Computer Vision CAP 5415 – Fall 2017 – Programming Assignment II

Girish Kumar Kannan – UCF ID 4196719 – University of Central Florida

Question 1 – Optical Flow

1. Lucas-Kanade optical flow estimation for flow field and flow vectors were implemented on the given two images – basketball and grove.
2. Lucas-Kanade optical flow estimation for flow field and flow vectors using multi-resolution Gaussian Pyramid framework were implemented on the given two images – basketball and grove.
3. Comparison – Variables shared with both steps – window size of 20, a 5x5 Gaussian filter with Sigma (σ) = 0.5 and algorithm correction parameter.

   Window size – more the size, points are less close to image edge; less the size, probable chances of points close to the image edge increases.

   Sigma – lesser value gives less blur, higher value gives more blur. Thus, a middle value of 0.5 is chosen. If high value is supplied, the Gaussian pyramid pixelates more, which in-turn drops down the efficiency of calculating the optical flow field and vectors.

   Algorithm correction parameter – just +/- 1 which multiplies the value with 'v' of u and v supplied to quiver diagram to obtain the optical flow plots. Reason for doing this – The algorithm designed, I believe, gives the correct direction of vectors and field in the image basketball but fails to give the correct direction in case of the grove images. I believe that the algorithm must be tweaked for every type of image/situation, which makes one algorithm unusable for another set of similar images. Thus, to change the direction of the grove's output, this correction factor is introduced. The output images before and after correction is included in the attachment.

   Basic Observations :
   Basketball – guy in the right – moves in the top right direction.
   Correction factor supplied = 1
   Grove – the plant – moves in bottom right direction
   Correction factor supplied = -1

   Inferences :
   The first method (without pyramid) takes less time than the second method (with pyramid) for 4 subsampling steps.
   The second method is more efficient than the first method as the second method identifies large pixel motions while keeping the size of the window small. This is the advantage of using Gaussian Pyramidal subsampling.

Best choices of comparison :
Basketball – The second down sampling (loop 2) in pyramid gives the best vectors and the first down sampling (loop 1) in pyramid gives the best field in my opinion.
Grove (corrected) – The third down sampling (loop 3) in pyramid gives the best vectors and the second down sampling (loop 2) in pyramid gives the best field in my opinion.

It can be said that as the pyramid increases, blur and features decreases for the same window. Thus, the error in calculation of field and vectors increases with down sampling. Therefore, for a high feature image, a pyramid of 2 to 3 levels would yield satisfactory results while for less feature image, a pyramid of 1 to 2 levels would yield satisfactory results.

Question 2 – Convolutional Neural Network

1. A model with one fully connected layer of 100 neurons with Sigmoid activation function was coded, and was trained and tested with Stochastic Gradient Descent with learning rate of 0.1 for 60 epochs with mini-batch size of 10 and no regularization. An accuracy of 97.14% was obtained.
2. A model with two convolutional layers of 20 kernels with pooling layer pooled over 2x2 regions with 2 strides for each convolutional layer along with one fully connected layer of 100 neurons with Sigmoid activation function was coded, and was trained and tested with Stochastic Gradient Descent with learning rate of 0.1 for 60 epochs with mini-batch size of 10 and no regularization. An accuracy of 98.76% was obtained.
   40 Kernels with 1 stride was not implemented as it did not yield good results (in my computer configuration) and the model failed to produce required accuracy level.
3. A model with two convolutional layers of 20 kernels with pooling layer pooled over 2x2 regions with 2 strides for each convolutional layer along with one fully connected layer of 100 neurons with ReLu activation function was coded, and was trained and tested with Stochastic Gradient Descent with learning rate of 0.1 for 60 epochs with mini-batch size of 10 and no regularization. An accuracy of 99.22% was obtained.
   40 Kernels with 1 stride was not implemented as it did not yield good results (in my computer configuration) and the model failed to produce required accuracy level.
4. A model with two convolutional layers of 20 kernels with pooling layer pooled over 2x2 regions with 2 strides for each convolutional layer along with two fully connected layers of 100 neurons with ReLu activation function was coded, and was trained and tested with Stochastic Gradient Descent with learning rate of 0.1 for 60 epochs with mini-batch size of 10 and L2 regularization with decay of 0.1. An accuracy of 98.04% was obtained.
   40 Kernels with 1 stride was not implemented as it did not yield good results (in my computer configuration) and the model failed to produce required accuracy level.
5. A model with two convolutional layers of 20 kernels with pooling layer pooled over 2x2 regions with 2 strides for each convolutional layer along with two fully connected layers of 1000 neurons with ReLu activation function and a Dropout layer with rate of 0.5 was

coded, and was trained and tested with Stochastic Gradient Descent with learning rate of 0.1 for 40 epochs with mini-batch size of 10 and L2 regularization. An accuracy of around ~10% was obtained.

40 Kernels with 1 stride was not implemented as it did not yield good results (in my computer configuration) and the model failed to produce required accuracy level.

Overall reasoning :

Convolutional layer with 40 kernels and a pooling layer for 1 stride with 2x2 pool size did not yield best results, probably due to my computer configuration. Thus, after playing around with the values for a very long while trying out several number combinations, I fixed the kernel value with 20 and strides of 2. I observed that if the kernel size increases or if stride is 1 or both, the accuracy drops severely. On the other hand, with kernel size of 10 and strides of 2, the accuracy is highly satisfactory but due to less filters/kernels, I had to choose 20 which did not fail the model badly. Time was also a factor – higher kernel size and least stride of 1, takes about 100 seconds per epoch in my GPU configuration and it would not compute well.

I also had to split the training and test set into batches since my GPU would not handle a tensor of 10,000 x 28 x 28 x 1 – low memory issues. I used 1000 items in a batch.

Special Case : Model 5 did not yield good results. After 12 or 15 epochs, no matter what the regularization is used (L2 or dropout or both or whatever), would yield an accuracy of ~10%. I do not understand what is causing the issue as the Model 4 is same as model 5, just without dropouts yielding 98% accuracy. Now for dropouts, for whatever value supplied, ranging from 0.01 to 0.99 inclusive of 0.1's and 0.9, would give the same accuracy at the end. However, I could make one observation, lesser the dropout value, longer it takes to fail (reaches 10% in epochs – around 20^{th} epoch), while higher the dropout rate, sooner it failed (reaches 10% in epochs – around 10^{th} epoch). As a proof of several failures, some of the Model 5 results are attached along.

(Special Note : Asked professor regarding the same, advised me to try different regularization techniques. I tried several of them and the model still fails for the described model specifications given in the PA2 question paper.)

I was able to achieve closer accuracies than what was needed as mentioned in the testCNN.py for every model; however, model 5 was a huge disaster.

The programs written are not excessively yet sufficiently commented. All files provided and procured from internet that was used as image data are attached in the final submission file.