ML TASK REPORT

Semantic Segmentation

JANUARY 31, 2015

AMAN RAJ

Delhi Technological University (formerly DCE)

New Delhi, India

Contents

1. Introduction to Problem

    1.1 Semantic Segmentation

    1.2 Related Work

2. Working Models and Findings

    2.1 Pixel Model

    2.2 Feature Model

    2.3 Integrated Channel Model

3. Some Results

4. References

1. Introduction to Problem

Image segmentation problem involves finding an efficient algorithm that can partition the image into various segments such that pixels belonging to a particular segment have some common characteristic or computed property such as color, intensity, or texture and hence are able to convey a distinguish information. The goal of segmentation is to transform the image representation into a form such that it becomes more meaningful and easier to analyze for the machines. Image segmentation is highly useful for object detection tasks and boundaries detection in an image.

1.1 Semantic Segmentation

Semantic segmentation is a very precise form of image segmentation process which requires pixel wise classification of images and is greatly helpful in challenging tasks such as remote sensing, driving assistance systems and precise object localization of objects in general. The given task required semantic segmentation of RGB aerial images taken by the on-board camera of an Unmanned Aerial Vehicle (UAV). Each pixel is classified among elven classes as per the following table.

Table No.1

S.No.	Object	Label Assigned
1.	Building	1
2.	Dirt	2
3.	Foliage	3
4.	Grass	4
5.	Human	5
6.	Pole	6
7.	Rails	7
8.	Road	8
9.	Sign	9
10.	Sky	10
11.	Others	0

This semantic segmentation problem pose one of the big challenges in this category as it has eleven classes and pooling features from such images which can well differentiate between pixels having same intensity but belonging to different classes. While doing this classification at pixel level it is very important to look at the neighbourhood of the candidate pixel so that while trying to identify certain pattern all the pixels of that pattern are assigned the same class value.

1.2 Related Work

There have been many works where researchers have proposed different approaches to solve the problem related to semantic segmentation. Björn Fröhlic et all presents in [2], a kernel based method which exploits the histogram intersection kernel for fast and exact Gaussian process classification. Cl ´ement Farabet et al in [3] proposes a multi-scale convolutional network that is trained from raw pixel values to extract dense feature vectors that encode regions of multiple sizes centered on each pixel. Christoph Göring et al in [4] uses a modified form of GrabCut algorithm where they generate multiple class-specific segmentations and classifies them by using shape and colour information. They uses an iterative segmentation and classification approach.

2. Working Models and Findings

2.1 Pixel Model

Since the task is of semantic segmentation, so in my first approach I used a pixel model. For every pixel in the RGB image a model is formed that contains four values namely the red, green, blue values and the corresponding depth value taken from the provided disparity 2-D image. Hence, the input data is of form 1x1 dimension and 4 channels. Here, these four values are being used as a features for a pixel.

A neural network model that contained four inner product layers was made and as supervised learning approach was used to teach the network to classify each pixel from the input image among the eleven possible classes as required as show in Table 1. The major portion of the time in this work here, was spent on preparing the dataset and making a working architecture to meet the goal. The selection of number of layers and different types of neurons was done by studying the learning curve with respect to the accuracy.

The hypotheses was that let's make the network learn about the occurrence of say pixels of building at different position in image using the depth value and RGB values. Development was done using python and neural network model was made using Caffe, a deep learning framework optimized for speed and accuracy. The observed characteristics of this network during the training were as follows:

Table No.2

S.NO.	Attributes	Explanation
1.	Robustness	Since accuracy fluctuates heavily with losses, not a robust approach
2.	Originality	Work is 100% original, 2-D disparity map of the provided image being used here as one of features.
3.	Efficiency	Training of model takes very least time among all and so the execution of this model.
4.	Accuracy	Reaches to maximum value=50%, then falls rapidly to minimum value=13% and repeats
5.	Thoroughness Evaluation	Accuracy, losses, change of learning rate, study of network during training in debug mode, backward computation in the network etc. studied.

Training result of this model:

```
I0105 23:17:44.967088 13079 solver.cpp:315]     Test net output #0: accuracy = 0.192971
I0105 23:17:44.968000 13079 solver.cpp:209] Iteration 550000, loss = 0.903284
I0105 23:17:44.968019 13079 solver.cpp:445] Iteration 550000, lr = 0.000488494
I0105 23:17:49.197477 13079 solver.cpp:209] Iteration 555000, loss = 0.325797
I0105 23:17:49.197512 13079 solver.cpp:445] Iteration 555000, lr = 0.000485248
I0105 23:17:53.418231 13079 solver.cpp:264] Iteration 560000, Testing net (#0)
I0105 23:18:05.268615 13079 solver.cpp:315]     Test net output #0: accuracy = 0.425882
I0105 23:18:05.269558 13079 solver.cpp:209] Iteration 560000, loss = 0.663334
I0105 23:18:05.269579 13079 solver.cpp:445] Iteration 560000, lr = 0.000482052
I0105 23:18:09.491011 13079 solver.cpp:209] Iteration 565000, loss = 0.249858
I0105 23:18:09.491047 13079 solver.cpp:445] Iteration 565000, lr = 0.000478905
I0105 23:18:13.695220 13079 solver.cpp:264] Iteration 570000, Testing net (#0)
I0105 23:18:25.451074 13079 solver.cpp:315]     Test net output #0: accuracy = 0.194202
I0105 23:18:25.452033 13079 solver.cpp:209] Iteration 570000, loss = 0.0450553
I0105 23:18:25.452052 13079 solver.cpp:445] Iteration 570000, lr = 0.000475805
I0105 23:18:29.690601 13079 solver.cpp:209] Iteration 575000, loss = 0.121298
I0105 23:18:29.690639 13079 solver.cpp:445] Iteration 575000, lr = 0.000472752
I0105 23:18:33.928601 13079 solver.cpp:264] Iteration 580000, Testing net (#0)
I0105 23:18:45.732103 13079 solver.cpp:315]     Test net output #0: accuracy = 0.45854
I0105 23:18:45.733147 13079 solver.cpp:209] Iteration 580000, loss = 2.77404
I0105 23:18:45.733168 13079 solver.cpp:445] Iteration 580000, lr = 0.000469744
I0105 23:18:49.934759 13079 solver.cpp:209] Iteration 585000, loss = 0.0283198
I0105 23:18:49.934792 13079 solver.cpp:445] Iteration 585000, lr = 0.00046678
I0105 23:18:54.163251 13079 solver.cpp:264] Iteration 590000, Testing net (#0)
```

2.2 Feature Model

Since, in previous approach the neighbourhood of each pixel wasn't taken care of while deciding the features of that particular pixel, so intended features weren't much robust and very much prone to illumination since intensity values are being used. The second approach as explained in this section was an improvement over the previous. It considers taking a pixel with its some neighbourhood i.e. an image patch and calculate a feature model for each such patches. The algorithm can be defined as follows:

- Many 16x16 non-intersecting patches from the RGB images are formed, for each patch HOG features and Local binary Pattern (lbp) features are extracted which results in one dimensional 1x36 HOG feature vector and a 1x36 lbp feature vector. Now, a feature-model is formed by arranging these vectors into 6x6 2-D feature vector having two channels, channel=1 (corresponds to HOG features) and channel=2(corresponds to lbp features).

- The corresponding 16x16 image patch from the labelled images are taken, and the label of centre pixel of this 2-D matrix is assigned as the label to the current feature model of the 16x16 image patch. This label can be anything from 0 to 10 as per the class to which it belongs.

- The idea is to form a robust feature model of each pixel that encompasses the information about the neighbourhood that can help in teaching the network in classifying the pixels in different classes.

- By using this methodology training and testing data was prepared. A Convolutional Neural Network (CNN) was made which takes these 6x6 2-D feature of each 16x16 patch with the corresponding label of this patch. The network was designed to pool secondary features from such feature models. The network architecture and obtained result attributes are as follows:

Table No.3

S.NO.	Attributes	Explanation
1.	Robustness	System more robust than previous technique as heavy fluctuation in accuracy has decreased.
2.	Originality	Some third party code used to generate HOG and LBP features, rest the codes written in python is 100% original.
3.	Efficiency	Training of this model takes more time than the previous approach.
4.	Accuracy	Maximum Value=30% and remains above 25% all the time, an improvement than before.
5.	Thoroughness Evaluation	Accuracy, losses, change of learning rate, study of network during training in debug mode, backward computation in the network etc. studied along with the output feature maps of convolutional layers.

Training result of this model:

```
I0114 18:16:22.589814 24180 solver.cpp:315]     Test net output #0: accuracy = 0.2688
I0114 18:16:22.590268 24180 solver.cpp:209] Iteration 10000, loss = 2.33453
I0114 18:16:22.590288 24180 solver.cpp:445] Iteration 10000, lr = 0.000594604
I0114 18:16:22.781023 24180 solver.cpp:209] Iteration 10500, loss = 2.1756
I0114 18:16:22.781059 24180 solver.cpp:445] Iteration 10500, lr = 0.000583693
I0114 18:16:22.975044 24180 solver.cpp:264] Iteration 11000, Testing net (#0)
I0114 18:16:23.723137 24180 solver.cpp:315]     Test net output #0: accuracy = 0.28479
I0114 18:16:23.723567 24180 solver.cpp:209] Iteration 11000, loss = 2.19593
I0114 18:16:23.723585 24180 solver.cpp:445] Iteration 11000, lr = 0.000573239
I0114 18:16:23.913023 24180 solver.cpp:209] Iteration 11500, loss = 1.06185
I0114 18:16:23.913058 24180 solver.cpp:445] Iteration 11500, lr = 0.000563211
I0114 18:16:24.144248 24180 solver.cpp:264] Iteration 12000, Testing net (#0)
I0114 18:16:24.891032 24180 solver.cpp:315]     Test net output #0: accuracy = 0.15906
I0114 18:16:24.891469 24180 solver.cpp:209] Iteration 12000, loss = 1.34338
I0114 18:16:24.891487 24180 solver.cpp:445] Iteration 12000, lr = 0.000553583
I0114 18:16:25.081243 24180 solver.cpp:209] Iteration 12500, loss = 1.962
I0114 18:16:25.081280 24180 solver.cpp:445] Iteration 12500, lr = 0.000544331
I0114 18:16:25.293035 24180 solver.cpp:264] Iteration 13000, Testing net (#0)
I0114 18:16:26.052397 24180 solver.cpp:315]     Test net output #0: accuracy = 0.27233
I0114 18:16:26.052819 24180 solver.cpp:209] Iteration 13000, loss = 2.29349
I0114 18:16:26.052834 24180 solver.cpp:445] Iteration 13000, lr = 0.000535432
I0114 18:16:26.241901 24180 solver.cpp:209] Iteration 13500, loss = 1.74972
I0114 18:16:26.241933 24180 solver.cpp:445] Iteration 13500, lr = 0.000526865
I0114 18:16:26.431562 24180 solver.cpp:264] Iteration 14000, Testing net (#0)
I0114 18:16:27.164849 24180 solver.cpp:315]     Test net output #0: accuracy = 0.26764
```

2.3 Integrated Channel Model

After exploring the work done in the previous section, I decided to use CNN to pool some primary features for the image patches with aim to increase accuracy and robustness in the system. This work was inspired by Piotr dollar's work [1], where he extracts some integral channel features from images. Piotr has shown the combination of Gradient, LUV and Gradient Histogram channel of an image to extract first and second order features, as an excellent candidate to increase the accuracy of AdaBoost learning algorithm among other possible channels combinations. The current approach can be presented as follows:

- Image patches of size 28x28 from the RGB images are formed, this time patch size is increased to increase the neighbourhood information in the patch being processed. Before that for each image it's following forms are calculated:
  - LUV colour channels of image
    To capture colour of different pixels.
  - Local Binary Pattern image
    To detect textures in images like flat surface etc.
  - Image gradient using Laplacian filter of kernel size
    To detect various edges information.
  - Histogram of Gradient image (HOG)

Laplacian kernel

$$\text{kernel} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\Delta \text{src} = \frac{\partial^2 \text{src}}{\partial x^2} + \frac{\partial^2 \text{src}}{\partial y^2},$$

- The above aspects of image are called as integral channels here namely LUV channel, LBP channel, Gradient channel and HOG channel. Now, 28x28 corresponding patches from each of these image channels are formed. Hence, for each 28x28 RGB image patch in the image we have obtained a corresponding patch of same size having total six channels (L, U, V, LBP, Gradient, HOG). This is called Integrated Channel Patch Model (ICPM). The label of each such patch is formed as in previous section, by taking the corresponding patch in the labelled image into consideration and using the centre value.

- Now, a Supervised Learning technique is used where a CNN is formed with an aim to pool the primary features from these six forms of a patch so that with its assigned label network can be trained to classify them among our eleven classes. Network architecture and obtained result attributes are shown below.

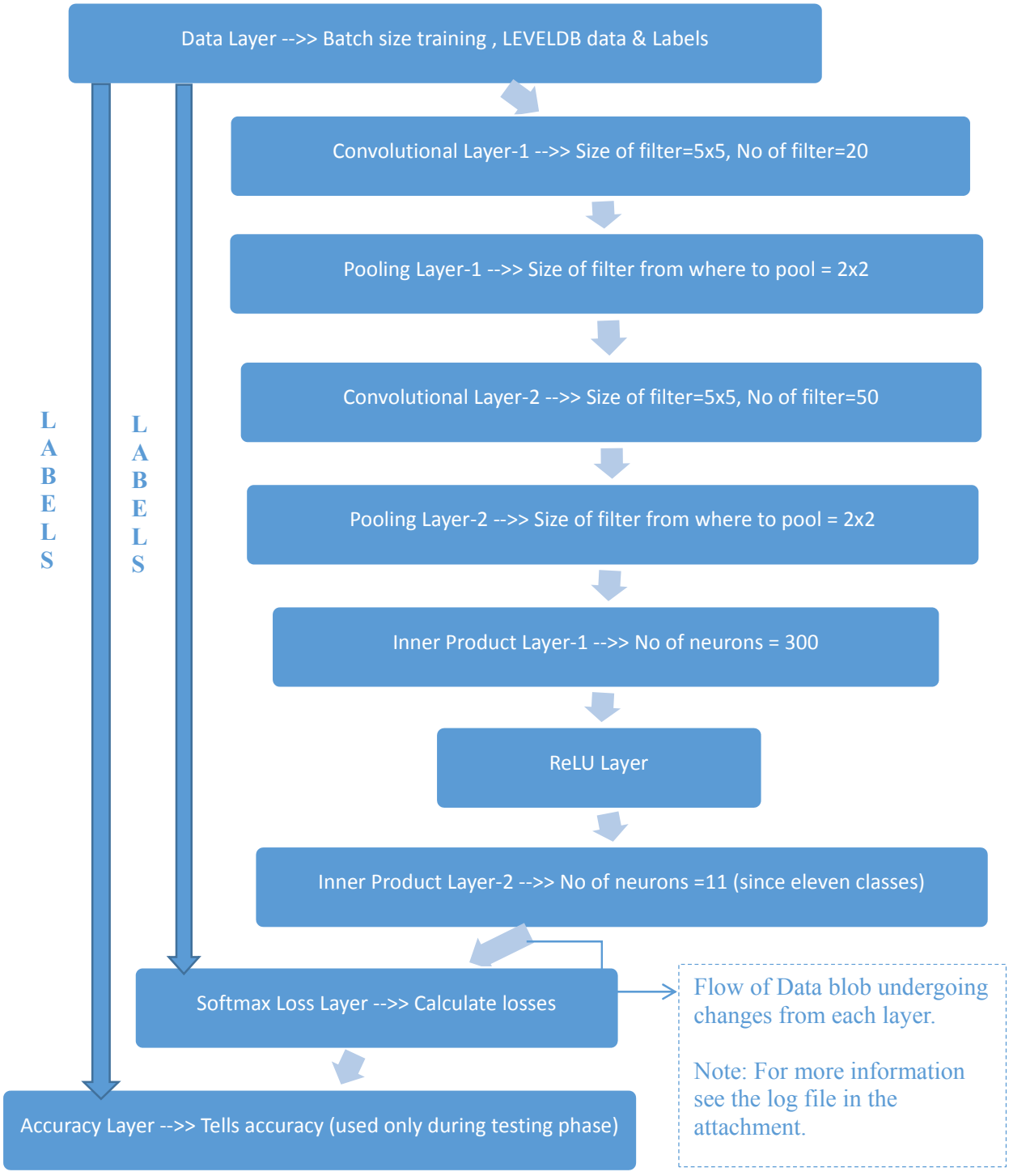
Table No. 4

S.NO.	Attributes	Explanation
1.	Robustness	System's robustness is more or comparable to previous case, with no heavy fluctuation in accuracy and invariant to change in pixel intensity since better representation of image are taken in form of six channel to extract features.
2.	Originality	Third party code is used to get HOG image, rest code is 100% original, e.g. wrote own code to get LBP image as per its original research paper. All codes in python.
3.	Efficiency	Model training takes maximum time out of all and so maximum utilization of available resources as the size of data blob is 6x28x28.
4.	Accuracy	Maximum accuracy=30% and most of time it's close to this value.
5.	Thoroughness Evaluation	Accuracy, losses, change of learning rate, study of network during training in debug mode, backward computation in the network etc. studied along with the output feature maps of convolutional layers.

Training result of this model:

```
I0131 21:36:06.999136  6277 solver.cpp:315]     Test net output #0: accuracy = 0.2872
I0131 21:36:07.040307  6277 solver.cpp:209] Iteration 1500, loss = 2.28662
I0131 21:36:07.040350  6277 solver.cpp:445] Iteration 1500, lr = 0.000900485
I0131 21:36:28.104363  6277 solver.cpp:264] Iteration 2000, Testing net (#0)
I0131 21:38:41.589251  6277 solver.cpp:315]     Test net output #0: accuracy = 0.175613
I0131 21:38:41.631285  6277 solver.cpp:209] Iteration 2000, loss = 2.16079
I0131 21:38:41.631326  6277 solver.cpp:445] Iteration 2000, lr = 0.000872196
I0131 21:39:02.713466  6277 solver.cpp:264] Iteration 2500, Testing net (#0)
I0131 21:41:16.165653  6277 solver.cpp:315]     Test net output #0: accuracy = 0.277927
I0131 21:41:16.207975  6277 solver.cpp:209] Iteration 2500, loss = 1.9722
I0131 21:41:16.208020  6277 solver.cpp:445] Iteration 2500, lr = 0.000845897
I0131 21:41:37.290052  6277 solver.cpp:264] Iteration 3000, Testing net (#0)
I0131 21:43:50.658323  6277 solver.cpp:315]     Test net output #0: accuracy = 0.17528
I0131 21:43:50.701203  6277 solver.cpp:209] Iteration 3000, loss = 2.08596
I0131 21:43:50.701247  6277 solver.cpp:445] Iteration 3000, lr = 0.000821377
I0131 21:44:11.808259  6277 solver.cpp:264] Iteration 3500, Testing net (#0)
I0131 21:46:30.378854  6277 solver.cpp:315]     Test net output #0: accuracy = 0.281987
```

Convolutional Neural Network used for the current task, which I hope to improve further:



3. Some Results

For example: for the below RGB, its labelled image is shown along with the six channels extracted as proposed above.

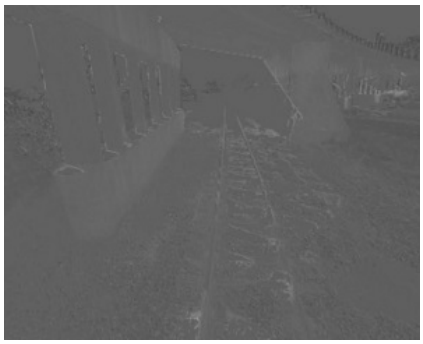
RGB Image

Labelled Image

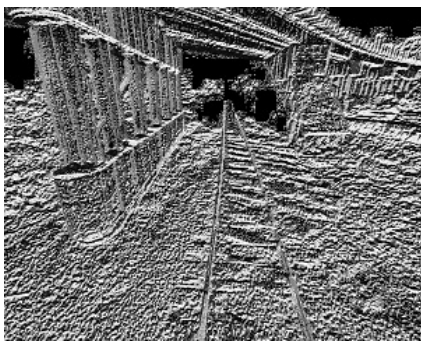
Histogram of Gradient (HOG)

L Channel

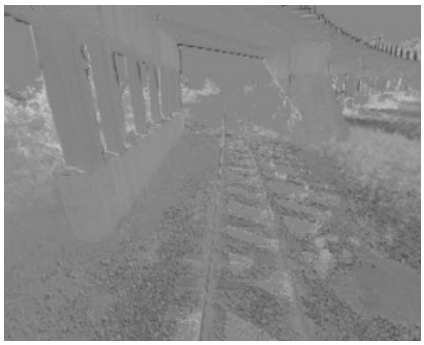
Gradient of Image using Laplacian filter



U Channel

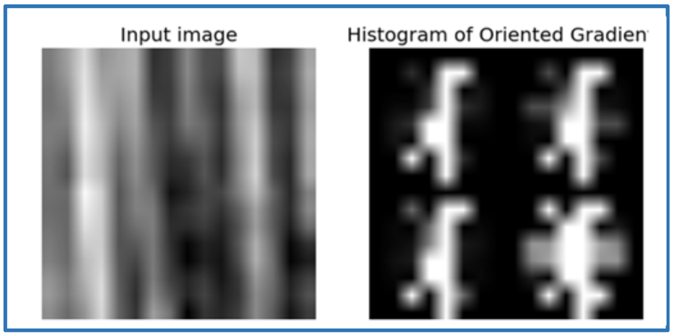


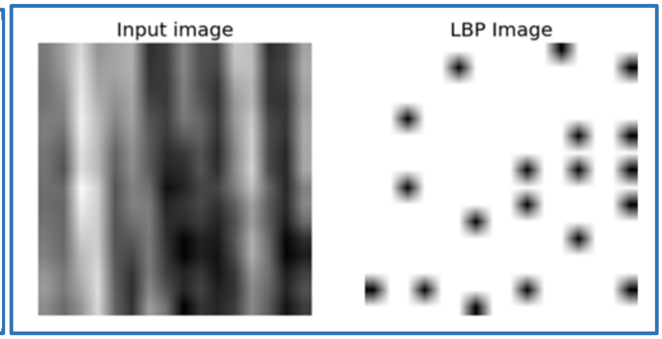
Local binary patter (LBP) image



V Channel

HOG and LBP image for a patch of size 28x28 can be shown as (scaled for better view):





Different class prediction after the network has been drained with the probabilities values:

```
prediction shape: (11,)
prediction class: 2
prediction: [ 0.13497825  0.02969771  0.21310911  0.14020696  0.09372822  0.0021613
  0.0040826   0.08128796  0.16149107  0.00732704  0.13192979]
shape (6, 28, 28)
D_new shape: (28, 28, 6)
prediction shape: (11,)
prediction class: 2
prediction: [ 0.13497825  0.02969771  0.21310911  0.14020696  0.09372822  0.0021613
  0.0040826   0.08128796  0.16149107  0.00732704  0.13192979]
shape (6, 28, 28)
D_new shape: (28, 28, 6)
prediction shape: (11,)
prediction class: 2
prediction: [ 0.13497825  0.02969771  0.21310911  0.14020696  0.09372822  0.0021613
  0.0040826   0.08128796  0.16149107  0.00732704  0.13192979]
shape (6, 28, 28)
D_new shape: (28, 28, 6)
prediction shape: (11,)
prediction class: 2
prediction: [ 0.13497825  0.02969771  0.21310911  0.14020696  0.09372822  0.0021613
```

4. References

1. Piotr Dollár, Zhuowen Tu, Pietro Perona, Serge Belongie. Integral Channel Features, 2009.

2. Björn Fröhlich and Erik Rodner and Joachim Denzler. As Time Goes By - Anytime Semantic Segmentation with Iterative Context Forests. Symposium of the German Association for Pattern Recognition (DAGM), 2012

3. Cl ´ement Farabet, Camille Couprie, Laurent Najman, Yann LeCun. Learning Hierarchical Features for Scene Labeling, 1998.

4. Christoph Göring and Björn Fröhlich and Joachim Denzler. Semantic Segmentation using GrabCut. International Conference on Computer Vision Theory and Applications (VISAPP), 2012.

5. Semantic Segmentation with Deep Learning Michael Cogswell and Dhruv Batra, Virginia Tech, Blacksburg, VA.

6. Rich feature hierarchies for accurate object detection and semantic segmentation Tech report (v5) Ross Girshick Jeff Donahue Trevor Darrell Jitendra Malik, UC Berkeley.

Note: For more details about the current CNN network model that I am working on to improve further and the codes developed during this can be found in attached zip file or check my GitHub repository.