Image classification using CIFAR10 and analysis of hyperparameter

Abhinav Tiwari | Manasa Bhimaraya Singhekar | Nilesh Nerkar

ABSTRACT :

The research is on a object recognition task on CIFAR 10 dataset. The major winning Convolutional Neural Networks for this problem has been widely used leading to architecture such as AlexNet, VGGNet, ResNet, GoogleNet.The above architecture includes tens to hundreds of millions of parameters, which impose considerable computation and memory overhead. This limits their practical use for training, optimization and memory efficiency.
We aim to explore and build a simple CNN architecture and compare it the pretrained model such as VGG16 to see how well it performs on the CIFAR 10 dataset.
Initially, we iteratively tried to play with specific hyper parameters and then proceeded to play with the architecture of the model to finally land on the configuration that gave us the best possible outcome for the task at hand. We would summarize the effect of hyper parameters tuning on the model efficiency and accuracy.

INTRODUCTION :

The CIFAR-10 dataset (Canadian Institute For Advanced Research) is a collection of images [1] that are commonly used to train machine learning and computer vision algorithms. It is one of the most widely used datasets for machine learning research. The CIFAR-10 dataset contains 60,000 32x32 color images in 10 different classes. The 10 different classes represent airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. There are 6,000 images of each class.Computer algorithms for recognizing objects in photos often learn by example. CIFAR-10 is a set of images that can be used to teach a computer how to recognize objects. Since the images in CIFAR-10 are low-resolution (32x32), this dataset can allow researchers to quickly try different algorithms to see what works. Various kinds of convolutional neural networks tend to be the best at recognizing the images in CIFAR-10. CIFAR-10 is a labeled subset of the 80 million tiny images dataset. When the dataset was created, students were paid to label all of the images.

Background to this our research focus was on object detection. CIFAR10 contains vast resources for this task. One of the intriguing of the dataset is that although the state-of-the art architecture such as AlexNet, VGG, ResNet, GoogleNet provide higher accuracy , the architecture include tens to hundreds of parameters which impose considerable computation and memory overhead. This limits their practical use for training , optimization and memory efficiency. Another blog [2] implies that the human accuracy of classification ended up being 94%. These factors led us to consider and experiment with CNN(Convolution Neural Network) Architecture and hyper parameters tuning to see if it's really necessity have a complex and large set of parameters to achieve accuracy. We would be considering VGG16 pretrained model as our base to achieve efficient model with less parameters and computational overhead.

Below are the questionable images that makes the image classification of CIFAR10 questionable.

VGG16[3] (also called OxfordNet) is a convolutional neural network architecture named after the Visual Geometry Group from Oxford, who developed it. It was used to win the ILSVRC (ImageNet) competition in 2014.

This model has an accuracy of 92.6%

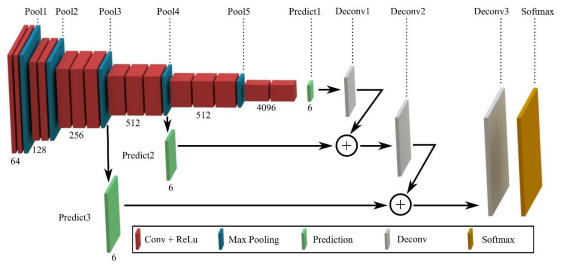
Model	Acc.
VGG16	92.64%
ResNet18	93.02%
ResNet50	93.62%
ResNet101	93.75%
MobileNetV2	94.43%
ResNeXt29(32x4d)	94.73%
ResNeXt29(2x64d)	94.82%
DenseNet121	95.04%
PreActResNet18	95.11%
DPN92	95.16%

METHODS AND CODE :

The approach was to start from a Vanilla 4 layer[4] model and then gradually increasing the depth and width of the network to understand how the each parameter will affect the efficiency and performance of the model.

1. Change the optimizer
2. Change the cost function
3. Change the Kernel Regularizer
4. Include/ exclude data augmentation
5. Activation functions
6. Number of epochs
7. Number of layers
8. Network Architecture
9. Network Initialization

Build a final model that encompasses best results from the conducted experiments that's comparable to VGG16.



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

The above is the architecture of the Vgg network

RESULTS:

4 layer Configuration

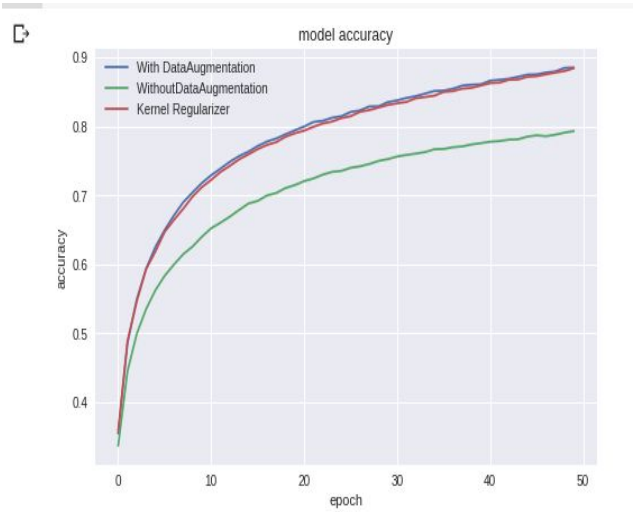
Network	Best Hyper-Parameter	Accuracy (Training)
4 layer	Optimizer SGD Adam RMSprop ADAGRAD ADADELTA	Best SGD 82%
4 layer	Activation function Relu TanH Elu Sigmoid	Best Relu 77.58%
4 layer	Epochs- 20,50, 80,100	100
4 layer	Cost function Categorical Hinge	70%

For the 4 layers experiments, we tried out the above hyperparameters and noted the accuracies. It is been concluded from our experiments that among SGD, Adam, PMSprop, AAGRAD and ADADELTA Optimizers SCD was the best with an accuracy of 82%. For Activation Functions using 4 layers Relu gave the highest accuracy of 77.58%. Further 100 epochs gave the highest accuracy and the Cost function Categorical Hinge have around 70% accuracy.

6 layer Configuration

Network	Parameters	Accuracy
6 layer	With Data Augmentation	82.2%
6 layer	Without Data Augmentation	81%
6 Layer	Kernel Regularizer L1	78.42%
6 Layer	Kernel Regularizer L2	82.2%

For the 6 layers experiments, the highest accuracy was noted for Data Augmentation using Kernel L2 Regularizer which is 82.2%.



The above graph represents the training accuracy for 6 layer model with different tuning parameters with least for when experimented without Data Augmentation

CONCLUSIONS:

The final model with data augmentation resulted in training accuracy of 87% with a average training time of **22** minutes on Google colab with distributed GPU.
The pre-trained model of VGG stands at 93.52% with training time of **30 minutes.**

SCOPE:
The following were the deliverable:
Try out different experiments using CNN Architecture for image classification into 10 classes of CIFAR 10 dataset
1) Research about the existing models with less computation power
2) CNN using 4 layers
3) CNN using 6 layers
4) VGG16 pretrained model[4]

The above experiments were conducted and pipelined as we were exploring each architecture.This meant having to explore each resource, the different configurations of CNN's and optimize the network in order to understand the difference in each parameter makes in terms of model efficiency.

We split the work between the three of us where we all conducted research about the topic and then each member explored sub-heading 2, 3 and 4 individually and conveyed their understanding.

The project really helped all three of us to understand most of the relevant concepts taught in class about Neural Networks and left us with a thirst for further research in the domain.

CONTEXT:
The final architecture built was combination of parameters and hyperparameters that gave the best results from the above conducted experiments.

The reference for the architecture and explanation of the each layer are in the references.

From the given architectures, we tweaked the model to in terms of depth and width. We introduced dropout, batch normalisation and data augmentation to have a model that's stable in terms of efficiency and performance. Our approach was to build and extend the work we did on the 4th assignment and then include external research in our notebooks to come up with the model that we finalized on.

REFERENCES:

1.https://www.cs.toronto.edu/~kriz/cifar.html

2.http://karpathy.github.io/2011/04/27/manually-classifying-cifar10/

3.https://arxiv.org/pdf/1409.1556.pdf

4.https://github.com/geifmany/cifar-vgg/blob/master/cifar10vgg.py

5.https://github.com/kuangliu/pytorch-cifar

6.https://www.youtube.com/watch?v=gmBfb6LNnZs&list=PLZbbT5o_s2xq7LwI2y8_QtvuXZedL6tQU&index=36

7.https://blog.plon.io/tutorials/cifar-10-classification-using-keras-tutorial/