Object Classification

Raj Thakkar [1], Priyanshi Gupta [2], Vathepalli Vamsi Bushan [3]

Abstract
This project has the goal of performing object classification based on different objects present in the images. The objects we have selected for this project are the ones easily found in a university classroom such as Laptop, TV, Cellphone, Notebook etc.For humans, it is somewhat easy to identify objects by looking at them, because we have been seeing them through out our lives and we are able to recognize them.
Through this project, we are trying to implement a machine learning model that is capable of identifying objects just as a human would (through images). Different architectures of CNN have been implemented to compare their performance on the object dataset created by us specifically for this problem.

Keywords
Image classification — Convolutional Neural Network — Machine Learning — VGGNet — RESNET — InceptionV3 — Xception

[1] *MS in Data Science, Luddy School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN, USA*
[2] *MS in Data Science, Luddy School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN, USA*
[3] *MS in Data Science, Luddy School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN, USA*
*__Corresponding author__: rajthakk@iu.edu, prigupt@iu.edu, vbushan@iu.edu

Contents

Introduction

Our primary goal is to classify objects in a given image by implementing different Convolutional Neural Network (CNN) architectures such as SmallerVGGNet, Resnet,Inception V3 and Xception. We will compare the results obtained (accuracy) by using different CNN architectures mentioned above and present a comparative analysis of the performance of the architectures listed above.

1. Background

Since AlexNet in 2012, the focus on Deep Learning for Computer Vision, in general, has increased rapidly. Every year new algorithms or models keep on outperforming the previous ones in the annual ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Over the years, since 2012, the algorithms that won this challenge include- ZFNet (2013), GoogLeNet (2014), VGGNet (2014), ResNet (2015), DenseNet (2016), SENet (2017). At the core of each of these algorithms is a Convolutional Neural Network (CNN). In addition, these architectures, include several optimizations to solve the challenges associated with object recognition, including-viewpoint variation, scale variation, intra-class variation, image deformation, image occlusion, illumination conditions, and background clutter. Drawing inspiration from the recent advances in Object Recognition, we wanted to check the performance of ResNet, VGGNet, Inception V3, and Xception on a personally curated dataset. We have focused only on Object Recognition, and we have handled the challenge of viewpoint variation through a state-of-the-art technique called Data Augmentation.

2. Dataset

2.1 Web scraping

We implemented a web scraping script to get images for each of the 9 classes mentioned in the classes sub section. The script enabled us to get 400 images for every object. We then

proceeded to clean the data that we got from the web because it may contains images that will deteriorate the behaviour of our model. Therefore, in the end we were able to add from 250 to 300 images for each class.

2.2 Classes

In all we have 9 classes of objects easily found in a university classroom which are:

1. Classroom Chair
2. Whiteboard Marker
3. Laptop
4. Cell Phone
5. TV
6. Classroom Table
7. Water Bottle
8. Backpacks
9. Notebooks

The dataset is presented in 9 different folders, one for each object where the folder name corresponds to the object's class name.

2.3 Data Preprocessing

The data preprocessing consisted of the steps listed below:

1. Overlapping and irrelevant images have been removed manually as we wanted to detect only single object in each image. Multiple object detection was not the focus of this project
2. Each image was loaded and converted into TensorFlow compatible array for further computation.
3. Since we had limited dataset, we have implemented data augmentation during training using Keras so that we can train on more data and reduce the chances of overfitting
4. Class label corresponding to images were converted into one hot encoding format by using inbuilt methods for further computation.

3. Convolutional Neural Network Architectures

Four different Convolutional Neural Network Architectures were explored for this project - SmallVGGNet, RESNET, Xception and InceptionV3

3.1 SMALLER VGGNET

Small VGGNet is a CNN architecture that is a smaller, more compact variant of the VGGNet network , introduced by Simonyan and Zisserman in their 2014 paper VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION

Its architecture is presented in the following figure:

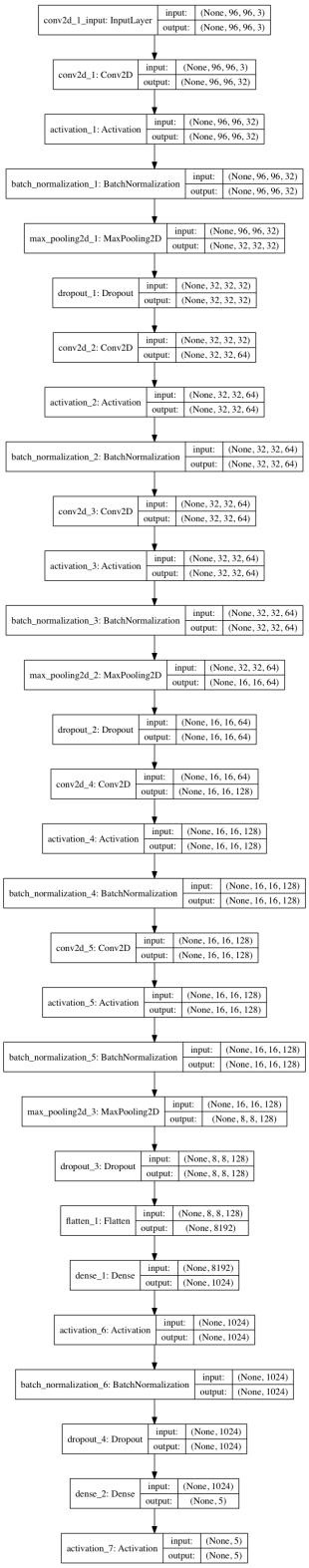

Figure 1. Smaller VGGNet Architecture

We referred the tutorial Keras and Convolutional Neural Networks (CNNs) for the implementation of this model.

The small VGGNet architecture we are using shares simi-

lar characteristics with all VGGNet like: VGGNet-like architectures are characterized by:

1. It only uses 3*3 convolutional layers stacked on top of each other in increasing depth
2. It reduces volume size by max pooling
3. It has fully-connected layers at the end of the network prior to a Softmax classifier

3.2 ResNet

ResNet was claimed to be one of the most groundbreaking work in terms of computer vision and deep learning. It can produce compelling results even after training hundreds and thousands of layers. It has enhanced the performance of many computer vision applications other than image classification. It becomes difficult to train a deep neural network as if may suffer from a problem called vanishing gradient. Therefore, ResNet introduced the core idea of "skip connections" to deal with this problem.

The skip connection in ResNet is explained by the figure below:

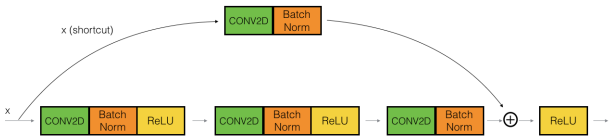

Figure 2. Skip Connection in ResNET

With ResNet, the gradients can flow directly through the skip connections backwards from later layers to initial filters. The network is 152 layers deep and down sampling of the volume though the network is achieved by increasing the stride instead of a pooling operation.

The complete model architecture of ResNET is given in the figure below:

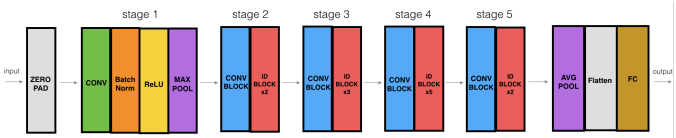

Figure 3. ResNet-50 Architecture

3.3 InceptionV3

Inception architecture adopts factorization of convolution in its design. Inception layer is a combination of 5x5 convolution layer,3x3 convolution layer and 1X1 convolution layer which is concatenated to an output. This output is then fed to the next stage as input. The whole idea of factorization is inspired by the fact that it reduces the number of connections and parameters without affecting the efficiency of the network. This helps in reducing overfitting. It gives good computational cost and has proven to be more efficient than VGGNet. Inception version which we have used in this project is 42 layers deep.

The architecture of InceptionV3 is given below:

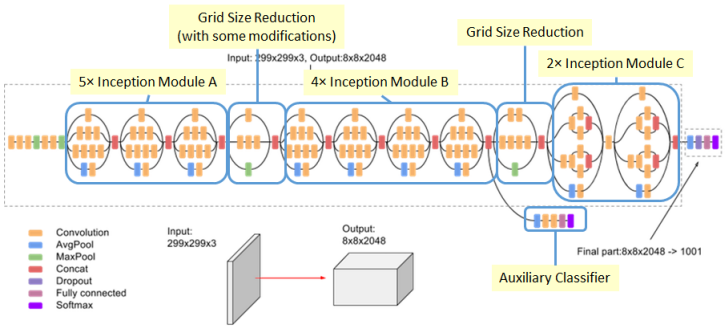

Figure 4. InceptionV3 Architecture

3.4 Xception

Xception stands for Extreme version of Inception which replaces inception module with depth-wise separable convolutions. It is a 36 layer deep network which has Linear stack of depth-wise separable convolution layers with skip connections. Usage of residual connections increases the accuracy thus proving xception network to be more efficient than VGGNet , ResNet , and Inception-v3.

The architecture of Xception is given below:

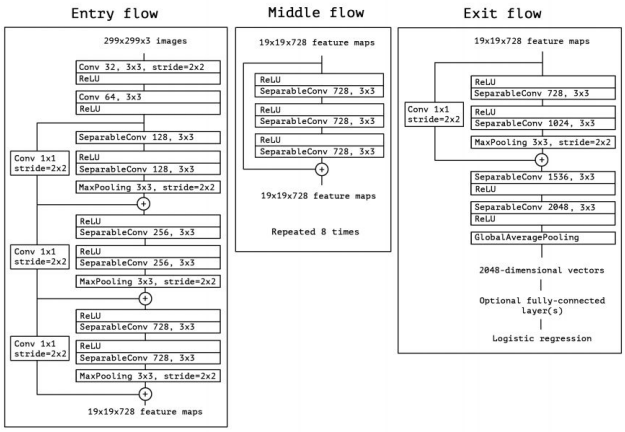

Figure 5. Xception Architecture

4. Methodology

For this project, we created our own custom dataset which would be used to test the performance of different CNN architectures implemented. The dataset was created by scraping the web pages using Bing Search API as discussed in the Data Section.

The models were implemented using Object Oriented Programming in a module which can be imported by any Python script that needs to use the models we have implemented. As the dataset size was relatively small for training a CNN, we used the 'ImageDataGenerator' from Keras to augment the dataset by applying random transformations such as rotations,

shearing etc. to generate additional training data. This is primarily done to avoid over-fitting.

The classes in the dataset were converted to one hot encoded vectors which can then be used in the script to train the model. It also helps in converting the predicted class by a model to human readable class label.

To validate the performance of the models while training, the training data is split into train(80%) and validation(20%) datasets. At each epoch, the performance of the model on validation dataset is tested. We also used checkpoints to store the model which has the best validation accuracy. To illustrate this further, let's say validation accuracy at epoch 61 is 58%. At epoch 62, the validation accuracy decreases to 57.6% then the model at epoch 62 will not be saved. Now if the validation accuracy is ¿58% at epoch 70 then only the model at epoch 70 will be saved. This helps us in having the best possible model (model that performs well on validation dataset) irrespective of number of epochs used to train the model

The scripts have been designed such that the user can decide which models needs to be trained, the file in which the best model weights should be saved, the dataset which will be used to train the model and the file used to store the binarised labels of the given dataset. This makes the script more generalised and it can be used to train CNN models on any dataset of user's choice.

The saved model is then used to classify the images we have taken using our personal cell phones to evaluate the performance of the models trained.

5. Results

Due to computational limitations of the personal computer we have, we were only able to train 2 models: SmallerVGGNet and RESNET on the dataset we have. The accuracy of the models on validation dataset are given below:

Table 1. Architecture and Validation and accuracy results

Architecture	Epochs	Batch Size	Accuracy
SmallerVGGNet	200	32	84%
ResNet-50	200	32	78%

Since, the validation accuracy of SmallerVGGNet is better than ResNet-50, we have used SmallerVGGNet to check its performance on the pictures taken for various classes taken using our personal cell phones.

6. Discussion

Contrary to our expectations, the SmallerVGGNet has better validation or generalization accuracy than ResNet50, on our dataset. The training accuracy of our ResNet50 model was 96%, while the validation accuracy was 78%. We suspect that due to the overwhelming number of trainable parameters (23 million approx.) in the ResNet50 model, our network is overfitting the test data. To improve the generalization accuracy we could either use drop-out or free some of the fully connected layers while training.

Further, while testing our best model (Smaller VGGNet) on real-world images taken from smartphones, we found that the model is sensitive to noise. This was expected as we trained our models on relatively clean images- with no occlusions, and only one object per image. So, to increase the robustness of our model, we could use Convolutional Autoencoders.

Through this project we also got a chance to work with

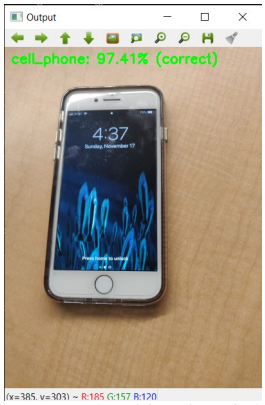

Figure 6. Correct Classification for Cell Phone

Figure 7. Correct Classification for Water Bottle

Figure 8. Correct Classification for Whiteboard Marker

Figure 9. Incorrect Classification for Backpack

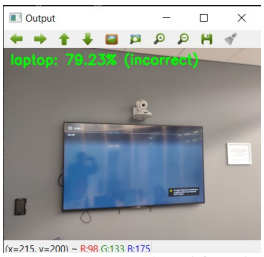

Figure 10. Incorrect Classification for TV

Figure 11. Incorrect Classification for Classroom Table

heavy models such as Inception V3 and Xception. We got to know more about how skip connections work with convolution factorizations and depth-wise separable convolutions. These model needs high computational power as they work with high input dimension which was one of the reasons that we were not able to train our model on these architectures.

7. Conclusion

The models implemented worked quite well even on the images taken using the personal cell phones. If more computational power was available, we could have implemented other models too and Xception would have perhaps outperformed the other models given the batch size and number of epochs remained constant.

Moreover, the project helped us in getting a grip on implementing various CNN architectures using Tensorflow and Keras. It built on top of the materials covered in the class and bolstered our understanding of various CNN architectures. The only drawback we have seen is that CNNs need a lot more memory and Computational capabilities at least as compared to Deep Neural networks. Even though the number of parameters in CNNs are huge they are not very slow to train because of the recent advances in GPUs and their computational capabilities.

Contribution

Everyone on the team participated and worked towards achieving the goals set up from the beginning. A rough division of the work among team members is given below:

Table 2. Contribution of Team Members

Task	Owners
Web Scraping for Dataset	Raj
Dataset Preprocessing	Priyanshi
Smaller VGGNet Implementation	Raj
ResNet	Vamsi
InceptionV3	Priyanshi
Xception	Priyanshi
Model Evaluation	All
Project Presentation	All
Project Report	All

Acknowledgments

We would like to thank Professor Ariful Azad for his efforts and dedication during this semester in order to introduce us to the field of Linear Algebra, Machine Learning and Deep Learning and to push us to dig in it and discover different aspects of it.

Many thanks to the Assistant Instructors of Introduction to Intelligent Systems course : Kaikai Zhao and Zhengyi Li for their help and support all through the semester and for their guidance for the accomplishment of this project.

We would also like to thank all the community of developers and researchers who not only contribute greatly to the fields of Machine Learning and Deep Learning but also enable us to learn from their experiences and build on them.

References

[1] Understanding and visualising ResNets by Pablo Ruiz

[2] An overview of ResNet and its variant by Vincent Fund

[3] Understanding Inception Network from scratch by Faizan Shaikh

[4] Imagenet: VGGNet, ResNet, Inception and Xception with Keras by Adrian Rosebrock