Joshua Zapusek, Josue Perez, Aziz Boudy

CMPEN/EE 454

Sept 29, 2020

Project 1: Forward Pass Convolutional Neural Network

Part 1: Project Summary

In this project, we implemented an eighteen-layer forward pass convolutional neural network. These eighteen layers consisted of five Linear Rectification Units, six Convolutions, three Maximum Pooling layers, one Fully Connected layer, and one SoftMax layer. The layers, when operating sequentially, working to produce meaningful assessments as to the classification of an image dataset (cifar10).

We implemented these functions through external methods, unified by a main skeleton script in MATLAB. As such, the input to the network was a 32x32x3 colored image, and after moving through all other seventeen layers, was an array of probabilities indicated what image class the test image belonged. There were ten classes: airplane, automobile, bird, cat, deer, dog, frog, horses, ship, truck.

We were tasked with computing the accuracy of our network. For this assessment, we compared ground truth values of the input images to that of the predicted output classifications. With this information, we were able to compute the classification rate of the CNN.

The primary objective of this project was to perform some practical applications of convolution (covered in lecture) and to get familiarized with using MATLAB in a computer vision setting.

Part 2: Function Implementation

Image Normalization

Input: 32x32x3 image

Output: 32x32x3

Purpose: To normalize the color channel of the image for further image processing

$$\text{Out}(i, j, k) = \text{In}(i, j, k)/255.0 - 0.5$$

As seen above, this function simply takes the image, divides by 255 (each pixel having 256 possible values) and subtracting from the result by 0.5.

ReLU

Input: NxMxD matrix

Output: NxMxD matrix

Purpose: To convert any possible negative value in input to a zero

$$\text{Out}(i,j,k) = \max(\text{In}(i,j,k)\ ,\ 0)$$

To implement this function, we simply took the maximum of the input array and zero.

MaxPool

Input: 2Nx2MxD matrix

Output: NxMxD matrix

Purpose: To reduce the overall size of the array in, often referred to as down sampling. Here, the aim is to reduce the dimensions of the image while maintaining key feature details.

$$\text{Out}(i,j,k) = \max(\{\ \text{In}(r,c,k) \mid (2i-1) < r < 2i \text{ and } (2j-1) < c < 2j\ \})$$

Convolution

Input: NxMxD matrix

Output: NxMxD matrix

Purpose: Apply a set of linear filters and biases to further glean more data as to the image in question. This method is the primary means to extracting meaning features from our images.

$$\text{Out}(:,:,l) = \sum_{k=1}^{D_1} F_l(:,:,k) * \text{In}(:,:,k) + b_l\ .$$

Essentially, we apply a number of filters (D2) to our input image, so the resulting array will have a depth of D2. So, for each filter in the filter bank we run a convolution (using imfilter) for each NxM over D1 (with filter RxC << NxM) i.e. the result of the first loop over 'l' (in equation above) is an NxM sheet in with D2 sub 0.

For this function, we did not use vectorized format, as we are new to MATLAB and were unsure how to get it to work properly. Rather, we simply implemented a double loop in the fashion outlined above, were the convolving takes place inside the nested loop (i.e. for each D1).

Full Connected

Input: NxMxD matrix

Output:1x1xD matrix

Purpose: To obtain the final weights for each possible class when undergoing image classification.

$$\text{Out}(1,1,l) = \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{D_1} F_l(i,j,k) \times \text{In}(i,j,k) + b_l$$

As seen above, the computation requires the sum of a sum of a sum. Being new to MATLAB, we did not know how to vectorize a triple loop involving the multiplication of multidimensional arrays. So, we implemented a nested for loop to compute Out (seen above). In addition, out function was actually a quadruple loop, as we needed to loop through all l (or filter banks) seen above.

SoftMax

Inputs: 1x1xD matrix

Outputs: 1x1xD matrix

Purpose: To convert the set of scalar vales from fully connected layer into a set of probabilities for determining the images class.

$$\text{Out}(1,1,k) = \frac{\exp(\text{In}(1,1,k)-\alpha)}{\sum_{k=1}^{D}\exp(\text{In}(1,1,k)-\alpha)}$$

$$\alpha = \max_{k}\ \text{In}(1,1,k)$$

As seen above, our implementation was to simply compute alpha, then compute Out and return.

A table showing all eighteen layers and their parameters can be seen below:

layer #	layer type	input size	filterbank size	num biases	output size
1	imnormalize	$32 \times 32 \times 3$			$32 \times 32 \times 3$
2	convolve	$32 \times 32 \times 3$	$3 \times 3 \times 3 \times 10$	10	$32 \times 32 \times 10$
3	relu	$32 \times 32 \times 10$			$32 \times 32 \times 10$
4	convolve	$32 \times 32 \times 10$	$3 \times 3 \times 10 \times 10$	10	$32 \times 32 \times 10$
5	relu	$32 \times 32 \times 10$			$32 \times 32 \times 10$
6	maxpool	$32 \times 32 \times 10$			$16 \times 16 \times 10$
7	convolve	$16 \times 16 \times 10$	$3 \times 3 \times 10 \times 10$	10	$16 \times 16 \times 10$
8	relu	$16 \times 16 \times 10$			$16 \times 16 \times 10$
9	convolve	$16 \times 16 \times 10$	$3 \times 3 \times 10 \times 10$	10	$16 \times 16 \times 10$
10	relu	$16 \times 16 \times 10$			$16 \times 16 \times 10$
11	maxpool	$16 \times 16 \times 10$			$8 \times 8 \times 10$
12	convolve	$8 \times 8 \times 10$	$3 \times 3 \times 10 \times 10$	10	$8 \times 8 \times 10$
13	relu	$8 \times 8 \times 10$			$8 \times 8 \times 10$
14	convolve	$8 \times 8 \times 10$	$3 \times 3 \times 10 \times 10$	10	$8 \times 8 \times 10$
15	relu	$8 \times 8 \times 10$			$8 \times 8 \times 10$
16	maxpool	$8 \times 8 \times 10$			$4 \times 4 \times 10$
17	fullconnect	$4 \times 4 \times 10$	$4 \times 4 \times 10 \times 10$	10	$1 \times 1 \times 10$
18	softmax	$1 \times 1 \times 10$			$1 \times 1 \times 10$

Table 1: An 18-layer convolutional neural net.

Flowchart:

Add chart here

The convolutional neural network, including convolve, Rectifying, pooling, fully connected, and softmax can be seen below:

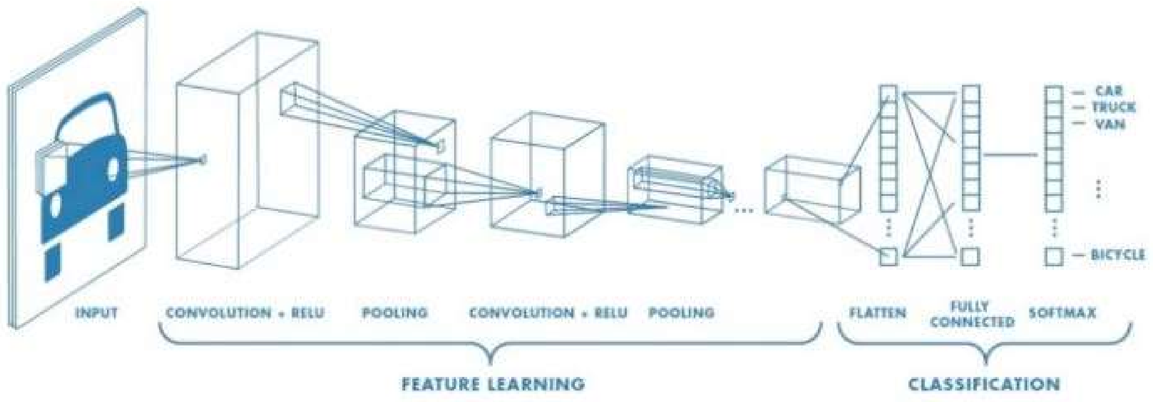

Image Source: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Part 3: Experiment Observations:

We experienced some difficulties integrating all the functions. We ended up with only a 0.14 accuracy score, which is far too low.

Part 4: Team Contributions:

Josue Perez: Full Connect, Max Pool, ReLU, debugging and testing

Aziz Boudy: No contribution outside of week two progress update (did introduction)

Joshua Zapusek: Convolve, Normalization, Main, SoftMax, project report