Assignment 2 - ELL793

Shauryasikt Jena, 2018EE10500
Amokh Varma, 2018MT60527

Indian Institute of Technology, Delhi

Abstract

With the increase in the amount of data available in the world, it makes sense for us to try and make the best use out of it. Towards this goal, people have attempted to employ various data driven methods to carry out different forms of inferences on the images. One of the most commonly used data driven methods can be put togehter under the common heading of "Deep Learning" . In this assignment, we will see few of the common methods in Deep Learning applied to Computer Vision Problems. We will try to analyse the properties of these different methods as well. The methods discussed here are - " feed Forward Networks " , "Convolutional Neural Networks" and "Transfer Learning" .

Introduction

Convolutional neural networks formed the backbone behind a lot of the progress made in deep learning during the 2010s. These networks have revolutionized tasks such as image classification and object detection, but they also work remarkably well in other contexts such as text classification, speech recognition, or any domain where a filter can be used to detect similarities in regions of input data. CNN's produce state of the art results because they constantly preserve the spatial structure of an image. This means that better features will be extracted for the classification task at the end. A CNN takes a 32x32x3 image slides a filter of the same depth over the image to produce a 2D activation map which contains a score that measures the similarity between the filter and the image. The stack of activation maps is used for the next layers in the network which, depending on the architecture, is either a convolutional layer or pooling layer. In the final step the image is classified using a feedforward neural network.

Deep convolutional neural network models may take days or even weeks to train on very large datasets. A way to short-cut this process is to re-use the model weights from pre-trained models that were developed for standard computer vision benchmark datasets, such as the Imagenet image recognition tasks.

1. Transfer learning involves using models trained on one problem as a starting point on a related problem.
2. Transfer learning is flexible, allowing the use of pre-trained models directly, as feature extraction preprocessing, and integrated into entirely new models.

RESNET-18: ResNet-18 is a convolutional neural network that is 18 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images.

Method Used

Normalization

For normalization of images (RGB values ranging from 0-255), we standardise the data using mean and standard deviation of the MNIST/Imagenet datasets. We also try to divide by 255, but got better results with the former case.

Regularization

We used l2 regularization, also called ridge regression. In this regularization, squared magnitude of coefficients are added as a penalty term to the loss function.

Dropout

We randomly drop nodes during training to regularize and reduce overfitting of the dataset. Probabilistically dropping out nodes in the network is a simple and effective regularization method.

Early stopping

Early stopping is a form of regularization used to avoid overfitting when training a learner with gradient descent. Gradient Descent improves the learner's performance on data outside of the training set. Past that point, however, improving the learner's fit to the training data comes at the expense of increased generalization error. Early stopping rules provide guidance as to how many iterations can be run before the learner begins to over-fit.

Data Augmentation

For Tiny-CIFAR-10 Dataset, we used Image Data Augmentation to artificially increase the training datasize. We used the following data augmentation techniques:

1. Random Horizontal Flip

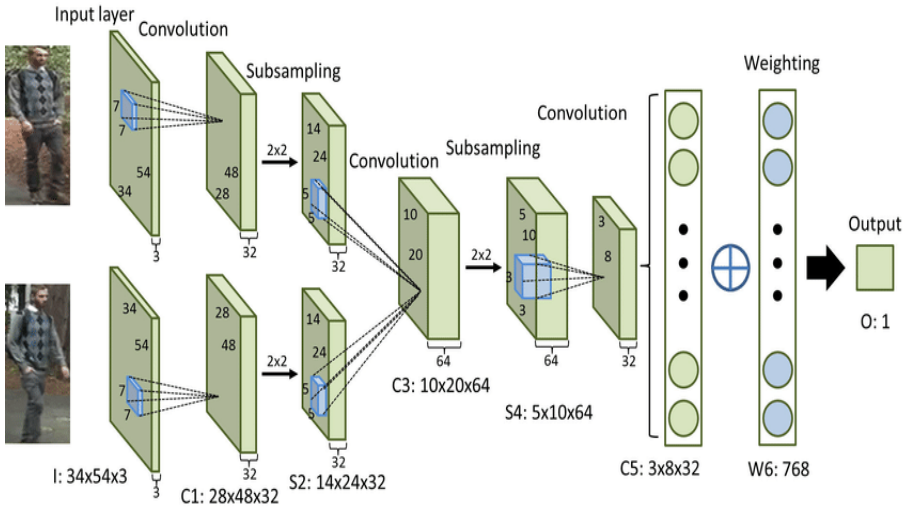

Figure 1: General CNN model

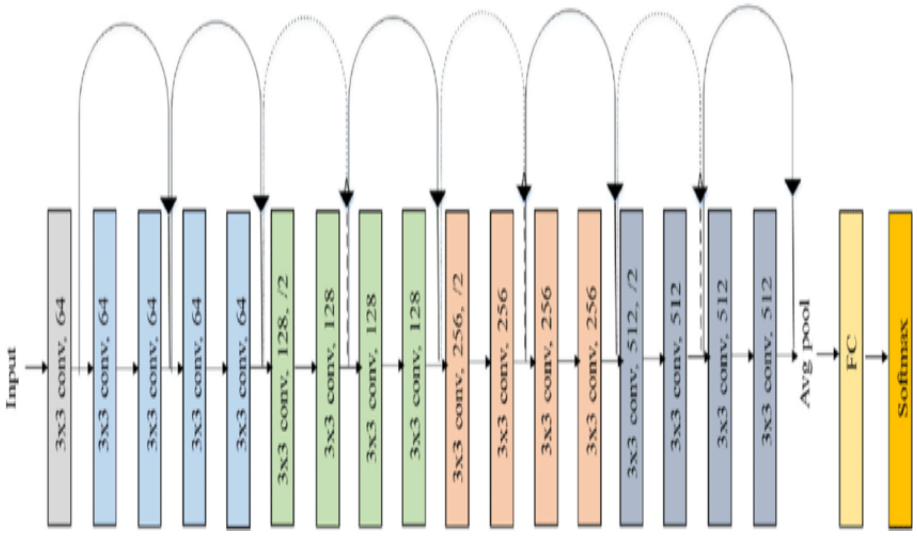

Figure 2: RESNET-18

2. Random Vertical Flip
3. Random Rotation

Transfer Learning

In transfer learning, the knowledge of an already trained machine learning model is transferred to a different but closely linked problem. Simple idea is to take a model trained on a large dataset and transfer its knowledge to a smaller dataset. In our case, ResNet-18 is a model trained on ImageNet (million images) and its knowledge is used on CIFAR-10 Dataset (a few thousand images). In implementation, we freeze the model's lower convolutional layers, add custom classifier, and train the classifier.

CNN

The architecture of the original convolutional neural network is shown in Figure 1. A Convolutional Neural Network (ConvNet) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. There are different types of layers like Pooling Layers, Padding Layers, Convolution Layers, Classification (Fully-Connected Layer), etc. in Deep CNN.

Residual Neural Network

ResNet-18 uses residual layers along with batch normalization, convolution, pooling layers. Residual neural network uses skip connections to form connection between layers and pass information without going through non-linearity and batch-normalization layers. Residual layers are added to avoid the vanishing gradient problem and to mitigate the degradation problem.

Dataset

CIFAR-10: The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

Tiny-CIFAR-10: For the Tiny-CIFAR-10 dataset, we take 500 images per class from CIFAR-10 for training, and then we used the same 10,000 images for testing as per the CIFAR-10 dataset.

MNIST: The MNIST dataset is an acronym that stands for the Modified National Institute of Standards and Technology dataset.

It is a dataset of 60,000 small square 28×28 pixel grayscale images of handwritten single digits between 0 and 9.

The task was to classify a given image of a handwritten digit into one of 10 classes representing integer values from 0 to 9, inclusively.

Results

We divide our results section into different subsections. Firstly, we talk about Problem 1, where we train feed-forward neural network with and without l2 regularization, dropout, early stopping. Afterwards, we discuss Problem 2, where we train CNN network using different variants as described. Finally, we visualize the activations of the CNN and later discuss the results.

Problem 1: MNIST

Here we build a simple feed-forward neural network from scratch, first without regularisation then with regularisation on MNIST data. We used log_softmax for output layer and ReLu for hidden layers. log_softmax is used for classification purposes and ReLu is added for non-linearity in the neural network. Below are the results:

a) : Without regularisation In this, we try different number of layers and different hidden layer dimensions, but we do not use regularisation. Further, there is no early stopping. Hence, we can clearly see that the model is severely overfitting by 250 epochs. The results can be seen in Table 1.

Sno	Layers	Dimension	Train Accuracy	Test Accuracy
1	1	500	99.12	**89.62**
2	2	500	**99.41**	88.24
3	3	500	99.37	87.21

Table 1: Performance on MNIST Dataset without regularisation using feed-forward neural network built from scratch

b) : With L2 regularisation Now we repeat the previous experiments with regularisations of 2 types. We use L2 regularisation and dropout with different rates. We can see that after a point, increase in dropout leads to decrease in performance. This is due to oversimplifation (or underfitting) of the model. Further, we also use early stop to stop after 10 epochs of no improvement in validation acccuracy.

Problem 2: CIFAR-10

In this, we try two different experiments. Firstly, we will train the network from scratch, with a resnet backcbone. Then, we train using the imagenet pretrained models.

a) : CNN from scratch Here, we let the model learn from scratch and also learn the residual layers of the Resnets. The results can be seen in Table 3 . We used a high value of batch size (128), as the images are quite small, so we need more data to get less noisy gradients. In this, we also do LR scheduling, where the learning rate is decreased by a factor of 10, whenever there is no improvement for 3 epochs. Further, we see overfitting in this case. Due to this, we use dropout in the last layer to improve our results. The best result is shown later.

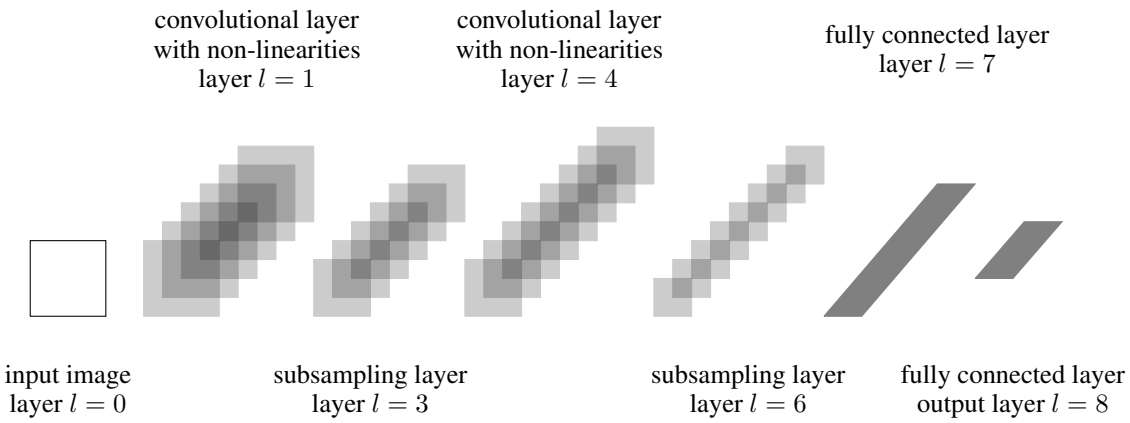

Figure 3: Architecture of a traditional convolutional neural network.

Sno	Layers	Dimension	Regularisation	Dropout	Train Accuracy	Test Accuracy
1	2	500	L2	None	99.14	97.62
2	2	500	None	0.2	**99.48**	**98.93**
3	2	500	L2	0.4	99.37	98.81
4	2	500	L2	0.6	98.36	98.21
5	2	500	L2	0.8	97.97	92.16

Table 2: Performance on MNIST Dataset with regularisation using feed-forward neural network built from scratch

Sno	Layers (after Resnet)	lr	Train Accuracy	Test Accuracy
1	1	1.0	77.21	67.62
2	1	0.1	84.23	69.93
3	1	0.01	**91.35**	**79.21**
4	1	0.001	86.71	76.36
5	1	0.0001	86.53	77.23

Table 3: Performance on CIFAR10 Dataset using CNN with resnet built from scratch

b) : Transfer Learning with ResNet-18 Note that, in the scratch case, we still initialise our model using the imagenet parameters. We however noticed that we could not get good results with transfer learning. This is because, the transfer learning model was unable to capture the complexity of the problem. We tackled this problem by increasing the number of fully connected layers on the top of the Resnet backbone. The results for this can be seen in 4 .

c) : Tiny-CIFAR-10 - CNN from scratch Here, we use a much smaller dataset , with only 500 images from each class. However, we will continue to test on the same dataset. Initially, we can see that our results are much worse than the previous sections explained in Table 3 and 4. But, to tackle this, we apply the different data augmentation methods that are explained in the methods section of the assignment. The results can be seen in Table 5

Visualization

Now, we try to look underneath the hood and try to find out the properties of the different layers of our CNN model. We do this in two parts :

Part 1 : Here, we look at the change in the same filter over training. We train the Resnet from scratch and plot the first few epochs. The result can be seen in

Part 2 : Here, we look at the change in the same filter over training. We train the Resnet from scratch and plot the first few epochs. The result can be seen in Figure 4. We can clearly how the filter is trying to focus on different parts of the image as the training continues.

After this, we plot the filters of different conv layers after training the model. The results for this can be seen in Figure 5 . Here, we can see that the starting layers have much less distinct shapes. The later layers have much more well defined shapes.

Conclusion

In this assignment, we learn how to use DL models to solve computer vision tasks. We learn how to use different kinds of hyperparameter tunings to improve our results. Further, we learn how to use the pretrained models to improve our performaceand visualise the the outputs of our convolutional layers to try and understand the kernel properties

Sno	Layers (after Resnet)	activation	Train Accuracy	Test Accuracy
1	1	relu	52.21	47.13
2	1	sigmoid	50.22	46.17
3	2	relu	56.33	48.19
4	2	sigmoid	53.02	47.46
5	3	relu	61.04	53.13
6	3	sigmoid	58.77	50.62
7	4	relu	**61.21**	**55.93**
8	4	sigmoid	57.12	49.22
9	5	relu	60.12	51.14
10	5	sigmoid	58.53	49.23

Table 4: Performance on CIFAR10 Dataset with Tranfer Learning

Sno	Augmentation	Train Accuracy	Test Accuracy
1	None	69.12	64.31
2	RH	72.34	68.13
3	RV	71.91	69.06
4	RR	71.18	69.11
5	RH+RV	73.15	69.71
6	RH+RV+RR	**74.27**	**70.13**
7	R	73.11	69.94

Table 5: Performance on Tiny CIFAR10 Dataset with CNN trained from scratch. RH, RV, RR , R stand for Random Horizontal Flip, Random Vertical Flip, Random Rotation and Random combination of above 3 respectively

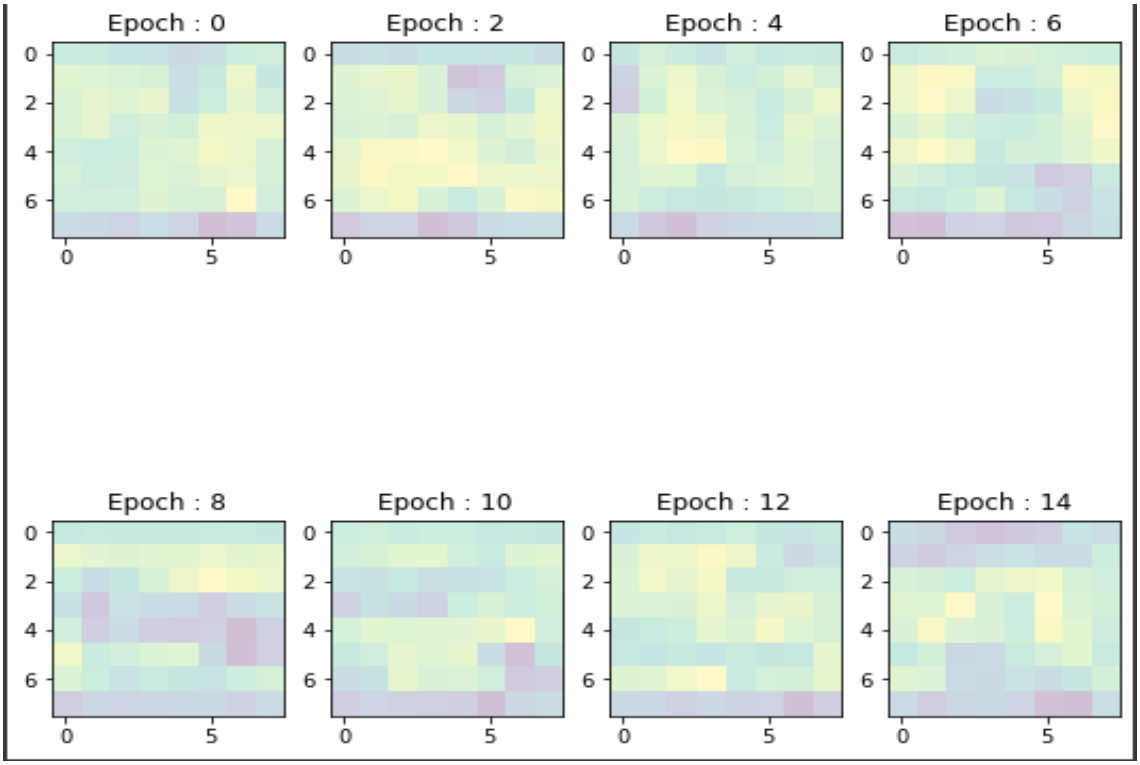

Figure 4: Visualisation of same layer with training

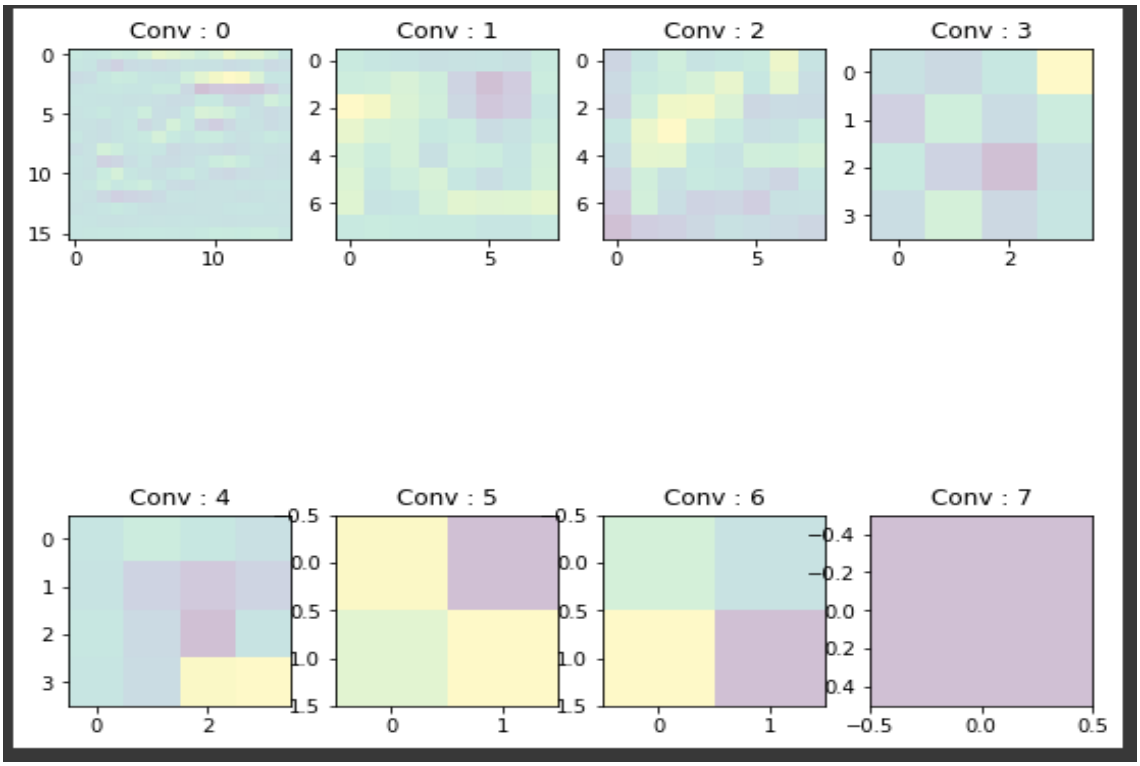

Figure 5: Visualisation of different layers