PROJECT REPORT – CMPEN 454

Project 1 - CNN for object recognition

Group Members: Hongshuo Wang, Zerui Li, Songyang Mao, Jiafu Chen

A. Summarize in your own words what you think the project was about. What were the tasks you performed; what did you expect to achieve?

The convolutional neural net is one of the most popular groups of neural networks that belongs to deep learning. It is created by powerful algorithms that can take an input image and transfer into another understandable image by using various building filters. They have now been widely used in image processing and analysis in areas such as car plate recognition, face identification, document analysis, etc.

This project was divided into different operational blocks while basic algorithm stages that include image normalization, ReLU, maxpool, convolution, fully connected and softmax. Then, putting all these together and implementing an 18-layer CNN to produce 10 object class. Finally, by collecting data from evaluation and passing debugging test, we can easily compare the accuracy of each algorithm and verify the expected results with ours from each layer.

Although we had plenty of time to work on this project, we still faced difficulties as we were developing the project. One of the major concerns we had is that most of our team members do not have coding experiences with Matlab before, so we would have to figure things out from the beginning. However, it was a great learning experience by exploring this powerful software throughout the whole project. It also gives us a new aspect of viewing pixel images and lead us to a new standpoint of image processing. This project not only provides us fundamental understandings of modern computer image processing design, but also enhances our programming skills on designing and implementing images through Matlab.

B. Present an outline of the procedural approach along with a flowchart showing the flow of control and subroutine structure of your Matlab code. Explain any design decisions you had to make. Even though the mathematical specification of each part of this project is fairly strict, there are a lot of different ways you could implement each building block in Matlab, ranging from C-like nested for-loop computations, to cleverly vectorized code. Be sure to document any deviations you made from the above project descriptions (and why), or any additional functionality you added to increase robustness or generality of the approach

Since this project is building a convolutional neural network (CNN) for object recognition in Matlab, our team first read the project requirement together and thought the project could be mainly divide into several parts. First part is the basic operations. In this part, the tasks are divided into six categories, which are Image Normalization, ReLU aka Rectified Linear Unit, Maxpool, Convolution, Fully Connected and Softmax. We found Convolution and Fully Connected are the most challenging part of this project. Since there are four members in our group, the first four tasks are assigned by two of the members while the last two tasks are assigned to the other two, and we spent the first two weeks working on this. Since most of our team members do not have programming skills with Matlab before, we are learning this new language as the project is being developed. This could become challenging for us since we would have to read the document of MATLAB to familiar with its grammar first. The second major part of this project is putting all the functions together before the deadline. As we finished each task from previous part, we worked to put them all together to do object recognition to implement an 18-layer CNN that takes a 32 by 32 color image as input and produces 10 object class probabilities as output.

Here is an outline of the procedural approach along with a flowchart showing the flow of control and subroutine structure of Matlab code:

1. The Basic Operations

In this part, we developed 6 MATLAB functions as the main components for convolutional neural net: Image Normalization (1.1), ReLU aka Rectified Linear Unit (1.2), Maxpool (1.3), Convolution (1.4), Fully Connected (1.5) and Softmax (1.6).

1.1 Image Normalization

```
function outarray = apply_imnormalize(inarray)
    input = double(inarray);
    out = input/255.0 - 0.5;
    outarray = out;
end
```

In this function, the input is an array given for the information of the image and we entered the equation from the project description, which is Out(i,j,k) = In(i,j,k) / 255.0 − 0.5, and converted it to MATLAB code.

1.2 ReLU

```
function outarray = apply_relu(inarray)
    input = inarray;
    out = max(input,0);
    outarray = out;
end
```

In this function, the input is an array from image information and we entered the equation from the project description again, which is Out(i,j,k) =max(In(i,j,k),0), and converted it to MATLAB code. We used the max() statement in MATLAB to compare the values in input array.

1.3 Maxpool

The equation that project description given is Out(i, j ,k) =max({In(r, c, k) | (2i – 1) < r < 2i and (2j – 1) < c < 2j }); however, MATLAB could not achieve this math equation directly by using statements. So, we tried another way to solve this question. We have r in range (2i - 1, 2i) and c in range (2j - 1, 2j), based on this we wrote a loop in MATLAB to find the max value in the pool by comparing four adjacent positions, and this became very effective. The output will return the maximum value of the image in a 2*2

area to reduce the spatial size of the image to a level; therefore the neural network could be much easier to manage the input image.

```
function outarray = apply_convolve(inarray, filterbank, biasvals)
    in = double(inarray);
    filter_bank = double(filterbank);
    v_bias = double(biasvals);


    d2 = size(filter_bank,4);
    [n,m,d1] = size(in);
    out = zeros(n,m,d2);

    for l=1:d2
        sum = double(zeros(n,m,1));
        for k=1:d1
            t = imfilter(in(:,:,k),filter_bank(:,:,k,l),0,'conv','same');
            sum = sum + t;
        end
        out(:,:,l)=sum + v_bias(l);
    end
    outarray = out;
end
```

1.4 Convolution

```
function outarray = apply_maxpool(inarray)
    %Out(i; j;k) = max(f In(r;c;k) j (2i□1) < r < 2i and (2 j□1) < c < 2 j g)
    %reference: https://www.mathworks.com/matlabcentral/answers/409032-how-do-
    input = inarray;
    [N,M,D] = size(input);

    n = N/2;
    m = M/2;

    out = ones(n, m, D);

    for i = 1:D

        cc = input(:,:,i);

        a11 = cc(1:2:end,1:2:end);
        a12 = cc(1:2:end,2:2:end);
        a21 = cc(2:2:end,1:2:end);
        a22 = cc(2:2:end,2:2:end);



        out(:,:,i) = max(max(a11,a12),max(a21,a22));
    end
    outarray = out;
end
```

As we have the given equation:

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

for computing the l-th channel of the output image using summations of 2D convolution, but MATLAB cannot directly process. So we wrote a loop as K

in range(1, D1) process such Fl * In, and add the bias. Since we have D2 numbers for output, we also need a loop outside the computing loop to add these number together to complete the convolution function.

1.5 Fully Connected

```
function outarray = apply_fullconnect(inarray, filterbank, biasvals)
    in = inarray;
    filter_bank = filterbank;
    v_bias = biasvals;


    out = zeros(1,1,10);
    [n,m,d1,d2] = size(filter_bank);

    for l=1:d2
        t = 0;
        for i=1:n
            for j=1:m
                for k=1:d1
                    t = t + imfilter(in(i,j,k),filter_bank(i,j,k,l),'conv','same');
                end
            end
        end

        out(1,1,l) = t + v_bias(l);
    end
    outarray = out;
end
```

Since there has total D2 numbers of filter banks, so we have a loop as from 1 to D2, since the equation is below:

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

So we have other 3 loops inside the 1 to D2 loop, which are i in range 1-N, j in range 1-M, and k in range 1-D1. Inside the loop, we times the value for Fl*In together and add bias for each one. Then, we add these values together to get the output array, as form of 1 * 1 * D2.

1.6 Softmax

In this function, our main job is to achieve with this equation:

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

where

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

```
function outarray = apply_softmax(inarray)
    in = inarray;
    d = size(in);
    out = zeros(1,1,d(3));
    a = max(in(1,1,:));
    t = sum(exp(in(1,1,:)-a));


    for k=1:d(3)
        out(1,1,k) = exp(in(1,1,k) - a)/t;
    end
    outarray = out;
end
```

To avoid massive code of calculating this, we first used code to represent the value of a and the sum of exp(In(1, 1, k) – a). Then, we used a loop for k in range(1, D) to get the corresponding k for Out(1, 1, k) and return a set of probabilities from 0 to 100% for the prediction as the evaluation neural network for the input array about the image.

2. Putting It All Together

In this part, we have developed 3 MATLAB functions putting 6 functions from previous section together which are: Input Parameters (2.1), Debugging (2.2), Round Debugging (2.3), Debugging Main (2.4), and Combination (2.5):

2.1 Input Parameters

```
%loading this file defines filterbanks and biasvectors
load 'CNNparameters.mat'
%sample code to verify which layers have filters and biases
for d = 1:length(layertypes)
    fprintf('layer %d is of type %s\n',d,layertypes{d});
    filterbank = filterbanks{d};
    if not(isempty(filterbank))
        fprintf(' filterbank size %d x %d x %d x %d\n', ...
            size(filterbank,1),size(filterbank,2), ...
            size(filterbank,3),size(filterbank,4));
        biasvec = biasvectors{d};
        fprintf(' number of biases is %d\n',length(biasvec));
    end
end
```

What we are mainly doing in this function is putting parameters into it to load data set as input of the convolutional neural net for object recognition.

2.2 Debugging

```
% Debugging check for CNN layers
function debugging(filteredImage, D)
    load 'debuggingTest.mat'


    if isequal(filteredImage, layerResults{D})
        fprintf('%i passes\n',D);
    else
        fprintf('%i ERROR\n',D);
    end


end
```

This function is designed to debug our code and help us making sure everything works.

2.3 Round Debugging

```
function round_debugging(filteredImage, D)
    load 'debuggingTest.mat'



    if isequal(round(filteredImage,8), round(layerResults{D},8))
        fprintf('%i passes\n',D);
    else
        fprintf('%i ERROR\n',D);
    end


end
```

This is a little different from function 2.2. This function is specially designed for the debugging of convolution to test and make sure that function would worked.

2.4 Debugging Main

In this function, we debugged the connected 18 layers of the neural networks that including Image Normalization, ReLU aka Rectified Linear Unit, Maxpool, Convolution, Fully Connected and Softmax to test and make sure that the neural network could work

2.5 Combination

In this function we connected three neural network that has six layers each: Image Normalization, ReLU aka Rectified Linear Unit, Maxpool, Convolution, Fully Connected and Softmax.

3. Quantitative Evaluation

In this part, we developed 2 MATLAB functions for quantitative evaluation, which are: Confusion Matrix (3.1), K-plot (3.2):

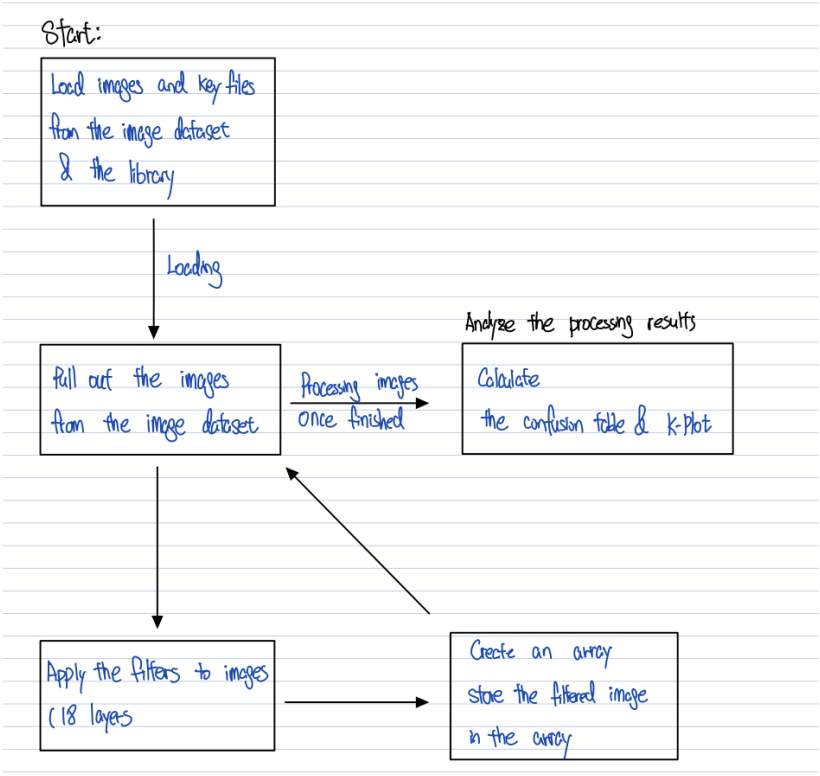
3.1 Confusion Matrix

In this function we created a 10 * 10 table to record the predictor predict object with label to which kinds of classes of the objects for each image in the data set that project is given. The output would be a 10 * 10 matrix that has the specific numbers of the prediction details for each class.

3.2 K-Plot

In this function we used the plot graph to show the classification accuracy with respect to the top-k classes. The input is an array with size (1,1,10), by a double layer loop we can have the max accuracy percentage for each of the situation as 1 to 10.

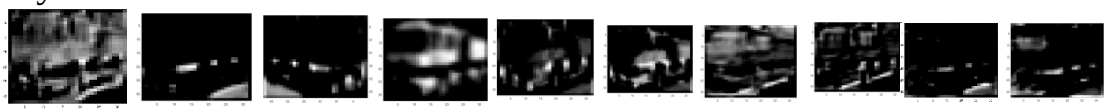
4. Implementation Process



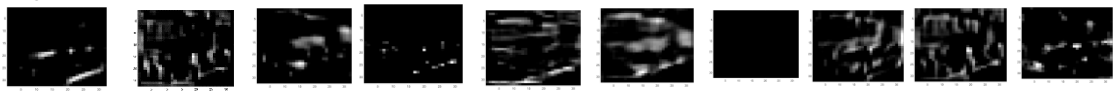
C. Experimental observations. What do you observe about the behavior of your program when you run it? Does it seem to work the way you think it should? Run the different portions of your code and show pictures of intermediate and final results that convince us that the program does what you think it does. Each channel of an intermediate result array in the CNN can be interpreted as a greyscale image. So, for example, the output from layer 2 of the CNN is an array of size 32×32×10, and this could be displayed as 10 small greyscale images, each of size 32×32. CNN practitioners often display these intermediate results and interpret them as images showing the results of learned “feature” detectors. The results from the final softmax layer could be displayed as a bar chart.

Input a truck image

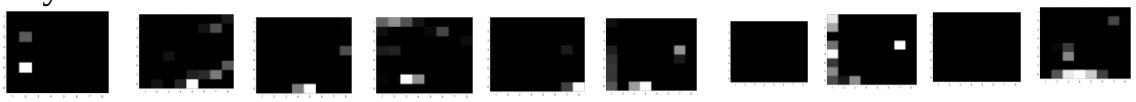
Layer 3:



Layer 5:



Layer 13:



Softmax: airplane 0.0041, automobile 0.7332, bird 0.0011, cat 0.0070, deer 0.0012, dog 0.0021, frog 0.0066, horse 0.0019, ship 0.0128, truck 0.2300

In most cases, the program works as expected. As we randomly picked an image about a truck and wanted to check its greyscale image from the processing of layer 3, 5, and 13, we found that as the layer became deeper, the features would be much lesser so that it would have more weight in the neural network. For the prediction, the result of this image became a little bit embarrassing. It mistakenly recognized a truck as an automobile, we believe this might due to features in this graph are similar with automobile rather than a normal truck, so the neural network get cheated.

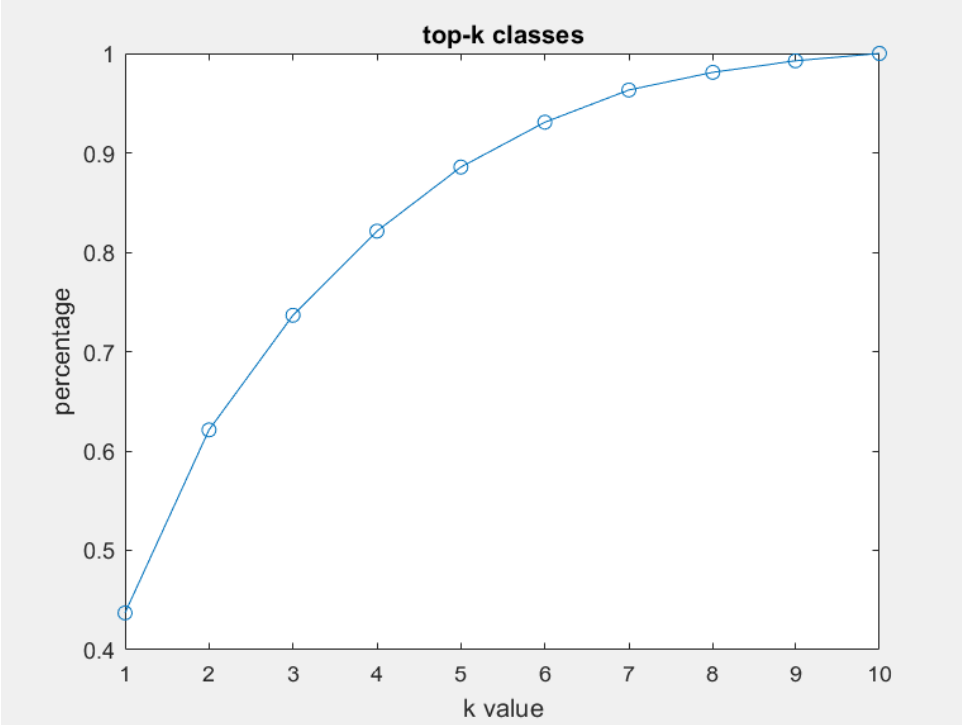
D. Run performance evaluation experiments as described above to compute the confusion matrix and classification rate. Discuss.

Here is the confusion matrix that we get:

	1	2	3	4	5	6	7	8	9	10
1	531	40	87	39	53	19	10	32	192	69
2	41	519	8	18	6	7	7	7	84	191
3	65	9	386	127	270	151	120	73	35	23
4	37	26	117	325	69	222	125	98	44	41
5	10	10	97	45	259	49	93	77	7	4
6	8	7	70	136	38	281	23	94	8	9
7	18	19	104	186	162	111	557	54	10	30
8	38	29	88	89	114	125	33	533	16	68
9	210	111	25	13	22	20	9	13	542	127
10	42	230	18	22	7	15	23	19	62	438

As we can see, most of prediction for the 10 classes from our neural network would have the accuracy that greater than 50% percent; however, as we find that the model didn't have a good prediction result for class 5 and 6, it might be the reason that the program have the weight of deer which take more value as recognized it to bird, and some features for dog may take higher weight for recognized to cat.

Here is the K-plot for the classification rate that we get:



We have the accuracy for K is 1 to 10: 0.4371, 0.6212, 0.7366, 0.8212, 0.8857, 0.9308, 0.9632, 0.9810, 0.9926, 1.0000.

From the plot, we can have 43.71% as accuracy for just the usual classification accuracy. However, as the plotted value for k = 2, which is the percentage of times that the correct class for an image is one of the 2 classes that have the highest computed probability scores for that image, we have the accuracy increased to 62.12%. As k = 4 the accuracy would higher than 80% and k=6 it has more than 90% for the prediction accuracy.

E. If you are in an exploratory mood, find some images of your own on the web and input them to the CNN to see how well it does on them. To do this, you will need to reduce the image size down to a 32×32×3 color thumbnail image. How would you do this in Matlab? (hint: we discussed generating thumbnail images in one of the lectures). Of course, you will get best results if the images you choose actually contain one of the object classes, and that object should pretty much dominate the image. What happens if you give it an image containing an object that it doesn't know about? For example, what happens if you input an image of your face? (no matter what the output classification for your face is, I'm sure it will be hilarious). Can you think of a test that you could perform on the output probabilities to extend the classification to include an "unknown" category.

Input airplane:

Output softmax: airplane 0.4665, automobile 0.0233, bird 0.0641, cat 0.1020, deer 0.0358, dog 0.0197, frog 0.0291, horse 0.0715, ship 0.0622, truck 0.1258

For the graph similar with the trained dataset, the model would be accuracy for predicting it.

Input Uncle Sam:

Output SoftMax: airplane 6.3443e-04, automobile 0.0027, bird 0.0068, cat 0.4403, deer 0.0063, dog 0.1911, frog 0.2415, horse 0.0382, ship 5.1766e-04, truck 0.0720

For the graph classification that is not similar with the trained dataset, the model would first find its largest weight as in the procession from the neural network and try to classify it to one of its trained classification.

F. Document what each team member did to contribute to the project. It is OK if you divide up the labor into different tasks (actually it is expected), and it is OK if not everyone contributes precisely equal amounts of time/effort on each project. However, if two people did all the work because the third team member could not be contacted until the day before the project was due, this is where you get to tell us about it, so the grading can reflect the inequity.

Hongshuo Wang: Image Normalization layer, ReLU layer, Putting all together, debugging

Zerui Li: Fully Connected layer, K-Plot, project report part B, D, and E

Songyang Mao: Softmax layer, Video presentation, Confusion Matrix

Jiafu Chen: Maxpool layer, Convolution layer, project report part A, C