Deeper Networks for Image Classification

Suyog Pipliwal 210634338
School of Electronic Engineering and Computer Science
Queen Mary University of London
Lonndon, United Kingdom
s.pipliwal@se21.qmul.ac.uk

***Abstract*—One of the main problems in computer vision is the image classification problem, which is concerned with determining the presence of visual structures in an input image. Image classification analyzes the numerical properties of various image features and organizes data into categories. In recent years, many advanced classification approaches, such as artificial neural networks, fuzzy-sets, and expert systems, have been widely applied for image classification, but each of them having some problems and their accuracy level is comparatively less.O ne of the advanced classification approaches is to use the deep convolutional neural network (CNN) architecture of deep learning which obtains successful results in solving many machine learning problems.The proposed method is trained and tested on MNIST and CIFAR10 dataset and achive classification accuracy of 99.7% and 84.9% respectively. The experimentation was done Macbook pro with M1 chip and Google Colab Pro.**

***Index Terms*—Computer vision, MNIST, CNN, VGG, Resnet, classification**

I. Introduction

Image classification is one the fundamental problem in computer vision that is is often used as a benchmark to measure the progess of image understanding. Convolutional neural networks have been the main design paradigm for image understanding tasks, as initially demonstrated on image classification tasks. In recent years, deep learning models that exploit multiple layers of nonlinear information processing, for feature extraction and transformation as well as for pattern analysis and classification, have been shown to achive high accuracy in image classification. Among them, CNNs have become the leading architecture for most image recognition, classification, and detection tasks. The big jump in this field is happen with the introduction of ImageNet Large Scale Visual Recognition Challenge(ILSVRC) [6] in 2010. The ILSVRC evaluates algorithms for object detection and image classification at large scale. This allow researchers to compare progress in detection across a wider variety of objects taking advantage of the quite expensive labeling effort. One of the ingredient to their success was the availability of a large training set, namely Imagenet [1] MNIST [8] and cifar10 [9]. All these large dataset allow training fof deeper classification model such as VGG [4] and RESNET [5]. Since then the ImageNet challenge is the de facto benchmark for computer vision classification algorithms. In 2014 VGG is was runner-up for this challenge with top-5 error rate of 7.3% andin 2015 Resnet was the winner with top-5 error rate of 3.57% .

A. Overview of CNN architecture

CNNs are feedforward networks in that information flow takes place in one direction only, from their inputs to their outputs. Just as artificial neural networks (ANN) are biologically inspired, so are CNNs. The visual cortex in the brain, which consists of alternating layers of simple and complex cells motivates their architecture. CNN architectures come in several variations; however, in general, they consist of convolutional and pooling (or subsampling) layers, which are grouped into modules. Either one or more fully connected layers, as in a standard feedforward neural network, follow these modules. Modules are often stacked on top of each other to form a deep model.

1) Convolutional Layers: The convolutional layers serve as feature extractors, and thus they learn the feature representations of their input images. The neurons in the convolutional layers are arranged into feature maps. Each neuron in a feature map has a receptive field, which is connected to a neighborhood of neurons in the previous layer via a set of trainable weights. Inputs are convolved with the learned weights in order to compute a new feature map, and the convolved results are sent through a nonlinear activation function. All neurons within a feature map have weights that are constrained to be equal; however, different feature maps within the same convolutional layer have different weights so that several features can be extracted at each location. More formally, the *k*th output feature map Y_k can be computed as

$$Y_k = f(W * x)$$

where the input image is denoted by x; the convolutional filter related to the *k*th feature map is denoted by W_k; the multiplication sign in this context refers to the 2D convolutional operator, which is used to calculate the inner product of the filter model at each location of the input image; and f(·) represents the nonlinear activation function

2) Pooling Layers: The purpose of the pooling layers is to reduce the spatial resolution of the feature maps and thus achieve spatial invariance to input distortions and translations. There are two type of pooling layers max pooling and average. Average pooling aggregation layers to propagate the average of all the input values, of a small neighborhood of an image to the next layer. Max pooling selects the largest element within each receptive field such that.

$$Y_{kij} = \max_{(p,q)\in R_{ij}} x_{kpq}$$

where the output of the pooling operation, associated with the kth feature map, is denoted byY_{kij}, x_{kpq} denotes the element at location (p, q) contained by the pooling region R_{ij} , which embodies a receptive field around the position (i, j). [7]

3) Fully Connected Layers: Several convolutional and pooling layers are usually stacked on top of each other to extract more abstract feature representations in moving through the network. The fully connected layers that follow these layers interpret these feature representations and learn a function between the high-level features given as an output from the convolutional layers.

II. Related Work

In recent years, CNN has gained popularity in computer vision tasks. Their ability to extract features from image data combined with the introduction of the gradient descent algorithm makes them the first choice in the image classification task. With the start of ILSVRC challenge in 2010 huge amount of data is avalibale to train and test such models. The main architecture that we going to foucs on are VGG16 and Resnet. Both of them perform very well in the challenge and show that such huge model can be train and develop succesfully.

1) VGG16: VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition”. The model achieves 92.7% top-5 test accuracy in ImageNet. It was one of the famous model submitted to ILSVRC-2014.It makes the improvement over AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another. VGG16 was trained for weeks and was using NVIDIA Titan Black GPU’s. VGG16 has a total of 138 million parameters. The architecture depicted below is VGG16.

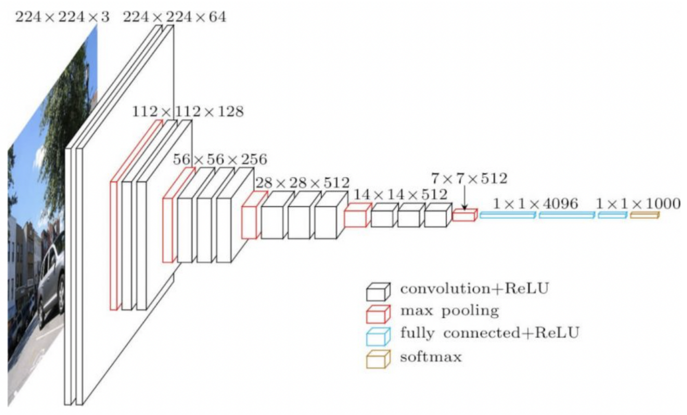


Fig. 1. The architecture of VGG16.

2) Resnet: While most of the state-of-art model are trying to take advantage of convolution and increasing depth of the model. This make model difficult to train them and the accuracy starts saturating and then degrades also. To solve this problem Resnet was introduce in 2015. ResNet, short for Residual Network is a specific type of neural network that was introduced in 2015 by Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun in their paper “Deep Residual Learning for Image Recognition”. To tackle this problem Resnet introduced ”skip-connection”. In standard cnn model the input ‘x’ gets multiplied by the weights of the layer followed by adding a bias term as

$$H(x) = f(Wx + b)$$

Now with the introduction of skip connection, the output is changed to

$$H(x) = f(x) + x$$

The skip connections in ResNet solve the problem of vanishing gradient in deep neural networks by allowing this alternate shortcut path for the gradient to flow through. The other way that these connections help is by allowing the model to learn the identity functions which ensures that the higher layer will perform at least as good as the lower layer, and not worse.

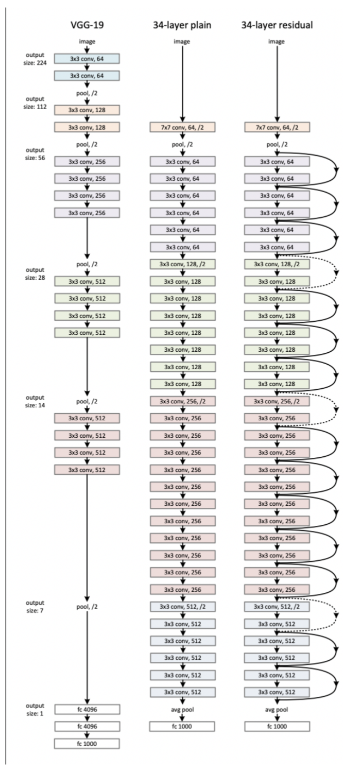


Fig. 2. Left: theVGG-19 [4] model as a reference. Middle: a plain network with 34 parameter layers.Right: a residual network with 34 parameter layers

III. Dataset

The dataset we wil be using MNSIT [8] and CIFAR10 [9]. The MNIST database of handwritten digits from 0 to

9 , has a training set of 60,000 examples, and a test set of 10,000 examples in 10 classes. It is a subset of a larger set available from NIST. The CIFAR-10 dataset consists of images of airplane,automobile,bird,cat, deer, dog, frog, horse, ship, truck. There are total 60000 in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

TABLE I
SAMPLE IN TRAINING DATA FOR EACH CLASS

Classes	Samples in MNIST	Samples in CIFAR10
1	5923	5000
2	6742	5000
3	5958	5000
4	6131	5000
5	5842	5000
6	5421	5000
7	5918	5000
8	6265	5000
9	5851	5000
10	5949	5000

A. *Preprocessing*

The images in both the dataset are resized to 256x256, transform from single channel to 3 channel images, croped from the center and are normalized to mean vector of [0.485, 0.456, 0.406] and standard deviation of [0.229, 0.224, 0.225].

IV. PROPOSED ARCHITECTURE

The VGG and RESNET have shown great success in image classification in the ImageNet dataset. These models are fundamentally divided into two sections one the feature extractor and then a classification block which learns a mapping between these high level feature and the output. We are using the same anology in the proposed architecture. There are two three block the model. First block extract the feature using VGG16 model, it produces a vector of shape(512,7,7). The second block extract feature using RESNET16 and also produce a vector of shape (512,7,7). Then these vector are concatenate to produce the most effiecent feature possible for given input. The third block is linear classifier which take these feature the learn the mapping between these feature and the output. The linear classifier consist of linear layer with relu as activation function with a dropout to prevent overfitting. The architecture is figure 3 There are total of 32,332,842 trainable

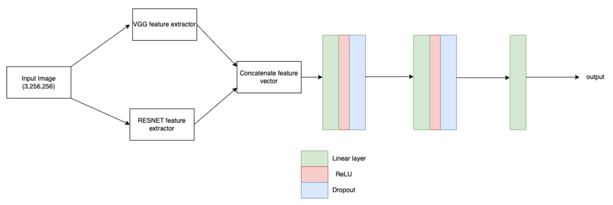


Fig. 3. Proposed Architecture

parameters.

V. EXPERIMENTATION

The model was trained using Cross-entropy as loss function and Stochastic Gradient Descent (SGD) as optimizer. The hyper parameter for training as shown in table II We trained

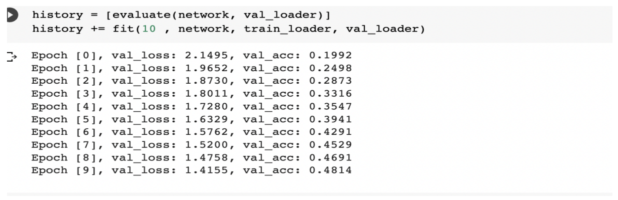
TABLE II
HYPER-PARAMETERS

Parameter	Value
learning rate	0.001
momentum	0.5
batch size for training	64
batch size for testing	100
epochs	100

and tested this model on two dataset MNIST and CIFAR10. The training was done in small steps of 10. For training and testing we divided the dataset into three parts, training dataset, validation dataset and testing dataset. The length of validation is 5000 samples for both the datasets with same hyper parameters. The training was done in phase in each aftet each 10*th* epoch we plot loss curve and tested the model and obtained confusion matrix

A. *Training on CIFAR10*

We will first discuss training and testing on CIFAR10 dataset. The training was done in small step of 10. The log for first 10 epochs is shown in figure 4 Even with first 10 epochs

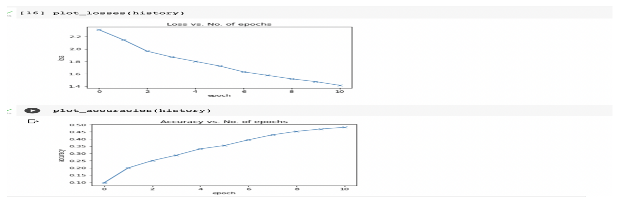


```
history = [evaluate(network, val_loader)]
history += fit(10 , network, train_loader, val_loader)

Epoch [0], val_loss: 2.1495, val_acc: 0.1992
Epoch [1], val_loss: 1.9652, val_acc: 0.2498
Epoch [2], val_loss: 1.8730, val_acc: 0.2873
Epoch [3], val_loss: 1.8011, val_acc: 0.3316
Epoch [4], val_loss: 1.7280, val_acc: 0.3547
Epoch [5], val_loss: 1.6329, val_acc: 0.3941
Epoch [6], val_loss: 1.5762, val_acc: 0.4291
Epoch [7], val_loss: 1.5200, val_acc: 0.4529
Epoch [8], val_loss: 1.4758, val_acc: 0.4691
Epoch [9], val_loss: 1.4155, val_acc: 0.4814
```

Fig. 4. First 10 epochs in CIFAR10

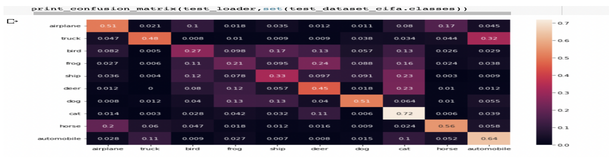
we obtained a good accuracy for some class suggesting the model is not overfitting the dataset. The corresponding loss and accuracy graph is shown in figure 5. At this point we



```
[16] plot_losses(history)
plot_accuracies(history)
```

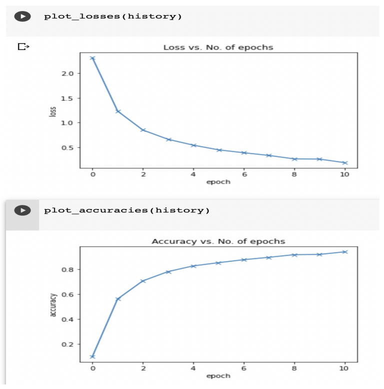
Fig. 5. loss vs epoch and Accuracy vs epoch graph for first 10 epoch

tested our model on CIFAR10 testing dataset and obtained the confusion matrix as shown on figure 6 Now for last 10 epochs we obtained the following loss curve as showen in figure 7.



```
print_confusion_matrix(test_loader,set(test_dataset_cifa.classes))
```

Fig. 6. Confusion matrix after first 10 epochs

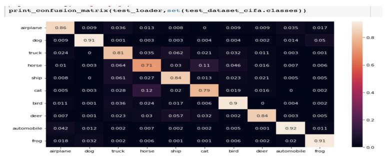


```
plot_losses(history)
```

```
plot_accuracies(history)
```

Fig. 7. loss vs epoch and Accuracy vs epoch graph for last 10 epoch

The smooth loss curve show the model is learning succesfully and is not overfitting the dataset. The confusion matrix obtained after training on testing dataset is figure 8
The average accuracy for all the class we obtained is 84.9%



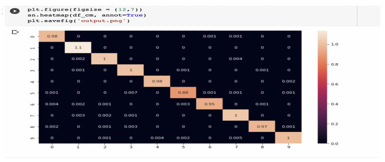
```
print_confusion_matrix(test_loader,set(test_dataset_cifa.classes))
```

Fig. 8. Final confusion matrix for CIFAR10 dataset

on CIFAR10 testing dataset. The whole training and testing process took around 18 hrs on google colab pro.

B. Training on MNIST dataset

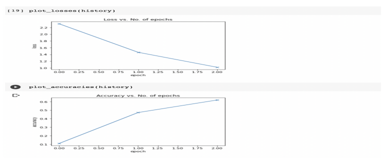
For training on the MNIST dataset, we follow the same approach of training and testing the model after each 10*th* epoch. But this dataset is very simple. We achive very high accuracy even after 10*th* epoch, so we deiceded to stop the training just after 10*th* epoch and printed out the confusion matrix as shown in figure 9. We achive the average accuracy



```
plt.figure(figsize = (12,7))
sn.heatmap(df_cm, annot=True)
plt.savefig('output.png')
```

Fig. 9. Confusion matrix for MNIST dataset

of 99.7% on this dataset and the crossponding loss curve is shown in figure 10



```
plot_losses(history)
```

```
plot_accuracies(history)
```

Fig. 10. Loss and Accuracy for MNIST dataset

VI. Result and Conclusion

For comparing the performance we are using accuracy as our parameter. We found that our proposed model achieve state-of-the-art performance on all the datasets. The performace comparsion is made in table III The performance of our proposed model is comparable with state-of-the-art model. In this paper, we presented an novel method for image classification by aggregating the convolutional features extracted from the different CNN models. The source code of model is publicly avaliable and hosted on 'https://github.com/suyog-pipliwaal/computer-vision'. Also, the source code is along with the report in the code folder. There are two files the code folder named "mnist.ipynb" and "cifar.ipynb", each file contained the same code except the dataset used is different. In mnist.ipynb we trained and tested model on MNIST dataset and in cifar.ipynb we used CIFAR10 dataset.

TABLE III
Comparsion of accuracy for different datasets.

Dataset	Accuracy
CIFAR10	84.7%
MNIST	99.7%

References

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In Conference on Computer Vision and Pattern Recognition, pages 248–255, 2009.
[2] Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine, 29(6), 141–142.
[3] Krizhevsky, Alex, Vinod Nair, and Geoffrey Hinton. "Cifar-10 and cifar-100 datasets." URl: https://www. cs. toronto. edu/kriz/cifar. html 6.1 (2009):
[4] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale visual recognition." arXiv preprint arXiv:1409.1556 (2014).
[5] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
[6] Olga Russakovsky*, Jia Deng*, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. (* = equal contribution) ImageNet Large Scale Visual Recognition Challenge. IJCV, 2015.
[7] Yu, Xiaoyuan, et al. "Key point detection by max pooling for tracking." IEEE Transactions on Cybernetics 45.3 (2014): 430-438.
[8] LeCun, Y., C. Cortes, and C. Burges. "The mnist dataset of handwritten digits (images)." NYU: New York, NY, USA (1999)
[9] Krizhevsky, Alex, and Geoffrey Hinton. "Learning multiple layers of features from tiny images." (2009): 7.