Deeper Networks for Image Classification

Name: Moad Saad Khorchef
ID: 200934655
Queen Mary University
London, United Kingdom

I. Introduction

Convolutional Neural Networks (ConvNet) are achieving an excellent success in computer vision field Mainly in image retrieval and classification domain. as an example, it is applied to image processing tasks, human activity recognition [1]. One, encouraging news is that most of this progress isn't just the results of more powerful hardware, larger datasets, and greater models, but mainly a consequence of latest ideas, algorithms, and improved network architectures. The aim of this work is to analyze the Performance and evaluate image classification with deeper networks. The few models of CNN namely, VGG and Resnet are used for this task. Dataset used for training are Mnist and CIFAR. The training process is coded with PyTorch.

ResNet as Winner of ILSVRC 2015 in image classification, detection, and localization, as well as Winner of MS COCO 2015 detection, and segmentation. VGG was the 1st runner-up, not the winner of the ILSVRC (ImageNet Large Scale Visual Recognition Competition) 2014 in the classification task narrowly beaten by GoogLeNet.

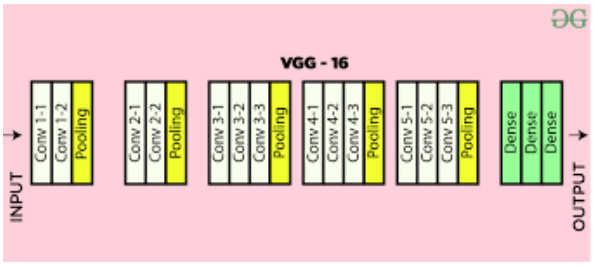
Deep Residual Network [2] has been the most revolutionary work in the computer vision / deep learning community over the past few years. Resnet allows you to train hundreds or even thousands of layers and still deliver excellent performance. Although VGG is based on AlexNet [3], there are some differences that make it different from other competing models. VGG uses a smaller window size and stride into the first convolutional layer. VGG also addresses another very important aspect of CNN, depth.

II. Critical Analysis and Related Work

A. VGG

Simonyan et al. [4] developed VGG in a paper that examines the depth of a Convolutional Neural Network with an in-depth study on ImageNet. By stacking a greater number of 3x3 convolutional layers on top of each other, the paper focuses on addressing all other design parameters and increasing the model's scope. This paper corresponds to current research at the time, which was convergent on the assumption that deeper networks contribute to better results, such as Goodfellow et al. [5], who used deep ConvNets for street number recognition, and GoogLeNet by Szegedy et al. [6], which won ILSVRC-2014 with a rather complex deep CNN.

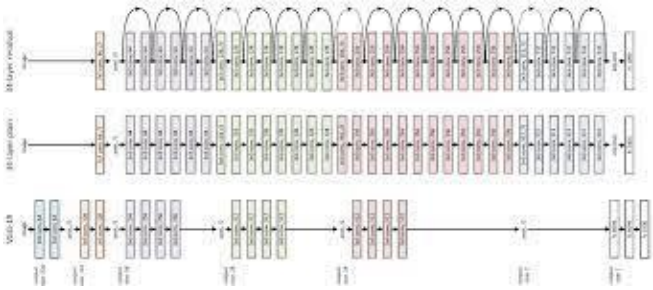
The classification error on ImageNet decreased as the ConvNet depth increased, from 11 layers to 19 layers in a network, according to the VGG paper. Another noteworthy finding is that the number of parameters was not significantly higher than in previously proposed shallow networks.



VGG16

The image above depicts VGG16, a well-known network derived from the paper. The current convolutional layers progressively decrease the size of the 224x224 image to a 7x7 image. The main benefit of using a stack of layers with smaller kernel sizes rather than a single layer with a large kernel size is that the former allows several ReLU to be incorporated, making the decision function more discriminative.

B. ResNet



ResNet was introduced by He et al. [2] as a method for effectively training deeper neural networks. The paper tackles the issue of accuracy loss in very deep networks. They do this by constructing a residual network consisting of the blocks seen on the right. We can now let these layers match a residual mapping instead of hoping that each few stacked layers directly fit a desired underlying mapping. A residual network makes sense because we expect the deeper network to have no more training error than the shallower counterpart if we simply add layers that map identity only to the shallower network. As a result, if the identity mappings

are ideal, mapping the residual to zero is simpler than fitting the underlying mapping identity.

The baseline plain network they create has less filters and lower complexity than VGG, according to the paper. In contrast to VGG19, the plain baseline has 19.6 billion FLOPs, while the residual network with 34 layers has 3.6 billion FLOPs.

The paper also adds Batch Normalization to network architecture, which means that forward propagated signals have non-zero variance, meaning that the gradient vanishing problem is addressed in both directions.

III. Model Description

A. Model Architecture

1) VGG

VGG accepts a 224x224 pixel RGB image as input. To keep the input image size constant for the ImageNet competition, the authors cropped out the middle 224x224 patch in each image.

VGG's convolutional layers have a very small receptive field (3x3, the smallest scale that captures both left and right and up and down). There are also 1x1 convolution filters that perform a linear transformation of the input before passing it through a ReLU unit. The convolution stride is set to 1 pixel in order to maintain spatial resolution after convolution.

VGG has three completely connected layers, the first two of which each have 4096 channels and the third of which has 1000 channels, one for each class.
VGG's hidden layers all use ReLU (a huge innovation from AlexNet that cut training time). Local Response Normalization (LRN) is not used by VGG because it increases memory consumption and training time without improving accuracy.

VGG uses very small receptive fields instead of large receptive fields like AlexNet (11x11 with a stride of 4). (3x3 with a stride of 1). The decision function is more discriminative now that there are three ReLU units instead of only one. There are also fewer parameters (27 times the number of channels vs. 49 times the number of channels in AlexNet).

Without modifying the receptive fields, VGG uses 1x1 convolutional layers to make the decision feature more non-linear.

VGG may have a large number of weight layers due to the limited size of the convolution filters; of course, more layers means better efficiency. However, this isn't an unusual element. In the 2014 ImageNet competition, GoogLeNet [6], another model that uses deep CNNs and small convolution filters, was also presented.

We will be using the VGG16 and VGG13 variants of VGG in this project.

2) ResNet

As can be seen, all ResNet models start with a typical 7x7 convolutional layer and a max pooling layer; these layers also have padding, which is not shown in the table. After that, there are four "layers," each with a different number of blocks.

There are two different blocks used, one called the Basic Block, and one called the Bottleneck block.
The initial 7x7 convolutional layer, batch normalisation, a ReLU activation feature, and a downsampling max pooling layer are all specified in our ResNet class. The four layers are then built using the config file, which defines the block to use, the number of blocks in the layer, and the number of channels in that layer. The number of channels in a layer for the BasicBlock is simply the number of filters for both convolutional layers within the block.
Also, the first layer has a one-stride stride, while the last three layers have a two-stride stride. This stride is only used in the "downsampling" residual route and to adjust the stride of the first convolutional layer within a block.
The BasicBlock comes first.

Two 3x3 convolutional layers make up the BasicBlock. Conv1 has a stride that differs depending on the layer (one in the first layer and two in the other layers), while conv2 always has a stride of one. Each of the layers has a padding of one, which means that before applying the filters to the input image, we add a single pixel around the entire image that is zero in every channel. A ReLU activation feature and batch normalisation follow each convolutional sheet.

When downsampling, we apply a convolutional layer to the residual path with a 1x1 filter and no padding. This is accompanied by batch normalisation and has a variable stride. A 1x1 filter with a stride of one does not affect the height or width of an image; instead, it has out channels number of filters, each with a depth of in channels, i.e. it is increasing the number of channels in an image through a linear projection rather than downsampling. With a stride of two, the image's height and width are reduced by two, since the 1x1 filter just passes over any other pixel - this time, the image is simply downsampled as well as the channels' linear projection.

Because each of the convolutional layers within a block uses the same number of filters, the BasicBlock has an expansion of one.

The ResNet paper also covers variants specifically for the CIFAR10/100 datasets. We will be using the ResNet20 and ResNet44 variants in this project.

There is no pooling layer after the first convolutional layer, which has a smaller filter scale, lower stride, and less padding. It also only has three layers instead of four, and its own block form. And one requirement is that each layer's number of channels must be exactly double that of the previous layer.

The only difference between the CIFAR variant BasicBlock and the regular BasicBlock is the Downsampling residual relation. In addition, according to the paper, the ResNet models for CIFAR use a downsampling relation with "zero padding" and "parameter-free shortcuts." This is accomplished by the use of an Identity module.

IV. Dataset Description And Preperation

A. Dataset Description

1) MNIST

There are 70,000 photos in total. Each MNIST picture is a digit between 0 and 9 written by government employees and census bureau employees. Each picture also has ten labels attached to it: zero, one, two, three, four, five, six, seven, eight, and nine. Each image is 28 pixels wide and 28 pixels tall, with a width of 28 pixels and a height of 28 pixels. The MNIST data is in reasonable quality. The digits have been preprocessed to ensure that they are perfectly aligned.

2) CIFAR

CIFAR is pronounced C-FAR (see-far). The number suffix in CIFAR10 and CIFAR100 refers to the number of classes/categories. Each CIFAR has a 32x32 pixel resolution. A total of 50,000 training images and 10,000 label images are available. Each CIFAR10 image corresponds to one of ten categories that have been labelled. Each of the CIFAR100 images corresponds to one of the 100 groups. ImageNet compares 1000 different categories. The 10 categories are: airplane, car, bird, cat, deer, dog, frog, horse, ship, and truck, in no specific order.

B. Dasaset Preproccessing

1) MNIST

I'd like to point out that VGG was not optimized for a dataset with such a limited input size as MNIST, which has images that are 28x28 pixels in size, so I'd like to talk about how I dealt with that first. I considered shrinking MNIST to the smallest size permitted by VGG and adjusting the network accordingly. MNIST was resized to 32x32, which means that the convolutional layers would minimise the image size to 1x1 with 512 filters over time. As a result, instead of the original 512x7x7, my input size for fully connected layers will be 512x1x1. We also normalised the image tensors through each channel by subtracting the mean and dividing by the standard deviation.

2) CIFAR

Instead of reserving a portion of the data from the training set for validation, we will simply use the test set as our validation set. This simply adds to the amount of data available for training. We will also normalise the image tensors through each channel by subtracting the mean and dividing by the standard deviation. In addition, when loading images from the training dataset, we can apply transformations at random. Take a random 32 x 32-pixel crop, and then flip the image horizontally with a 50% chance.

V. Experiments

Here we will show the experiment performed on CIFAR and MNIST using the models we have chosen to see which model performs better under predefined condition an hyperparameters.

While experimenting we found the having batch number set at 128 works better than 32, 64 some model don't even converge the batch number is low or achieve very low accuracies.

A. Expriments on MNIST

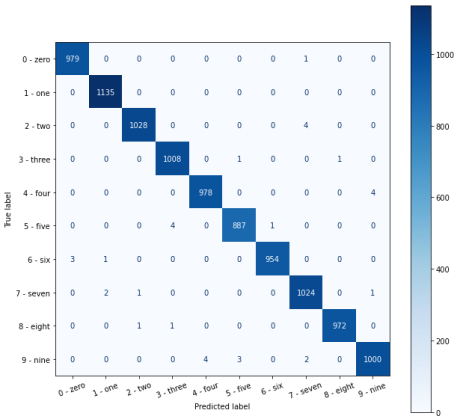
In training All the number of epochs were set to 50 to make it easy to compare models in the same level.

1) VGG

For VGG I used VGG13 and VGG16 in the experiment both performed extremely well in the MNIST classification. SGD optimizer seemed to work best than the Adam optimizer with latter does not seem to converge at all. We implemented a Learning Finder (LR) to get the best LR for the Model we are training. although, I tried different LRs [0.01, 0.001] and implemented LR schedular.

However, schedular had the best test and validation accuracy, even though the score of LR = 0.01 was nearly equally same.

And the biggest confusion was between 4 and 9 in both models, only confused between them 4 time with VGG13 model which outscored VGG16 by only 0.03%.



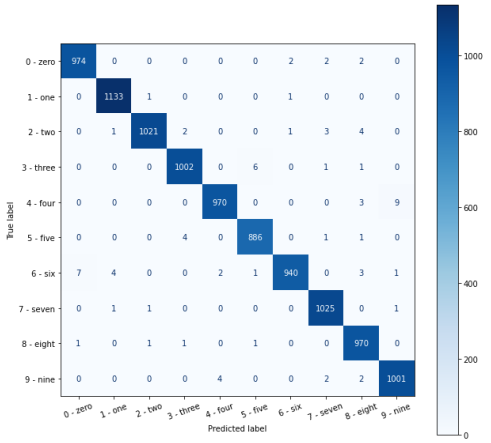
Comparing models	scores		
	LR = 0.01	***LR = 0.001***	***Schedul ar***
VGG13	99.64	99.57	**99.65**
VGG16	**99.62**	99.53	99.6

2) ResNet

For ResNet I used ResNet20 and ResNet44 in the experiment both performed extremely well in the MNIST classification. SGD optimizer also seemed to work best than the Adam optimizer with latter does not seem to converge at all same as VGG.
We also implemented a Learning Finder (LR) to get the best LR for the Model we are training. although, I experimented with different LRs [0.01, 0.001] and implemented LR schedular.

However, the schedular had the best test and validation accuracy again, even though the score of LR = 0.001 and LR = 0.01 were nearly equally same scores as well.

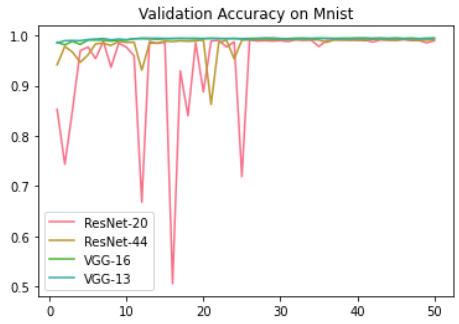
And the biggest confusion was again between 4 and 9 in both models, only confused between them 9 time, however ResNet44 seems to get confused between 7 and 2 and confused between 3 and 5 each 5 times.
The ResNet44 model has outscored ResNet22 by only 0.01%.



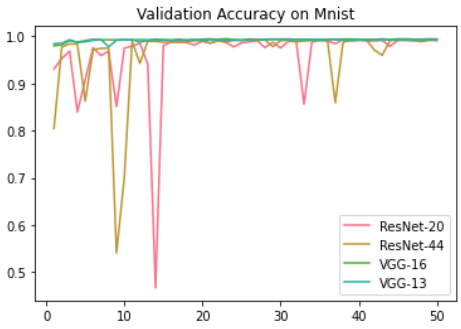
Comparing models	scores		
	LR = 0.01	***LR = 0.001***	***Schedular***
ResNet20	**99.37**	98.81	99.22
ResNet44	99.31	99.0	**99.38**

3) Comparing

VGG13 model outperformed all of the other model but by only a slight negligible margin. And all models done extremely well using LR = 0.01 or the schedular and the SGD optimizer however having a fixed LR gets better results. they failed poorly using the Adam optimizer.



Lr = 0.01



schedular

B. Expriments on CIFAR

In training All the number of epochs were set to 50 to make easy it to compare models in the same level.

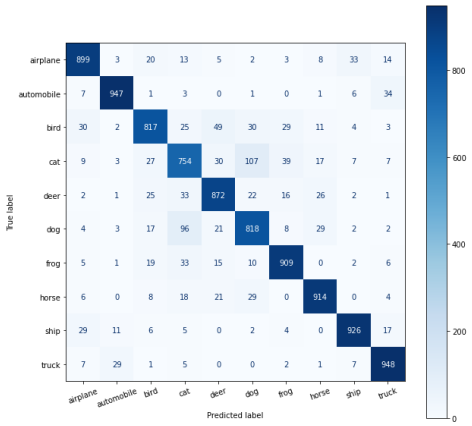
1) VGG

For VGG I used VGG13 and VGG16 in the experiment both performed fairly well in the CIFAR classification with highest accuracy reaching 89% would get any higher. SGD optimizer seemed to work best than the Adam optimizer with latter does not seem to converge at all. We implemented a Learning Finder (LR) to get the best LR for the Model we are training. although, I tried different LRs [0.01, 0.001] and implemented LR schedular.

However, using the LR = 0.001 here had the best test and validation accuracy again.

And the biggest confusion was between cat and dog in both models, with the best VGG model VGG16 was getting the

prediction wrong nearly 200 times combined which is high number. VGG16 model outscored VGG13 by 6.05%



Comparing models	scores		
	LR = 0.01	***LR = 0.001***	***Schedular***
VGG13	**87.77**	84.03	87.12
VGG16	88.04	**93.82**	87.82

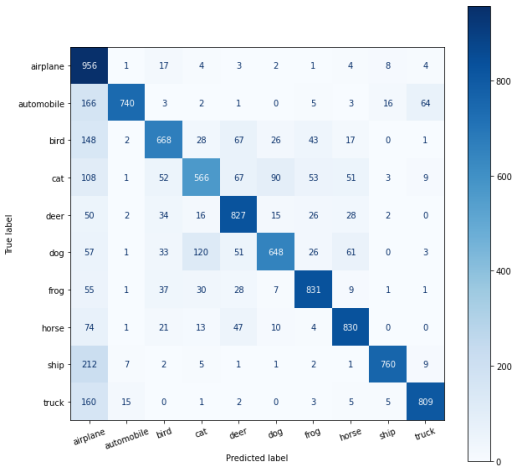
2) ResNet

For ResNet I used ResNet20 and ResNet44 in the experiment both performed extremely well in the MNIST classification. SGD optimizer also seemed to work best than the Adam optimizer with latter does not seem to converge at all same as VGG.

We also implemented a Learning Finder (LR) to get the best LR for the Model we are training. although, I experimented with different LRs [0.01, 0.001] and implemented LR schedular.

However, using the schedular here had the best test and validation accuracy again.

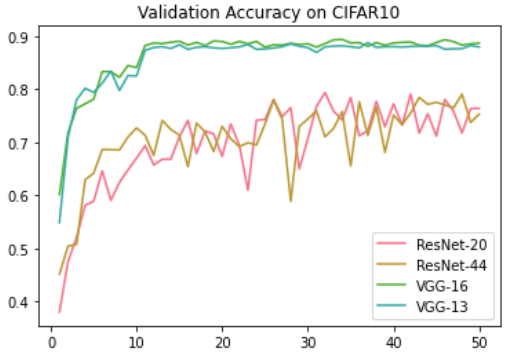
And the biggest confusion was between cat and dog in both models, with the best ResNet models ResNet20 was getting the prediction wrong over 200 times combined, which is high number it's also confusing between truck, airplane, and ship. The ResNet20 model has outscored ResNet44 by only 0.07%.



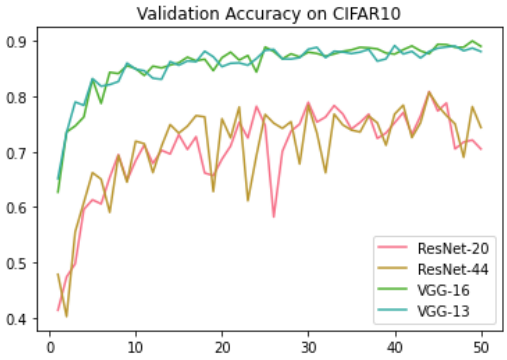
Comparing models	scores		
	LR = 0.01	***LR = 0.001***	***Schedular***
ResNet20	73.22	68.45	**76.35**
ResNet44	70.95	69.02	**75.65**

3) Comparing

VGG model were performing well in the CIFAR classification, VGG16 model outperformed VGG13 model but by only a slight negligible margin. And all models done extremely well using LR = 0.01 or the schedular and the SGD optimizer however using the schedular in this classification gets better results. they failed poorly using the Adam optimizer.



Schedular



LR=0.01

C. Experiment Evaluation

In these experiments that we have completed, VGG has outshone ResNet by getting better results especially in the harder task of CIFAR.

Comparing models	MNIST scores		
	LR = 0.01	***LR = 0.001***	***Schedular***
VGG13	99.64	99.57	**99.65**
VGG16	**99.62**	99.53	99.6
ResNet20	**99.37**	98.81	99.22
ResNet44	**99.31**	99.0	99.38

Comparing models	CIFAR scores		
	LR = 0.01	***LR = 0.001***	***Schedul ar***
VGG13	**87.77**	84.03	87.12
VGG16	88.04	**93.82**	87.82
ResNet20	73.22	**68.45**	76.35
ResNet44	70.95	**69.02**	75.65

VI. CONCLUSION

In this article, I compared and contrasted two deep convolutional networks, VGG and ResNet. As previously mentioned, VGG and ResNet are two significant milestones in the development of Deep Neural Networks. They're an excellent place to start learning about Deep CNN architectures.

The results show that representation depth is beneficial to classification accuracy, and that a deep convolutional neural network can achieve state-of-the-art efficiency on the dataset. VGG has a much simpler architecture than ResNet, in comparison to VGG, ResNet is a more intriguing architecture that is broader and takes significantly more computing power and training time.

VGG, on the other hand, outperforms ResNet in terms of precision. ResNet can achieve higher accuracy for CIFAR10 and MNIST datasets with sufficient preparation. Furthermore, it is clear that adding augmentation to a model increases the model's generalization, resulting in increased accuracy.

Furthermore, on the CIFAR dataset, hyper parameter tuning was difficult. The papers' original output figures were difficult to obtain, suggesting that the implemented models could be better.

VII. REFERENCES

[1] A. Krizhevsky, “One weird trick for parallelizing convolutional neural networks,” *CoRR*, vol. abs/1404.5997, 2014.

[2] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” *CoRR*, vol. abs/1512.03385, 2015.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[4] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition.” 2015.

[5] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet, “Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks.” 2014.

[6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going Deeper with Convolutions.” 2014.

VIII. APPENDIX

Training is timestamped please refere to code for more