Deep Convolution Image Classification

ECS795P Deep Learning and Computer Vision

Atul Yadav 200349549

Abstract

In this report, I trained various deep convolutional neural networks to perform image classification on various datasets such as MNIST, which contains 60,000 images of digits 0 to 9, and CIFAR, which contains 60000 images of various categories such as car, plane, and so on. Furthermore, I used Batch Normalization after each convolution layer, which was not done in the original papers. And, on the MNIST Dataset, Google achieved the highest accuracy and the lowest loss, whereas on the CIFAR10 data, which is much more difficult to classify, GoogleNet outperformed other networks and achieved the highest accuracy.

Introduction

Deep convolutional neural networks have led to a series of breakthroughs in the last decade, not only because of better GPUs or larger datasets, but also because of new algorithms and techniques to improve previous state-of-the-art models. Although no new datasets were used in the ILSVRC, new models were introduced each year and performed admirably on the Image Net dataset. A convolutional neural network is a type of neural network. As computer vision has grown in popularity, various attempts have been made to improve the architecture of Krizhevsky et al. (2012) [1] to achieve high accuracy.

Since model depth is increasing, a new question arises: Does stacking more depth lead to better learning networks? [2]. However, adding more depth causes a vanishing gradient problem, which hinders our model's learning; however, this problem has been addressed by researchers by normalizing or by introducing auxiliary output layers in some models.

In this report, I worked on various state-of-the-art deep learning models that achieved high accuracy and compared which model performed well on our various datasets.

Related Work

AlexNet. Alex Krizhevsky [1] trained a deep convolution neural network to classify 1.2 million high-resolution images in the Image Net LSVRC-2010 competition into 1000 distinct classes. On test data, they achieved top 1 and top 5 error rates of 37.5 percent and 17.0 percent, respectively. The network has 60 million parameters, and 650,000 neurons consist of 5 convolutional layers, some of which are followed by a max-pool layer, and 1000 neurons in the final SoftMax layer. Dropout was used to reduce overfitting. They also used a variant of this model in the ILSVRC-2012 competition and won with a top 5 error rate of 15.3 percent.

GoogleNet. In the ImageNet Large-Scale Visual Recognition Challenge 2014, Christian Szegedy [3] proposed a new model called Inception, which achieved state-of-the-art classification and detection accuracy. The network was meticulously designed to allow for increasing network depth and width. They used an auxiliary outputs unit, which may have solved the vanishing gradient problem, and while testing, the auxiliary outputs were closed. The network had 22 layers.

Vgg. Karen Simonyan 2015 [4] increased the network depth to 16-19 layers in this paper, and the model accuracy improved significantly. They used three (3) small convolutional filters. This model is also applicable to other image recognition datasets, where it achieves excellent performance even when used as part of a simple pipeline.

ResNet. It becomes more difficult to train the model as the depth of the convolutional neural network increases, but in 2015, Kaiming He [2] proposed a residual learning framework to improve training in deep networks. They demonstrated that residual networks are much easier to optimise and can outperform deep networks in terms of accuracy. On the Image Net dataset, a residual network with 152 layers that was 8 times deeper than the VGG model but was still less complex won first place with a 3.7 percent error rate.

Dataset

The CIFAR10 Dataset contains 60000 32*32 color images divided into ten classes, each with 6000 images. There were 10,000 test images and 50,000 training photos in all. Each of the 10000 photos in the dataset is separated into five training batches and one test batch. The test batch contains exactly 1000 images from each class, chosen at random. The remaining images are distributed in training batches in a random order; however, some batches may contain more images from one class than others. The training batches contain exactly 5000 photos from each class between them. CIFAR 100 is like CIFAR 10, however there are 100 classes and 600 photos each class. There are 50000 photos for training and 10,000 photos for testing.

A training set of 60,000 images and a test set of 10,000 images make up the MNIST dataset of handwritten digits. It is a subset of NIST's wider collection. In a fixed-size image, the digits have been sized-normalized and centered. The original NIST data was adjusted to fit in a 20*20-pixel box while maintaining the aspect ratio. The images were centered in a 28*28 picture because of the anti-aliasing approach employed in the normalizing method. NIST's Special Database 3 and Special Database 1, which include binary images of handwritten numbers, were combined to create the MNIST database. SD-3 was created by NIST as a training set, while SD-1 was created as a test set. SD-3, on the other hand, is far cleaner and easier to distinguish than SD-1. SD-1 was taken from Census Bureau personnel, while SD-1 was taken from high school students.

Architecture

AlexNet

The architecture of our AlexNet network is summarized. It contains five convolutional layer and 3 fully connected layers.

ReLU Nonlinearity

The most common method for modelling a neuron's output f as a function of its input x is to use $f(x) = tanh(x)$ or $f(x) = (1 + e^{-x})^{-1}$. These non-linearities are substantially slower than non-saturating linearities during gradient descent training. Rectified Linear Units are neurons with this non-linearity.$f(x) = max(0, x)$ [5].

Pooling

The output of neighboring clusters of neurons in the same kernel map is summarized by pooling layers in CNNs. Pooling units that are next to one other do not overlap. A pooling layer, to be more accurate, can be conceived of as a grid of pooling. each summarizing a neighborhood of size z*z centered at the position of the pooling unit; units spaced s pixels apart.

Model Architecture

The network has eight layers with weights, as shown in Figure 1, with the first five being convolutional and the latter three being fully linked. The output of the last layer is fed into a 10-way SoftMax algorithm, which generates a distribution of 10 class labels (instead of 1000 in the original model). Every convolutional and fully connected layer's output is subjected to the ReLU non-linearity. The first convolutional layer filters the 224x224x3 input picture using 96 11x11x3 kernels with a stride of 4 pixels and then filters it with 256 5x5x48 kernels. The third, fourth, and fifth convolutional layers are linked without any pooling or normalizing layers in between. The third convolutional layer has 384 3x3x256 kernels coupled to the second convolutional layer's (normalized, pooled) outputs. The fourth convolutional layer has 384 3x3x192 kernels, and the fifth convolutional layer has 256 3x3x92 kernels. Each of the completely connected layers contains 4096 neurons.

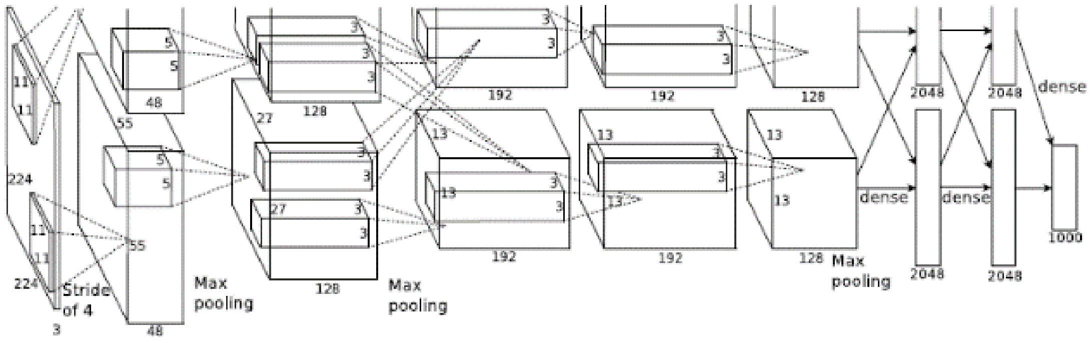


Figure 1: AlexNet [1]

GoogleNet

The basic idea behind the Inception architecture is to figure out how to estimate and cover an ideal local sparse structure in a convolutional vision network with conveniently available dense components. Yann LeCun's pioneering LeNet 5 network [6] inspired this name. A deeper and wider Inception network was utilized, with slightly lower quality, but it enhanced the outcomes when added to the ensemble. Here, the GoogleNet model is described in the Table 1.

Rectified linear activation is used in all convolutions, including those inside the Inception modules. Taking RGB color channels with mean subtraction, the size of the receptive field in our network is 224x224. The numbers 3x3 and 5x5 represent the number of 1x1 filters in before the 3x3 and 5x5 convolutions, there was a reduction layer. After the built-in max-pooling, the number of 1x1 filters in the projection layer may be seen in the pool proj column. Reversed linear activation is used in all these reduction/projection levels. When only layers with parameters are counted, the network has 22 layers (or 27 layers if we also count pooling). The total number of layers (independent building pieces) involved in the network's construction is around 100.

VGG

ConvNets are fed a fixed-size 224 224 RGB picture during training. The only preprocessing we do is subtract each pixel from the mean RGB value calculated on the training set. We utilise filters with a very small receptive field: 3 x3 (which is the smallest size to capture the notions of left/right, up/down, and centre) to send the image through a stack of convolutional layers. We also use 1x1 convolution filters in one of the setups, which may be thought of as a linear modification of the input channels (followed by non-linearity). The convolution stride is set to 1 pixel, and the spatial padding of the convolution layer input is set to 1 pixel for 3 convolution layers, preserving the spatial resolution after convolution. Five max-pooling layers, which follow some of the conv layer, oversee spatial. (max-pooling is not applied to all conv layers). Max-pooling is done with stride 2 over a 2x2 pixel window.

Three Fully Connected (FC) layers follow a stack of convolutional layers (which have varying depths in different architectures): the first two have 4096 channels each, the third does 1000-way ILSVRC classification and so comprises 1000 channels (one for each class) and 10 in our task. The soft-max layer is the final layer. In all networks, the completely connected levels are configured in the same way. Rectifier Linear Unit is applied to all hidden layers. Table 2 shows all the different versions of VGG architecture.

ResNet

ResNet has two simple designs: (i) the layers have the same number of filters for the same output feature map size: and (ii) the number of filters is doubled if the feature map size is halved to maintain the time complexity per layer. We use convolutional layers with a stride of 2 to do direct down sampling. A global average pooling layer and a 1000-way fully connected layer with SoftMax complete the network. In Fig. 3, there are a total of 34 weighted layers.

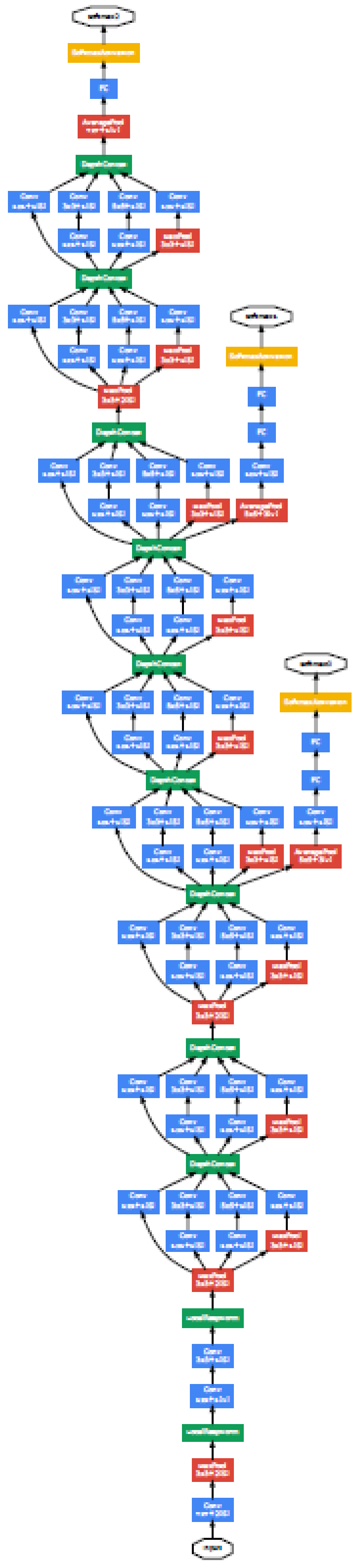
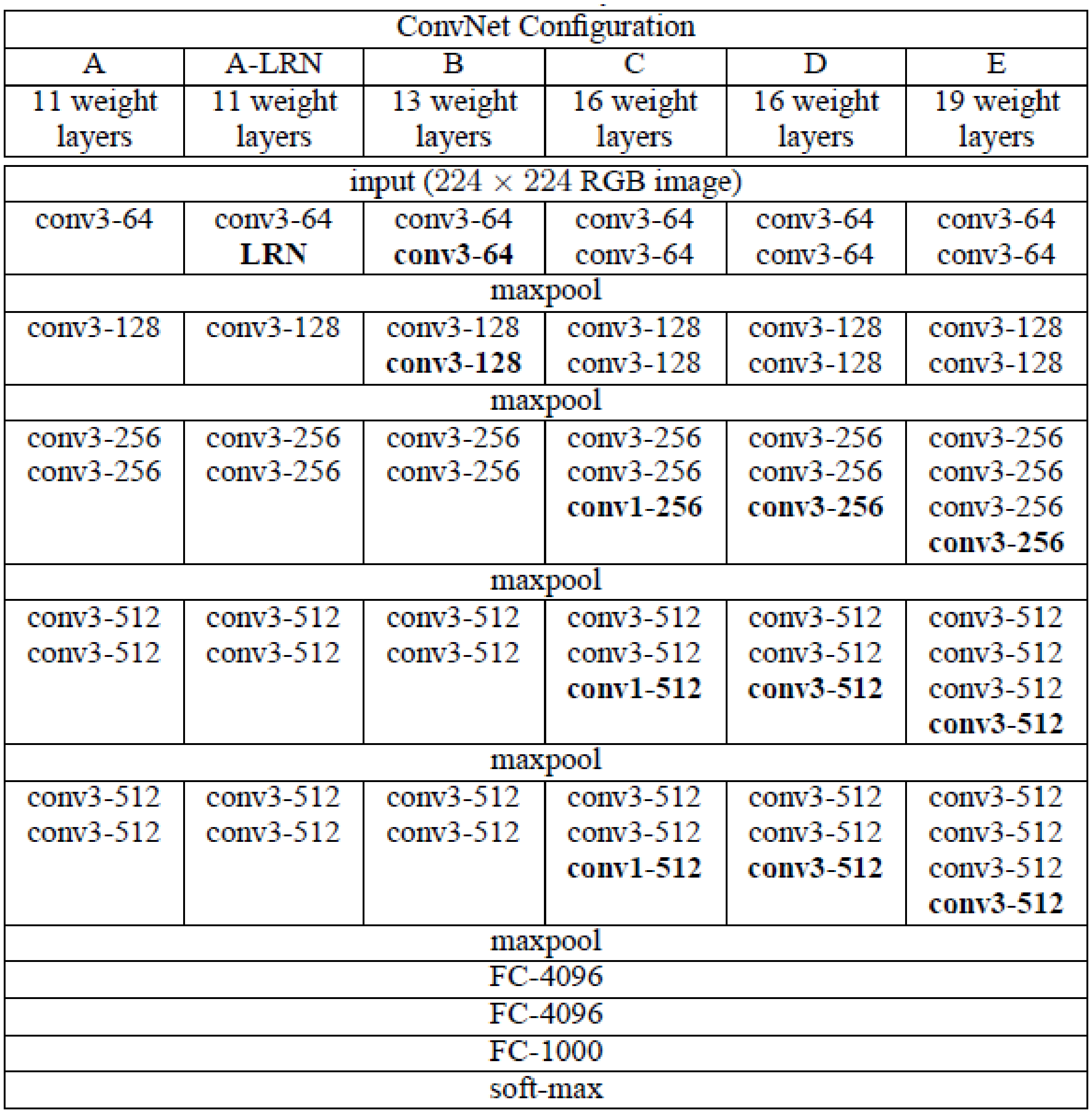


Figure 2: GoogleNet [3]



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	conv3-64	conv3-64	conv3-64	conv3-64
	LRN	**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	conv3-256	conv3-256	conv3-256	conv3-256
			conv1-256	**conv3-256**	conv3-256
					conv3-256
maxpool					
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
			conv1-512	**conv3-512**	conv3-512
					conv3-512
maxpool					
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
			conv1-512	**conv3-512**	conv3-512
					conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 1: Vgg [4]

Skip connections were introduced to the ordinary network, transforming it into its residual version. When the input and output dimensions are the same, the identity shortcuts can be utilized immediately (solid line skip connections in Fig. 3). When the dimensions increase in size (as seen by the dotted line skip connections in Fig. 3), we have two choices: (A) The shortcut still performs identity mapping, but with more zero entries to account for the increased dimensions. There is no additional parameter introduced by this option. (B) To match dimensions (1x1 convolutions), the projection shortcut is applied. When the shortcuts intersect feature maps of two sizes in both possibilities, they are done with a stride of 2.

Experiments

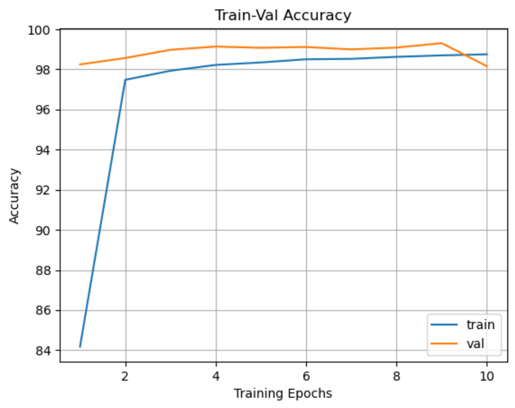
MNIST Classification

I ran experiments on the MNIST dataset, which contains 60,000 images from the training set and 10,000 images from the testing set. With the GPU's restricted power, a batch size of 32 was employed, with a learning rate of 0.001. Because the MNIST dataset is so quick to train, each model has a high level of accuracy.

We resized the images to 224 because the original model was designed for an input size of 224, and the only other data augmentations we used were normalization with mean and standard deviation.

AlexNet

This model has achieved above 98.7% and 99.3% accuracy in training and validation and the loss value was 0.028 in validation.



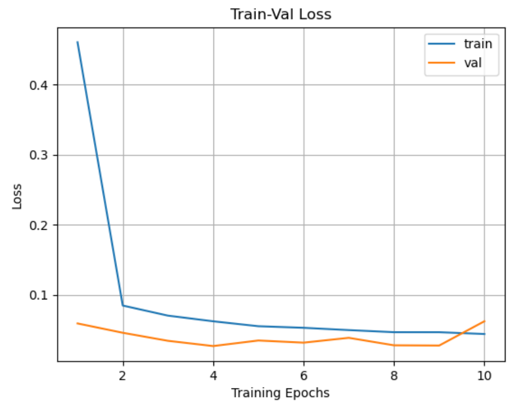
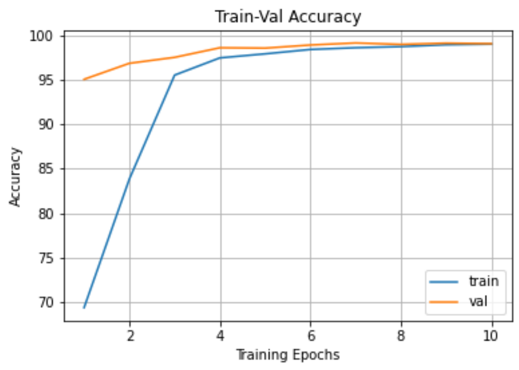


Figure 3 AlexNet Accuracy and Loss

VGG

This model has achieved above 99.04% and 99.09% accuracy in training and validation and the loss value was 0.04 and 0.035 in validation.



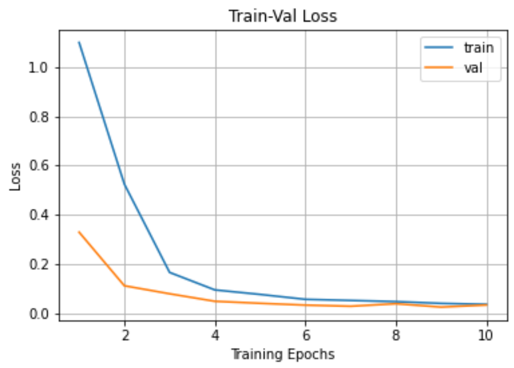
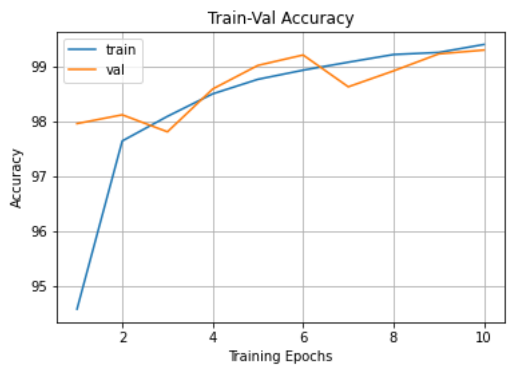


Figure 4 VGG Accuracy and Loss

ResNet50

This model has achieved 99.3% accuracy in validation set and the loss value was 0.020 in validation.



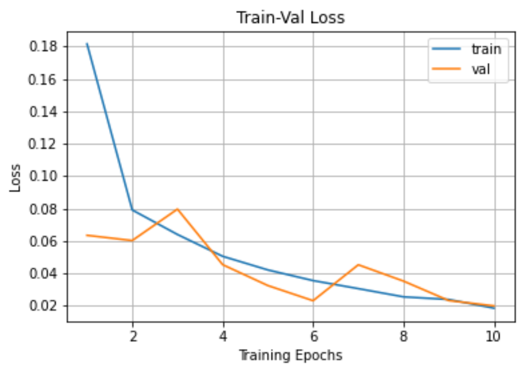
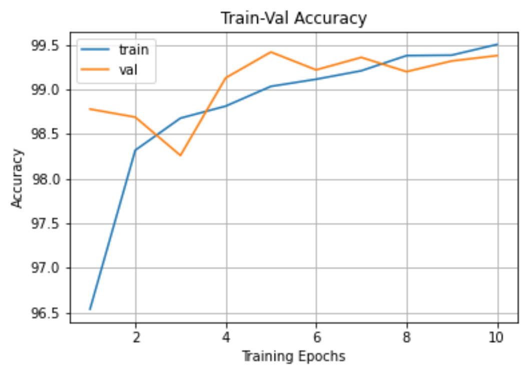


Figure 5 ResNet Accuracy and Loss

GoogleNet

This model has achieved 99.03% accuracy in training set and 99.42% on validation set and the loss value was 0.019 in validation.



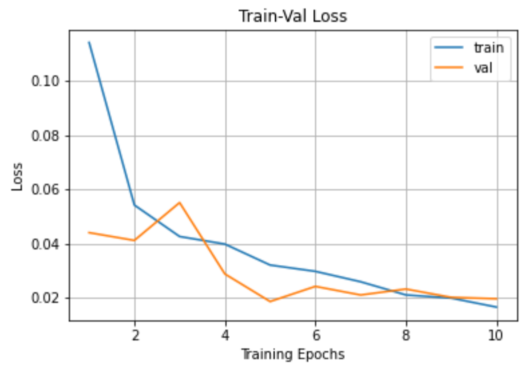
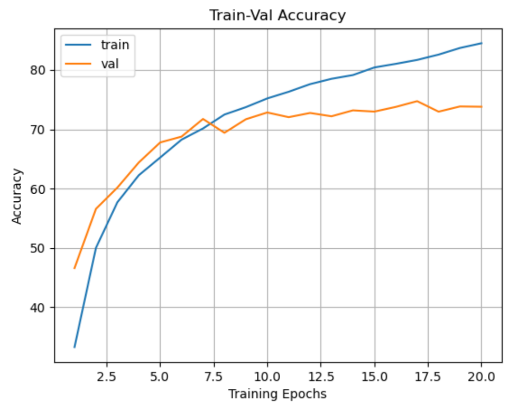


Figure 6 GoogleNet Accuracy and Loss

CIFAR10

AlexNet

Our AlexNet model has achieved accuracy 74.7 on validation set and 81 on training set. And after certain epochs our model starts overfitting.



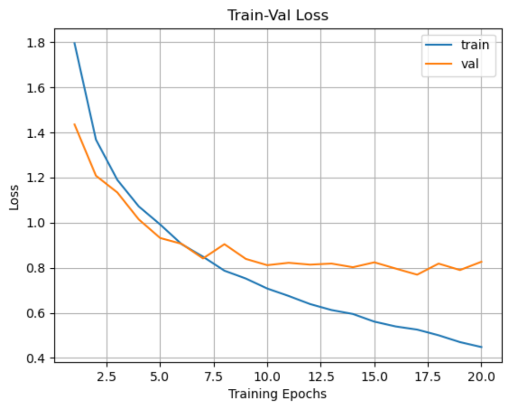
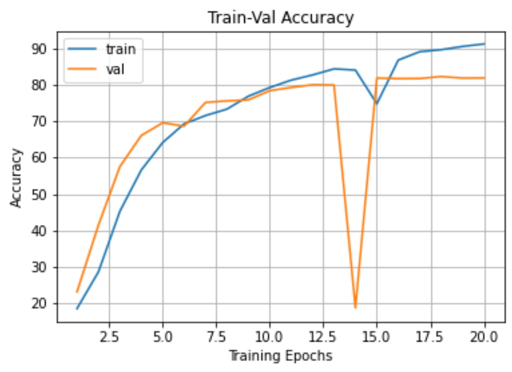


Figure 7 AlexNet Accuracy and Loss

VGG

Validation plot shoots around 13th epoch but overall accuracy achieved by VGG on CIFAR10 on validation set is 81.98.



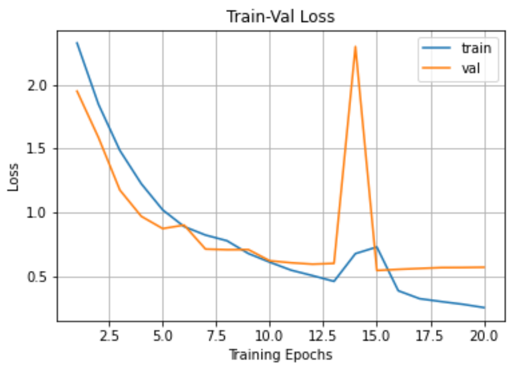
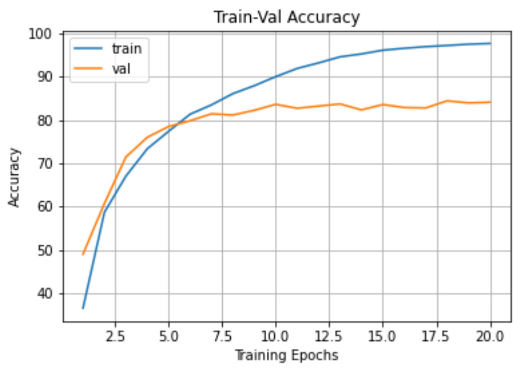


Figure 8 VGG Accuracy and Loss

ResNet50

Model starts overfitting after 7th epoch because it has 50 layers and the accuracy achieved by our model on validation set is 84.41.



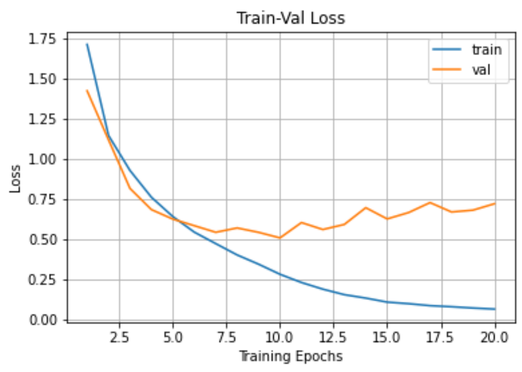
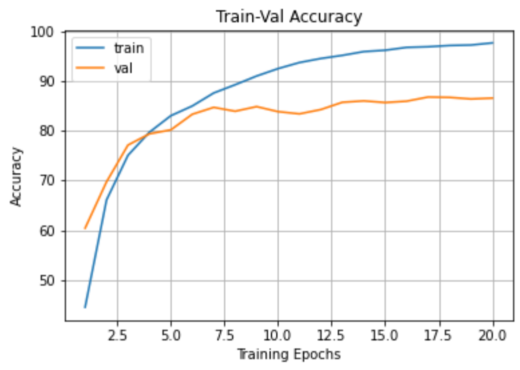


Figure 9 ResNet50 Accuracy and Loss

GoogleNet

Model achieves highest accuracy on CIFAR10 dataset 86.82.



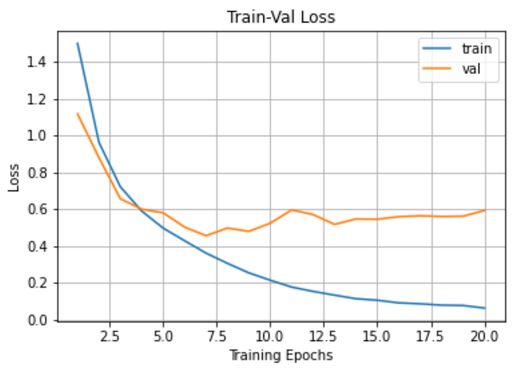


Figure 10 GoogleNet Accuracy and Loss

Results

In Table 3 On MNIST dataset every model performs very well but the model which achieves highest accuracy and lowest loss is GoogleNet. But CIFAR10 data is more complicated, and classes are much more difficult in classifying and model starts overfitting after few epoch because our networks are very deep and to avoid overfitting we might have to use larger dataset and use some regularization techniques such as Dropout [7] or L1 and L2 regularization.

Model	Dataset	Training Accuracy	Validation Accuracy	Validation Loss
AlexNet	MNIST	98.7	99.3	0.028
VGG16	MNIST	99.04	99.09	0.035
ResNet50	MNIST	99.4	99.3	0.020
GoogleNet	MNIST	99.03	99.42	0.019
AlexNet	CIFAR10	81.69	74.7	0.769
VGG16	CIFAR10	91.3	81.98	0.572
ResNet50	CIFAR10	97.18	84.41	0.670
GoogleNet	CIFAR10	96.94	86.82	0.5

Table 2: Results

Evaluation

Our models perform very well on MNIST Dataset but on CIFAR10 dataset their performance degrades, and model start overfitting. To avoid overfitting, we can introduce some regularization in our model such as dropout, L2 [s8] Regularization or we can decrease the depth of our model, we could use VGG11 instead of VGG16 or ResNet34 or ResNet18 instead of ResNet50. Because to train such kind of models we need much larger dataset or add regularization.

References

1. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems* 25 (2012): 1097-1105.

2. He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

3. Szegedy, Christian, et al. "Going deeper with convolutions." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

4. Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

5. Agarap, Abien Fred. "Deep learning using rectified linear units (relu)." *arXiv preprint arXiv:1803.08375* (2018).

6. LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, *1*(4), 541-551.

7. Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." *The journal of machine learning research* 15.1 (2014): 1929-1958.

8. Ng, Andrew Y. "Feature selection, L 1 vs. L 2 regularization, and rotational invariance." *Proceedings of the twenty-first international conference on Machine learning*. 2004.