EE-569

INTRODUCTION TO DIGITAL IMAGE PROCESSING

HOMEWORK #4

DIXITH REDDY GOMARI
USC ID-3098766483

gomari@usc.edu

Problem 1: CNN Training and its application to the CIFAR-10 Dataset

a) CNN Architecture and Training

Convolutional Neural Networks are made up of neurons that have learnable weights and biases. ConvNet architectures assume that inputs are images, which allows to encode certain properties into the architecture. These parameters make the forward function efficient enough that reduce the parameters in the network. The layers of ConvNet have neurons arranged in 3 dimensions: width, height and depth. Here, we are given the input images in CIFAR-10 which are nothing but the input volume of activations. The volume of the images has dimensions 32X32X3 which are width, height and depth respectively. The neurons in a layer will only be connected to a small region of layer before it, instead of all the neurons in a fully connected manner. The final output layer would have the dimensions 1X1X10, i.e., eventually image will be reduced to a single vector of class scores, which has only depth as a dimension.

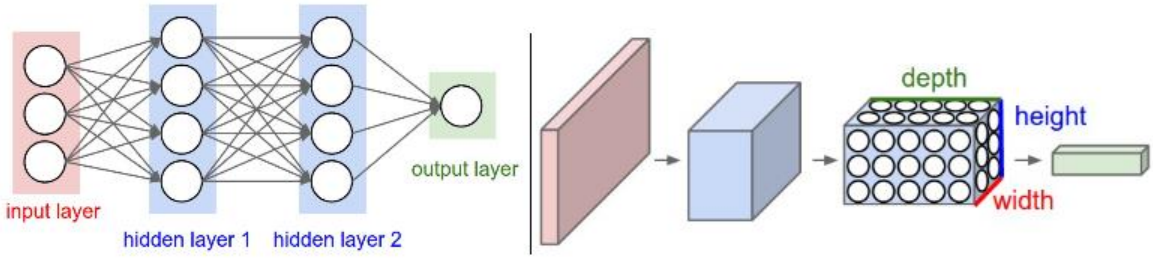


Figure 1 Visualization of ConvNet

Layers in ConvNets

There are 3 main layers which constitute a Convolutional Neural Network are Convolutional Layer, Pooling Layer and Fully-Connected Layer.

Let us consider an input which has dimensions 32X32X3 which are RGB values.

- Convolutional Layer computes the outputs of neurons that are connected to the local regions in an input.
- Each convolutional layer computes a dot product between their weights and the small region in which they are connected to the input.
- Here we are using 2 convolutional layers, of filter sizes 6 and 16 so the volume increases to 32X32X6 and 32X32X16.
- There are many activation functions like RELU, Leaky-RELU which apply elementwise activation function such as max(x,0) thresholding at 0.
- The activation function does not change the volume of the image which remains at 32X32X6 and 32X32X16.
- Max pooling layer will perform down sampling operation along the spatial dimensions, which results in reduction of dimension of the volume of the input.
- Fully connected layer will compute the scores of the class. Which results in further reduction of dimension, i.e., 1X1X10, where each number in those 10 numbers represent class score which represent the 10 categories of CIFAR-10.

- Both Convolutional layer and fully connected layers perform transformations that are a function of activations in the volume of the image and parameters of the image.
- Activation function and Max pooling layers will implement a fixed function.
- Gradient descent is applied to get the class scores which are consistent with labels in the training set for the images.
- Softmax function is one of the commonly seen classifiers which has a different loss function to that compared to SVM.
- In the softmax classifier, the function mapping stays unchanged when compared to SVM which is given as $f(x, W) = Wx$.
- We interpret the scores of the images as unnormalized log probabilities for each class and replace the hinge loss with cross-entropy loss which is of the form:

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right) \qquad \text{or equivalently} \qquad L_i = -f_{y_i} + \log\sum_j e^{f_j}$$

Where $f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}}$ is the **Softmax Function.**

ConvNet vs Multilayer Perceptron (MLP)

- Convolutional Neural Networks are nothing but MLPs with a special structure.
- In the perspective of image processing, CNNs have repetitive blocks of 2Dconvolutional kernels, repeatedly applied over each patch of the image.
- MLPs use independent weights at each spatial or temporal location to learn, which results in orders of magnitude more training data.
- And on top of this fundamental constraint in MLP, CNNs have many special techniques and layers, which makes it more efficient than a regular MLP.

Why CNNs work better than other Computer Vision Problems

- Since images exhibit a hierarchical structure i.e., pixels are the lowest level elements, they constitute to make up lines and curves, and they make up shapes. These shapes make up a complete image.
- CNN is a better mechanism when compared to other vision algorithms as it uses this hierarchical structure to find solutions.
- As we can see, in our architecture, the convolutional layer with the max pool layer interacts with the pixels of the image.
- The output obtained from this layer is nothing but the lines and curves.
- This output is again fed to another convolutional and pooling layer to obtain shapes.
- The repetitive process is what constitutes an image.
- Due to this ability to interact with deeper layers to establish a solution is what makes CNNs much better than other Computer Vision problems.

Loss Function and Back Propagation Optimization:

- Loss function is nothing but a function that maps an event onto one or more values which represents some sort of cost associated with that event.
- Loss function is generally used in optimization procedures to reduce the cost to obtain the desired results.
- It is mainly used in the back-propagation mechanism to reduce the costs involved in training the images.
- Backward propagation is a common method to train neural networks used with optimization methods such as gradient descent in this case.
- There are two phases in this algorithm, one is propagation and the other one is weight update.
- When input volume of image is given to the network, it is propagated forward into the network, layer by layer, until it reaches the final layer.
- Then this output is compared to the desired output using a loss function.
- Comparison evaluates the error value for each neuron in the output layer.
- These error layers are propagated backwards, starting from the output layer, so that each neuron has some error value which represents its contribution to the output layer.
- The back propagation uses this error values to calculate the gradient of the loss function w.r.t weights in the network.
- In the next phase, the gradient obtained is fed to the optimization method, gradient descent in this case, which updates the weights to minimize the loss function.
- This mechanism usually requires known outputs for each input value to find the loss function gradient, therefore it is a supervised learning method.

b) Application of the CNN to CIFAR-10 Dataset

Preprocessing steps and random network initialization schemes

Data Augmentation:

- Since training a neural network is done on a lot of images to achieve a good performance.
- If the original image data set is limited, then data augmentation boosts the performance of the neural net.
- There are several ways to perform data augmentation, such as horizontally flipping, color jittering, random corps etc.
- Multiple combinations of different processing can also be used such as rotation of the images and random scaling at the same time.
- In this case, we are rotating the images by an angle of 25^{o}.
- Data augmentation can also be done by raising the saturation and value of all the pixels to a power between 0.25 and 4 and multiply these by factor between 0.7 and 1.4 and add them to a value between -0.1 and 0.1.
- We can also add a value between -0.1 and 0.1 to the hue of all the pixels in the patch of the images.

Pre-Processing techniques:

- There are several preprocessing approaches, such as zero-center the data followed by normalization and PCA whitening technique.
- This process is normally implemented to make the features fall under the range -1 and 1.
- This is normally applied if the features have different ranges.
- Here, since we are using images, the range is already in between 0 and 255, so the procedure would hardly make any difference in the observations.
- In PCA Whitening, the data is first centered. Then, we compute the covariance matrix that gives us the correlation structure in the data.
- These methods are not generally used in the neural networks as they do not offer much difference to the observations.

Initializations:

It is very important to initialize its parameters. This can be done by:

All zero initialization

- In this initialization procedure, we set all the parameters and weights to zero, which is expected to give better results, but in practice that is not the case.
- It is so because if every neuron in the neural network computes the same kind of output, that means that they compute the same gradients during back propagation and undergo same parameter updates.

Initialization with random numbers

- Instead of initializing the weights to zero, we assign them to some small random numbers.
- The idea is that the neurons updates is random and unique in the beginning, so they will compute distinctive updates and integrate themselves as diverse parts of the full network.
- This method still has very little impact on the performance of the network.

Calibrating the Variances

- Here we normalize the variance of each neuron output to 1 by scaling its weight vector by square root of its number of inputs.
- This makes sure that all neurons in the neural network initially have approximately the same distribution of the output and amazingly improves the convergence rate.

Rectified Linear Units

- It is a robust initialization method that considers rectifier nonlinearities.
- This method enables to train extremely deep rectified models directly from scratch and to investigate deeper or wider network architectures.
- It improves model fitting with nearly zero extra computational cost and little overfitting risk.

Here, we have accuracy plotsfor training and testing data for different parameters.

Case 1:

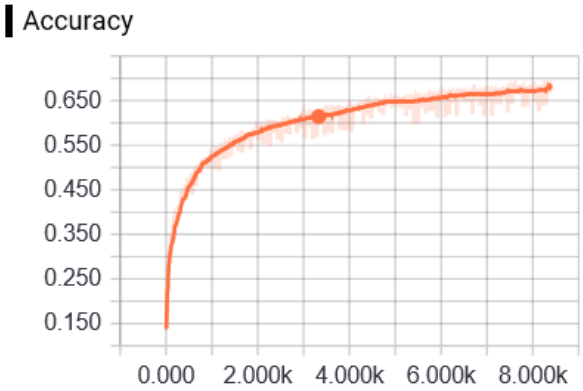
Activation: ReLU

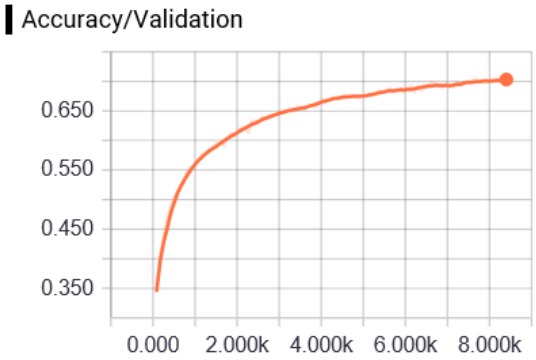
Epoch: 100

No initialization

Batch Size: 600

Learning Rate: 0.001





Case 2:

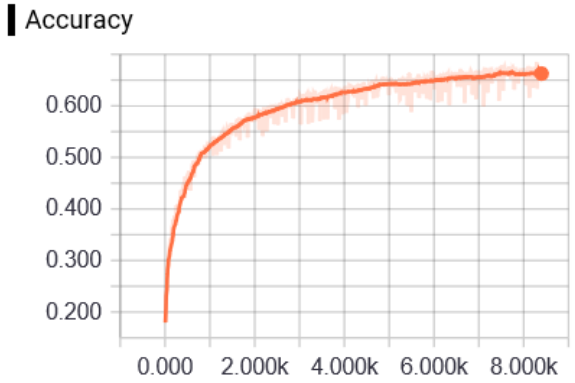
Activation: ReLU

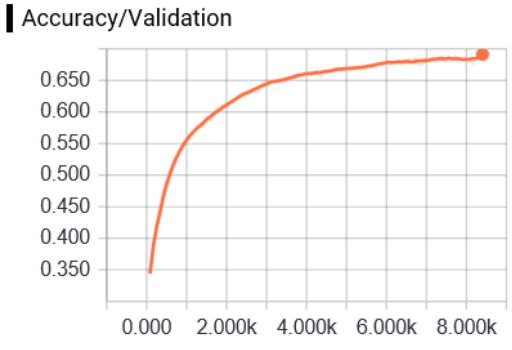
Epoch: 300

No initialization

Batch Size: 200

Learning Rate: 0.001





Case 3:

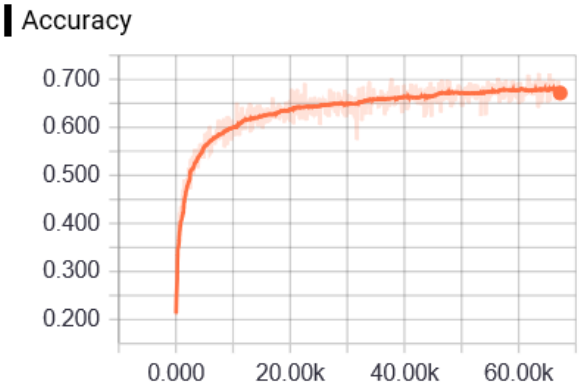
Activation: Leaky ReLU

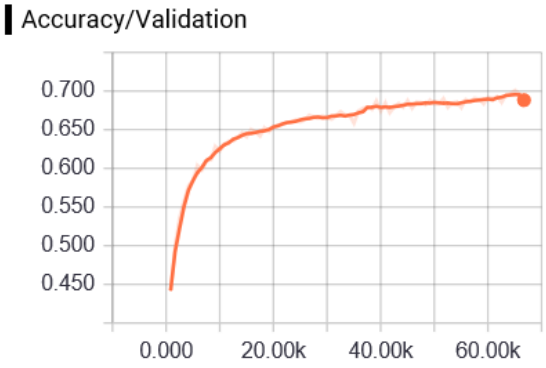
Epoch: 100

No initialization

Batch Size: 600

Learning Rate: 0.001





Case 4:

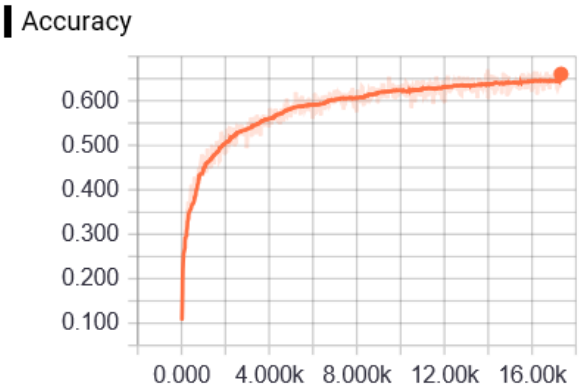
Activation: Leaky ReLU

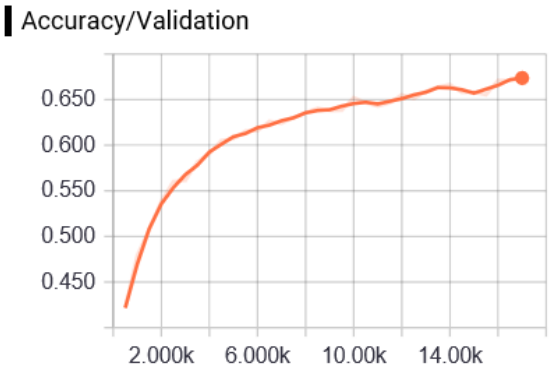
Epoch: 100

Initialization: xavier

Batch Size: 200

Learning Rate: 0.001





Case 5:

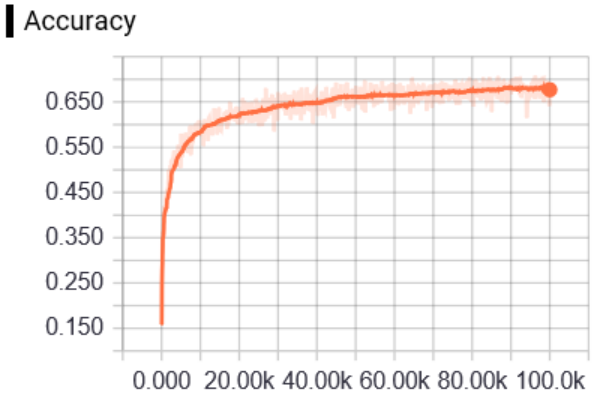
Activation: ReLU

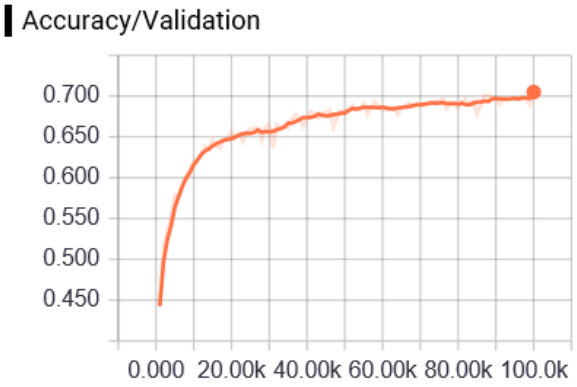
Epoch: 100

Initialization: xavier

Batch Size: 50

Learning Rate: 0.00095





Discussion:

- As we can see, that I have performed training of the images on the neural network, followed by classifying them into their respective classes.
- The accuracies obtained in training and testing the image data is shown above for various parameters such as type of activation, batch size, epoch etc.
- Typically, the training accuracy for the given parameters has an average of 68-69% and testing accuracy of around 69-70%.
- For a case, as in 4 the difference in training and testing accuracy is greater when compared to other cases in the lot.
- This might be due to overfitting problem occurred due to the specified parameters.
- Here we can see for case 5 we get better results, as the batch size is very small which trains the network more accurately, and learning rate is very small which makes it slow but eventually made it reach the optimum value.
- This accuracy of classification can be improved by either varying filter dimensions or increasing the depth of layers.

c) K-means with CNNs

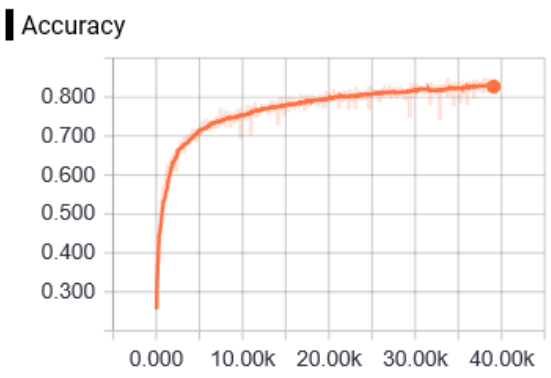
- Consider 1000-1500 random data samples from the images given and extract 15 random patches from those images.
- We need to perform k-means clustering as there are 6 filters, we put k=6.
- In the next step, we need to perform convolutions on the random data samples obtained in the first step by the filters initialized in the first stage of the network.

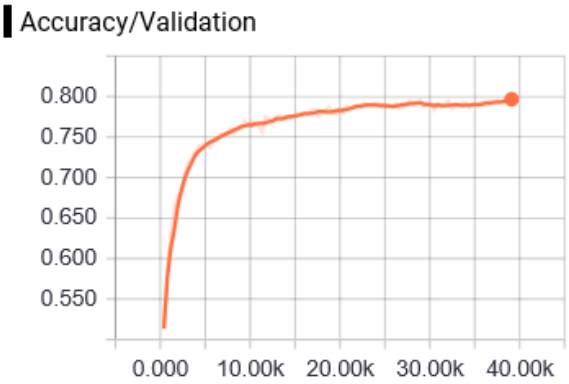
- This produces a 28X28X6 tensor for each of the data sample.
- Repeat the above process for the filters in the next layers.
- Once the network is set up, we need to perform k-means on the images.
- Since we have a patch size of 5X5X3, rearrange this into a 1D vector, so it will constitute to 75X1 vector.
- For each patch, we get this 1D vector, which we need to stack up, as the height of these stacks represent pixel values.
- Each 1D vector can be considered as a feature, so we perform k-means clustering on these 1D vectors and obtain the mean centroids of each cluster.
- Once these centroids are obtained, these can be taken as reference for updating the weights of the next layers.

Problem 2: Capability and Limitation of Convolutional Neural Networks

a) Improving your network for CIFAR-10 Dataset

- Based on the LeNet-5 architecture given in problem 1, I have improved its performance by tweaking some of its parameters.
- I have taken same number of layers in building my improved network, I have changed few filter parameters used in the previous problems.
- Here in the first layer I have increased the filter coefficient to 28 with patch size 5X5, followed by max pooling layer with size 2X2.
- In the next convolutional layer, I have used fc of 60 and patch size 5X5 with the same max pooling layer as the above.
- The fully connected layer has coefficient of 240, 160 and 10 with activations of relu, relu and softmax.
- These parameters has given an accuracy given below:





Discussion:

- As we can see, the accuracy has drastically increased by 10% as compared to the previous results.
- We can conclude that, doing data augmentation, or increasing the filter coefficients improves the classifying accuracy of a neural net.

b) State-of-the-Art CIFAR-10 Implementation

All you need is a good init by Dmytro Mishkin, Jiri Matas:

- In this paper, they have proposed a simple layer-sequential unit-variance initialization for weight initialization for deep net learning.
- This method mainly involves two phases; the first step involves pre-initialization of weights of each convolution layer with orthonormal matrices.
- Second phase proceeds from the first to the final layer, normalizing the variance of the output of each layer to be equal to 1.
- There are many ways of pre-initializations of weights of the convolution layer, such as ReLU:
    a) It is a robust initialization method that considers rectifier nonlinearities.
    b) This method enables to train extremely deep rectified models directly from scratch and to investigate deeper or wider network architectures.
    c) It improves model fitting with nearly zero extra computational cost and little overfitting risk.

  Random walk initialization:

    a) It keeps constant log of the norms of the back propagated errors.
    b) This does not give the desired results in this initialization mechanism.

  Knowledge distillation and Hints initialization:

    a) It uses mimicking teacher network predictions and internal representations, rather than minimizing the softmax loss.
- Layer sequential unit-variance initialization: this can be done by filling the weights with Gaussian noise with unit variance.
- In the next step, decompose them to orthonormal basis with OQ or SVD decomposition and replace weights with one of the components.
- Then the LSUV process estimates output variance of each convolution and inner product layer and it scales the weight to make the variance equal to 1.
- The influence of selected mini-batch size on estimated variance is almost negligible in wide margins case.
- This scheme is an orthonormal initialization combined with normalization performed on the first batch.
- The batch normalization is like the unit variance normalization procedure, whereas, initial ortho-normalization of weights matrices de-correlates layer activations.
- Such normalization is computationally highly efficient in comparison with full batch normalization.
- This mechanism is implemented on the CIFAR-10 dataset which gave the results given below.
- It produces networks with test accuracy better than standard methods.
- It is equally good when compared to complex networks such as FitNets.

Accuracy on CIFAR-10/100, with data augmentation

Network	CIFAR-10, %	CIFAR-100, %
Fitnet4-LSUV	**93.94**	70.04 (**72.34**†)
Fitnet4-OrthoInit	93.78	70.44 (72.30†)
Fitnet4-Hints	91.61	64.96
Fitnet4-Highway	92.46	68.09
ALL-CNN	92.75	66.29
DSN	92.03	65.43
NiN	91.19	64.32
maxout	90.62	65.46
MIN	*93.25*	*71.14*
Extreme data augmentation		
Large ALL-CNN	95.59	n/a
Fractional MP (1 test)	95.50	68.55
Fractional MP (12 tests)	**96.53**	**73.61**

Comparison between the above network and previous network:

- We can clearly see, there is a vast difference in the accuracy obtained in the proposed method and previous method.
- We obtain a whooping accuracy of 93% in the mechanism proposed by Dmytro Mishkin, Jiri Matas.
- As the above proposed method is very simple to execute as it only involves selecting good initialization methods.
- Whereas the previous methods, use a lot of layers when compared to this method, which makes it more computationally complex when compared.
- Coming to pros of the above method, it takes very less number of computations to obtain such a good result just with good initializations.
- The major disadvantage is that, without proper selection of initial weights, this method would horribly fail.
- The pros of the networks in problem 1 is that it does not require, initial pre-processing does not play a major part in determining the accuracy of the system, which reduces the computation steps.
- Major disadvantage with baseline and improved CNNs is that, they do not produce good accuracies in the classification of images.