Basic mathematical operations such as addition was converted to count and multiplication was converted to XNOR operation. This reduced computation drastically, but the network was limited to number of objects it could classify. For batch normalization and weight updating, bit shift was used. Gradient of the network was achieved by straight through estimator. This network performed with great accuracy for a smaller size dataset such as CIFAR. To get better results for larger dataset , two bit implementation was proposed by Hubara et al. (2017). Using similar approach, Rastegari et al. (2016) proposed Xnor-net using ± 1 for filters in convolution layer. It used a block of xnor to perform convolution operation. Inputs are first normalized on th first step. followed by a activation layer. After activation, binary convolution is computed. Output of the network is pooled values from final layers of the network. Zhu et al. (2016) proposed to use a third value 0 to the binary weights. At each layer W was used which served as scaling factor. This made output from the layer to result within ± W and 0. While reducing the accuracy loss, this network performed well compared to all previous implementation. It could execute for larger dataset with good accuracy. Some other implementation was by Andri et al. (2016), which used binary weights to reduce the execution time of the network. Another one by Leng et al. (2017) reduced the last bits by squeezeing the values using alternateing direction method of multipliers(ADMM).

To reduce the execution time of the network, Jacob et al. (2017) proposed to convert weights from float to integer. All of the parameters and inputs where stored in integer. After each layer in the network, the output from previous layer is converted to integer values. Making it easier to execute in the next layers. This implementation made network much efficient.

4 Methodology

The main aim is to compare convolution neural network with single class classification and multi class classification. This is done to check if the single class classification can provide better accuracy compared to multi object classification. CNN uses filters which extract features from image to classify an object. The reason that a single object classification could perform with better accuracy is that it can use more number of filters available to classify just single object and also accuracy of the multi object classification is average of

each object accuracy. This means that, a object can have better accuracy than several other objects in the network and multi object classification has to share filters with several objects, the accuracy of an object could also be affected by other objects.

Comparing the CNN for single object and multi object classification would require training and testing images and CNN networks. Selecting right CNN network and image data set is crucial for this comparison. Network should be suitable for both multi object and single object classification, so it should not result in an advantage for either of the classification. Image data set should have large number images for training and set of images for testing. This images should have higher height and width than any of the input height and width of the network used. Steps required to perform comparison are given in figure 3

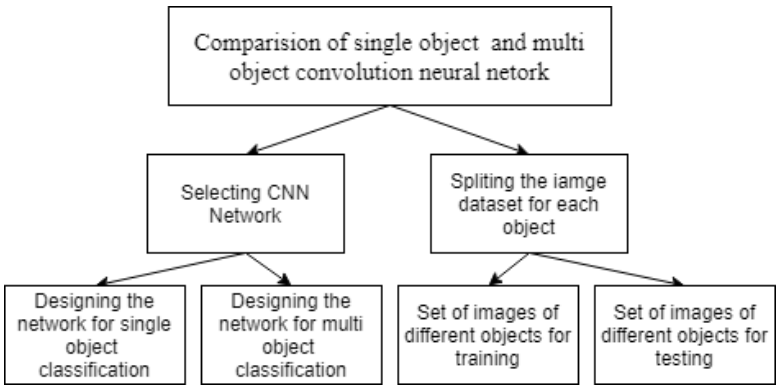


Figure 3: Steps to perform for comparing single object and multi object CNN

Parameters on which CNN are compared are based on the accuracy and error. These values show how well the network is capable of predicting correct output. Accuracy is the percentage of correct prediction to the total number images used. Error is average of the loss value form each image. While accuracy gives the percentage of network correctness, error shows how close the predicted outputs are to expected. Lesser error value means that network is quite confident with the prediction. For a better network, accuracy should be high and error should be low.

4.1 CNN for comparison

Using the previous implemented CNN network for comparison of both network would be much better, since designing the network is quite difficult. Even after designing the network it needs to be tuned for better performance, which is a time consuming one. Implemented CNN's are tested for various data set, making them perfect for comparison. From several CNN implementation, one which are trained for few number of objects should be used, as they are designed for few objects. Smaller network are much efficient for few number of objects. In smaller network, minimized network are much suitable for comparison. This is because minimized weights result fluctuate highly making them not great for implementation. From minimized networks, using a single network for comparison would not be suitable. Since a single network could favor single object or multi object. Using multiple networks for comparison could reduce this issue. Using two or three network to compare single object and multi object comparison could give a better understanding

4.1.1 Single object CNN

Implementing single object classification is not possible in CNN. This is because CNN requires atleast two possible results while training the network so that it can learn the difference between objects. When a single object is used to train the network, it is always expected to produce same output. Even when a different image is given to classify, network gives the same result. So to perform single object classification, another set of images which does not include the objects should be used while training.These images could be anything so that it provides network to differentiate from the object. So single object CNN would have two outputs, one representing the presence of object and other representing absence of object in image. Basic architecture of single object CNN is shown in figure 4a. This network needs to be trained for each individual objects with separate network that would also uses CNN with images that does not contain the object. After training the network, images from the testing set would be should used to check the accuracy and error by the network. This values are used to compare with multi object CNN with each object.

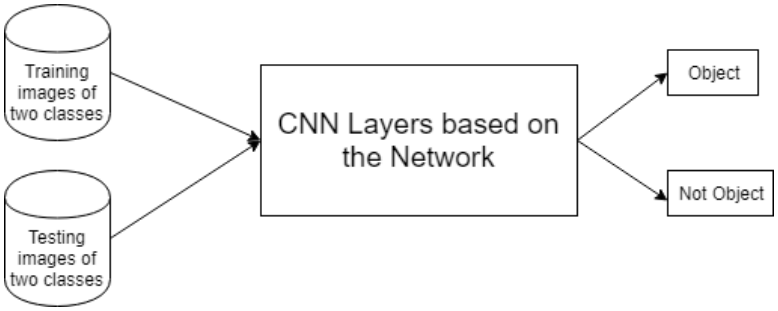
4.1.2 Multi object CNN

Multi object CNN should have similar set of images which represents none of the objects. This set of images represents as none of the above objects, which was used in single object CNN. This is used since most of the applications could have a random image without any object. And network should have a extra output which represents these images so that it does not decide between the objects. Apart from this, it gives similar implementation to the single object CNN, making the network perfect for the comparison. For number of objects used in multi object CNN, it could vary widely based on the approach. Because of the smaller network, smaller number of objects ranging till ten could be used. Since training and testing takes time to perform and for the convince six objects are used. Six plus another class is used for the multi object CNN. After training the network, a set of images for single object and images representing none of the objects is used to testing. This testing is done for each object to compare this results with each object from single object CNN.

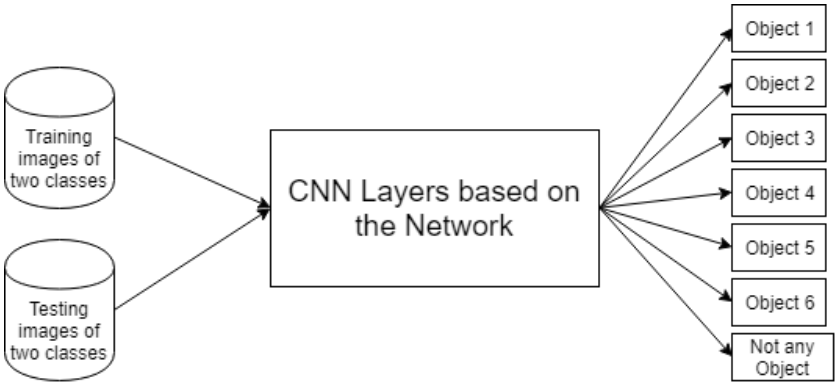
4.2 Image Dataset

Two sets of images are required, which are training and testing images. This images should be separated and not used while each other steps. If training images are used with testing or vise versa, accuracy and error results would invalid. Apart from two sets, a set images which does not contain any objects are required. These images could be background images, without the objects. By using background images, it limits the network to certain application such as table tennis ball classification by Zhao et al. (2017). Using the images with other objects which are not used could be done. This makes network to eliminate the features that are found in other objects.

Selecting the objects to be classified, is a tricky task. This is done based on the accuracy achieved in Girshick et al. (2014) and Redmon et al. (2015) and also objects importance to some of the application. The objects which are selected are airplane, car, dog, person , horse and train. These are the main objects used to classify. For images, any of the several datasets could be used. These datasets provide images for training and testing and labeled to the object. Some of the image dataset are Imagenet Russakovsky



(a) Single object CNN.



(b) Multi object CNN

Figure 4: Architecture used for the Single and Multi object CNN

et al. (2015), CIFAR Krizhevsky et al. (2014), Pascal VOC Everingham et al. (n.d.), COCO Lin et al. (2014).

5 Implementation

Comparing the single object CNN and multi object CNN requires a set of training and testing images which are classified to each object. For the image dataset, COCO dataset Lin et al. (2014) was used as it provided large set of images to represent a object. It has updated images with required number of classes and libraries to access and classify images. Selecting the images from COCO dataset is explained in section 5.1. For building convolution neural network, Keras on python 3 was used. Keras provides a higher level api for deep learning libraries such as TensorFlow, Microsoft CNKT and Theano. Details about the CNN implementation is explained in 5.2 CNN requires huge computation power to train the network, so cloud compute was used as it is a time consuming process. A Ubuntu instance on openstack is used for training the network.

5.1 Image Dataset

Images from COCO dataset was downloaded using python 3 library pycocotools. Details of images labeled to the object is taken from the library to download the image .Downloaded images are named after the objects which it represents and followed by id represented in the COCO dataset. This names are later used to classify images to its respective objects for training the network. Number of images labeled for a object can be high, so the number images per object are limited to 10000. Separate training and testing images are

provided and stored in different folder. Testing images are used to evaluate the network after training.

Different objects used to classify are airplane, bus, car, dog, person, train and images without previous mentioned objects which will be called as not class. Images for all the objects are download as COCO dataset classifies to the object except for not class. Multiple random classes other then mentioned objects are selected form the dataset. This provides a list of the images with classes requested, from which a random image is selected. This step is repeated for each image from random multiple classes. Images of diverse object are downloaded for the not class. As mentioned before, the number of objects are limited to 10000.

Testing images are downloaded in the similar manner as done for training images. But COCO dataset provides fewer images for testing, compared to training. Most of objects have testing images around 300. So for not class, number of images are limited to 300. Sample images from COCO dataset for each object is shown in fig 5

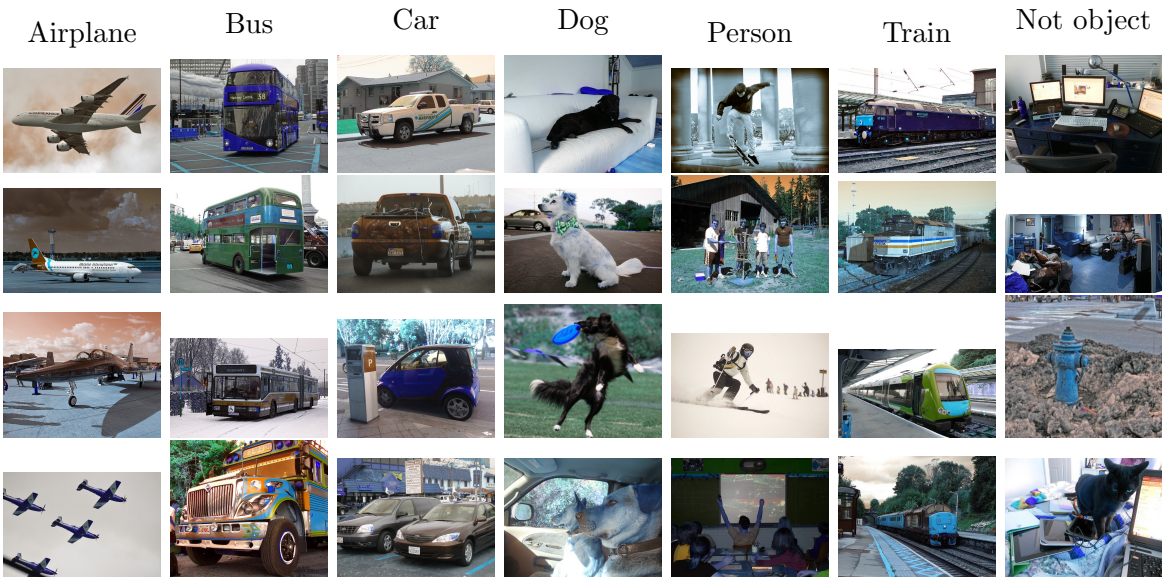


Figure 5: Sample images representing each object

5.2 Training CNN

Building the CNN network from scratch is challenging and time consuming task, because the network needs to be tested and optimized. Testing and optimizing a network requires several repeated computation of training the network. For optimized network and ease, an implemented network is a great choice. Some of the popular networks are Lenet5 Lecun et al. (1998), squeezenet Iandola et al. (2016), densenet Huang et al. (2017) etc. Lenet5 network is well optimized network and was designed for to classify 0 to 9 numbers from handwritten number. This network is smaller in size, making training process with less time. Also GPU is required for training on networks such as squeezenet and densenet. For second network, Lenet5 in input size of 64×64 is used. Structure for Lenet5 of size 32×32 and 64×64 is shown in Table 1 and 2. Input layer converts image of any size to specified size of 32×32 or 64×64. The layers are arranged sequential as given in table 1 and 2. Parameters are total number of weights and bias used in the layer. With

different input sizes, number of parameters are similar in both network for convolution layer. This is due to similar filter size of 5 × 5 and number of nodes in the layer.

Layer (type)	Output Shape	Parameters
input_img (InputLayer)	(32, 32, 3)	0
conv_1 (Conv2D)	(32, 32, 20)	1520
relu_1 (Activation)	(32, 32, 20)	0
maxpool_1 (MaxPooling2D)	(16, 16, 20)	0
conv_2 (Conv2D)	(16, 16, 50)	25050
relu_2 (Activation)	(16, 16, 50)	0
maxpool_2 (MaxPooling2D)	(8, 8, 50)	0
flatten (Flatten)	(3200)	0
full_connected_1 (Dense)	(500)	1600500
relu_3 (Activation)	(500)	0
full_connected_2 (Dense)	(2)	1002
softmax (Activation)	(2)	0
Total params: 1,628,072		

Table 1: Structure of Lenet5 for input size 32 × 32

Layer (type)	Output Shape	Parameters
input_img (InputLayer)	(64, 64, 3)	0
conv_1 (Conv2D)	(64, 64, 20)	1520
relu_1 (Activation)	(64, 64, 20)	0
maxpool_1 (MaxPooling2D)	(32, 32, 20)	0
conv_2 (Conv2D)	(32, 32, 50)	25050
relu_2 (Activation)	(32, 32, 50)	0
maxpool_2 (MaxPooling2D)	(16, 16, 50)	0
flatten (Flatten)	(12800)	0
full_connected_1 (Dense)	(500)	6400500
relu_3 (Activation)	(500)	0
full_connected_2 (Dense)	(2)	1002
softmax (Activation)	(2)	0
Total params: 6,428,072		

Table 2: Structure of Lenet5 for input size 64 × 64

Random image augmentation such as zoom, shear and horizontal flip are used to provide a different images while training. This converts single version of an image to multiple versions of the same image. Image augments can change between each training cycle called epochs. 25 epochs are computed for each model with evaluation step executed for the model using test images after each epoch. CNN require gradient descent to optimize the network, so adam optimizer Kingma and Ba (2014) is used. Adam optimization is a an efficient optimizer compared to other optimizer. A lower batch size of 32 is used, as the number of classes are few. Loss function binary cross entropy is used for calculating the loss. A keras callback function was used to log accuracy and error while training the network

A total of 7 models are used for each network type, in training. Of the 7,six models are single object CNN with output for 2 object. Each model are named after the object it classifies.This includes models for objects airplane, bus, car, dog, person and train. And the last model named as all objects, uses all of the objects images and not class. List of models and images used for training are provided in table 3..

	Model	Image of objects used for training
Single Object CNN	Airplane	Airplane, Not
	Bus	Bus, Not
	Car	Car, Not
	Dog	Dog, Not
	Person	Person, Not
	Train	Train,Not
Multi Object CNN	All Objects	Airplane, Bus, Car, Dog, Person, Train, Not

Table 3: Models used for each network type and images used for training the model

5.3 Testing the CNN

Training provides a final network for each model. This model is used to evaluate the accuracy and error using the testing images. For single object CNN the testing images include object images and not class images. For a multi object CNN, all testing images are used in evaluation. Multi object CNN test results include every object. For a similar comparison, multi object CNN are tested with image set of an object and not class. This step is repeated for each object used in training with not class. This provides a similar comparison for an object in single object CNN and multi object CNN.

6 Evaluation

Evaluation of a network is performed by testing the network for accuracy and error from test images. Images of the object and not class are used while testing each single object CNN. Similar approach is used for testing multi object CNN, where each object and not class is tested on the same multi object CNN.This provides a similar comparison on single object CNN and Multi object CNN. Accuracy of network is equal to percentage of correct prediction by the network. Error is average loss when the network predicts wrong class. Lower error shows a much stable network as the prediction are off by a small value.

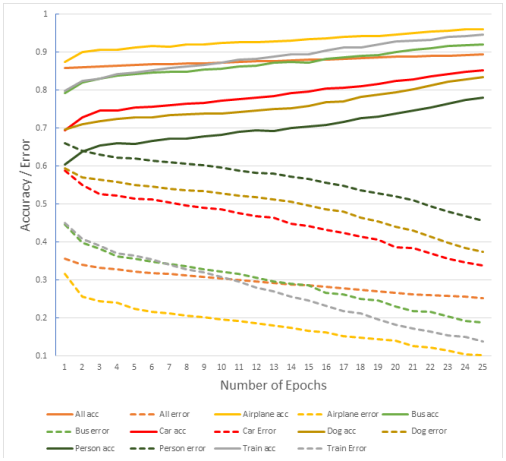
The evaluation results for lenet and lenet 64 is given in table 4. Single stage results are taken from last stage training evaluation as it is performed on the trained network. For multi object CNN, network is tested with images separately after training. Accuracy is described in percentage. Average accuracy and error is computed for each object class.

		Airplane		Bus		Car		Dog		Person		Train		Average	
		Acc%	Error	Acc%	Error	Acc%	Error	Acc%	Error	Acc%	Error	Acc%	Error	Acc%	Error
Lenet5	Single Object	93.9	0.214	84.4	0.401	83.5	0.410	68.7	0.725	66.6	0.718	87.9	0.354	80.83	0.402
	Multi Object	90.2	0.266	87.0	0.335	87.9	0.287	87.4	0.358	90.4	0.247	90.1	0.288	88.9	0.296
Lenet5 64	Single Object	91.6	0.266	85.2	0.334	82.9	0.438	72.5	0.631	73.6	0.658	89.2	0.352	82.5	0.446
	Multi Object	92.0	0.238	88.8	0.294	88.9	0.260	87.7	0.342	92.6	0.195	91.1	0.271	90.1	0.266

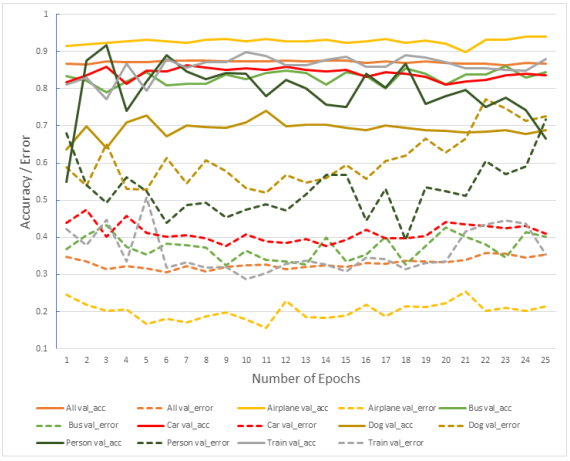
Table 4: Accuracy and Error for single and multi object CNN with different network

Accuracy and error while training a network computed after each epoch. This contains the accuracy and error for training images and test images. This values are saved using keras callback function. Values Lenet are plotted in figure 6. Training data of all 7 models

are plotted in left and testing data of all 7 models are plotted in right.Similarly Lenet 64 values are plotted in figure 7 with training on left and testing on the right.

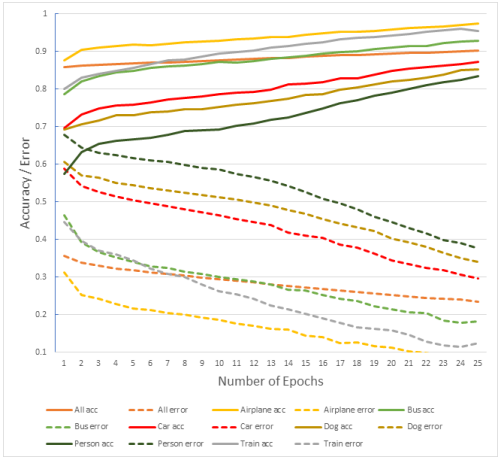


(a) Training accuracy and error

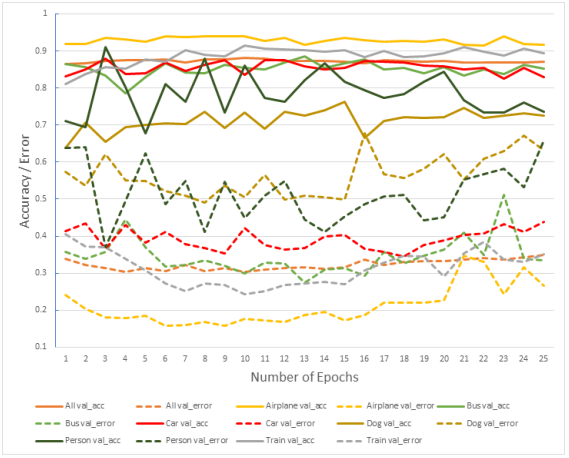


(b) Testing accuracy and error

Figure 6: Accuracy and error graph for Lenet



(a) Training accuracy and error



(b) Testing accuracy and error

Figure 7: Accuracy and error graph for Lenet 64

6.1 Discussion

Comparing single object CNN with multi object CNN shows that multi object CNN has provided better results. Apart form airplane object from lenet network, multi object CNN is capable of better accuracy and lower error. Average accuracy of single object CNN is quite closer to multi object CNN while average error nearly as twice as multi object CNN. Increasing the network input size has resulted by small improvement for most models and training evaluation results are smother compared to lenet.Increased input size has more impact on single object CNN. Results still are more dependent on the objects in single object CNN rather than multi object CNN. This was problem in Multi object CNN and it is even worse in single object CNN. Objects such as person and dog provided unstable results in single object CNN. Single object CNN are much more vulnerable to changes, making it unstable to use.

7 Conclusion and Future Work

Single object CNN where compared with multi object CNN for better accuracy. While results show that multi object CNN was able to provide much higher accuracy and was much efficient for object detection compared to single object CNN. So multi object CNN is still the better implementation for CNN as it requires single execution with better accuracy. Based on the number of required classification the size of the network can be designed.

Similar testing images used for evaluation of both CNN provided a deeper insight and displayed a stable classification. Increased input size of the network showed small improvement in accuracy and also improved accuracy widely for low accuracy models. Usage of Not class for single class CNN could have made it difficult for CNN to classify, as it did not consist of any important feature to differentiate. Comparing the multi object CNN with different number of object, could provide better insight. This could use similar comparison on test images and avoid not class for single class CNN. This could also compare the efficiency and accuracy of the network.

References

Andri, R., Cavigelli, L., Rossi, D. and Benini, L. (2016). Yodann: An ultra-low power convolutional neural network accelerator based on binary weights., *ISVLSI*, pp. 236–241.

Chollet, F. et al. (2015). Keras, `https://keras.io`.

Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R. and Bengio, Y. (2016). Binarized neural networks: Training neural networks with weights and activations constrained to+ 1 or-1, *arXiv preprint arXiv:1602.02830* .

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection, *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, Vol. 1, pp. 886–893 vol. 1.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J. and Zisserman, A. (n.d.). The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results, http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

Girshick, R. (2015). Fast r-cnn, *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448.

Girshick, R., Donahue, J., Darrell, T. and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation, *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587.

Goodfellow, I., Bengio, Y. and Courville, A. (2016). *Deep Learning*, MIT Press. `http://www.deeplearningbook.org`.

He, K., Gkioxari, G., Dollár, P. and Girshick, R. B. (2017). Mask R-CNN, *CoRR* **abs/1703.06870**.
URL: *http://arxiv.org/abs/1703.06870*

He, K., Zhang, X., Ren, S. and Sun, J. (2016). Deep residual learning for image recognition, *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications, *arXiv preprint arXiv:1704.04861* .

Huang, G., Liu, Z., Van Der Maaten, L. and Weinberger, K. Q. (2017). Densely connected convolutional networks., *CVPR*, Vol. 1, p. 3.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R. and Bengio, Y. (2016). Binarized neural networks, *Advances in neural information processing systems*, pp. 4107–4115.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R. and Bengio, Y. (2017). Quantized neural networks: Training neural networks with low precision weights and activations, *The Journal of Machine Learning Research* **18**(1): 6869–6898.

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J. and Keutzer, K. (2016). Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size, *arXiv preprint arXiv:1602.07360* .

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift, *CoRR* **abs/1502.03167**.
URL: *http://arxiv.org/abs/1502.03167*

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H. and Kalenichenko, D. (2017). Quantization and training of neural networks for efficient integer-arithmetic-only inference, *arXiv preprint arXiv:1712.05877* .

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization, *CoRR* **abs/1412.6980**.
URL: *http://arxiv.org/abs/1412.6980*

Krizhevsky, A., Nair, V. and Hinton, G. (2014). The cifar-10 dataset, *online: http://www. cs. toronto. edu/kriz/cifar. html* .

Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks, *in* F. Pereira, C. J. C. Burges, L. Bottou and K. Q. Weinberger (eds), *Advances in Neural Information Processing Systems 25*, Curran Associates, Inc., pp. 1097–1105.
URL: *http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf*

Lecun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998). Gradient-based learning applied to document recognition, *Proceedings of the IEEE* **86**(11): 2278–2324.

Leng, C., Li, H., Zhu, S. and Jin, R. (2017). Extremely low bit neural network: Squeeze the last bit out with ADMM, *CoRR* **abs/1707.09870**.
URL: *http://arxiv.org/abs/1707.09870*

Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., Hays, J., Perona, P., Ramanan, D., Dollár, P. and Zitnick, C. L. (2014). Microsoft COCO: common objects in context, *CoRR* **abs/1405.0312**.
URL: *http://arxiv.org/abs/1405.0312*

Lippmann, R. (1987). An introduction to computing with neural nets, *IEEE Assp magazine* **4**(2): 4–22.

Rastegari, M., Ordonez, V., Redmon, J. and Farhadi, A. (2016). Xnor-net: Imagenet classification using binary convolutional neural networks, *European Conference on Computer Vision*, Springer, pp. 525–542.

Redmon, J. (2013–2016). Darknet: Open source neural networks in c, `http://pjreddie.com/darknet/`.

Redmon, J., Divvala, S. K., Girshick, R. B. and Farhadi, A. (2015). You only look once: Unified, real-time object detection, *CoRR* **abs/1506.02640**.
URL: *http://arxiv.org/abs/1506.02640*

Redmon, J. and Farhadi, A. (2016). YOLO9000: better, faster, stronger, *CoRR* **abs/1612.08242**.
URL: *http://arxiv.org/abs/1612.08242*

Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement, *arXiv* .

Ren, S., He, K., Girshick, R. and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks, *in* C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama and R. Garnett (eds), *Advances in Neural Information Processing Systems 28*, Curran Associates, Inc., pp. 91–99.
URL: *http://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks.pdf*

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C. and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge, *International Journal of Computer Vision (IJCV)* **115**(3): 211–252.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition, *CoRR* **abs/1409.1556**.
URL: *http://arxiv.org/abs/1409.1556*

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A. (2015). Going deeper with convolutions, *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features, *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, Vol. 1, IEEE, pp. I–I.

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks, *European conference on computer vision*, Springer, pp. 818–833.

Zhang, X., Zhou, X., Lin, M. and Sun, J. (2017). Shufflenet: An extremely efficient convolutional neural network for mobile devices, *CoRR* **abs/1707.01083**.
URL: *http://arxiv.org/abs/1707.01083*

Zhao, Y., Wu, J., Zhu, Y., Yu, H. and Xiong, R. (2017). A learning framework towards real-time detection and localization of a ball for robotic table tennis system, *Real-time Computing and Robotics (RCAR), 2017 IEEE International Conference on*, IEEE, pp. 97–102.

Zhu, C., Han, S., Mao, H. and Dally, W. J. (2016). Trained ternary quantization, *arXiv preprint arXiv:1612.01064* .