CSE 676 – Deep Learning (Fall 2020)
Project One Description

Sargur N. Srihari

University at Buffalo, The State University of New York
Buffalo, NY 14620
Contact: 716-645-6162 (O), srihari@buffalo.edu

Release: 2020, September 23

Due: 2020, October 15, Midnight

1 Introduction to Deep Learning Neural Networks

Many deep learning models have been proposed and implemented towards the task of extracting features from a given image for classification. There is a lot of literature discussing new architectures from the point of view of the layers composition and recognition performance. Hence, it is very informative as an introductory project to analyze the aspects of a few architectures in terms of memory usage and inference time and how computational cost impacts the recognition accuracy.

2 Networks

2.1 VGGNet [1]

The VGG network architecture was introduced by Simonyan and Zisserman in their 2014 paper, Very Deep Convolutional Networks for Large Scale Image Recognition.

This network is characterized by its simplicity, using only 3×3 convolutional layers stacked on top of each other in increasing depth. Reducing volume size is handled by max pooling. Two fully-connected layers, each with 4,096 nodes are then followed by a softmax classifier.

2.2 ResNet [1]

Unlike traditional sequential network architectures such as AlexNet, OverFeat, and VGG, ResNet is instead a form of "exotic architecture" that relies on micro-architecture modules (also called "network-in-network architectures").

The term micro-architecture refers to the set of "building blocks" used to construct the network. A collection of micro-architecture building blocks (along with your standard CONV, POOL, etc. layers) leads to the macro-architecture (i.e,. the end network itself).

First introduced by He et al. in their 2015 paper, Deep Residual Learning for Image Recognition, the ResNet architecture has become a seminal work

2.3 InceptionNet [1]

The "Inception" micro-architecture was first introduced by Szegedy et al. in their 2014 paper, Going Deeper with Convolutions

The goal of the inception module is to act as a "multi-level feature extractor" by computing 1×1, 3×3, and 5×5 convolutions within the same module of the network — the output of these filters are then stacked along the channel dimension and before being fed into the next layer in the network.

The original incarnation of this architecture was called GoogLeNet, but subsequent manifestations have simply been called Inception vN where N refers to the version number put out by Google.

The Inception V3 architecture included in the Keras core comes from the later publication by Szegedy et al., Rethinking the Inception Architecture for Computer Vision (2015) which proposes updates to the inception module to further boost ImageNet classification accuracy.

The weights for Inception V3 are smaller than both VGG and ResNet, coming in at 96MB.

3 Project Requirements

3.1 Implemention and Coding

3.1.1 Task Definition

1. Implement VGGNet 16, ResNet 18, InceptionV2 architectures using **SGD**[2] and **ADAM**[3] optimization with the following variations in network regularization schemes:
   (a) No Regularization
   (b) Batch Normalization [4]
   (c) Dropouts [5]
2. Use CIFAR-100 dataset for training and testing.
3. Calculate Precision, Recall and Accuracy to evaluate each of the 18 experiments.
4. Implement **Early Stopping** regularization scheme in all the experiments.

3.1.2 Submission Definition

All the below items must be compressed in a zip file format

1. A folder named "notebooks" containing 18 Jupyter notebooks containing the complete code for each experiment.

   Additional requirements within the notebooks:

   (a) Each jupyter notebook must contain an ultimate cell which when executed generates the required metrics for the experiment performed in that notebook.

   (b) The name of the notebook must follow the pattern {Arch}_{Setting}_{Optimizer}.ipynb (example: InceptionV2_ADAM_NoRegularization.ipynb)
2. A folder named "checkpoints" containing 18 weight files with the same name as the jupyter notebook. The weight files should be the best weights only.
3. A folder named "Report" containing the report in PDF format.
   The requirements for the report are as follows:

   (a) The project report must be well-organized and preferably be written in LaTeX. Students may use the latest NIPS style files (available at https://nips.cc/Conferences/2019/PaperInformation/StyleFiles).

3.2 Report

The report must contain the following points:

1. Introduction section describing the project in general and why is it useful to do the comparison.

2. Describe implementation of the architectures, the training as well as the specifics of the training procedure used.
3. Explain the networks as best as possible in your words.
4. Plots must be included in the report which shows the performance of the given model in terms of Accuracy and Loss over the trained epochs.
5. Display a chart like the one shown in Figure 1 in the report to compare the above networks with various combinations and explain the results section.

	Arch	VGG 16			ResNet 18			Inception v2		
Optimizer	Score	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy
	Setting									
SGD	With BatchNorm	1	1	1	2	2	2	3	3	3
	With DropOut	4	4	4	5	5	5	6	6	6
	No Regularization	7	7	7	8	8	8	9	9	9
ADAM	With BatchNorm	10	10	10	11	11	11	12	12	12
	With DropOut	13	13	13	14	14	14	15	15	15
	No Regularization	16	16	16	17	17	17	18	18	18

Figure 1: Sample output table, where 1, 2, 3, ... represent the experiment number.

6. A conclusion section.

4 Plagiarism and cheating

1. Third party codes and snippets are allowed given proper citations are provided in both report and notebooks. A failure to provide references will result in a zero for the project.
2. Instructors will load the weight files back in the networks and cross-verify with other students. Ideally no two students should have same parameters.
3. Instructors will use high tech tools to cross-verify the code and report with other students for plagiarism.
4. Attempts to violate Academic Integrity mentioned in points 2 and 3 will result into "F" grade for the course.

5 Doubts and Queries

Kindly do not assume scenarios in case of any queries. Do reach out to the instructors during office hours or through emails to clarify your queries.

6 Grading Rubric

Weight	Requirement
30%	Code clarity and ease of execution
30%	Results (Plots, Evaluation Table)
30%	Conceptual Clarity in Report
10%	Report Formatting

References

[1] R. Adrian, "Imagenet: Vggnet, resnet, inception, and xception with keras," https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/, 2017.

[2] S. Ruder, "An overview of gradient descent optimization algorithms," 2016.

[3] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[4] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[5] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html