Deep Learning
Prof. Prabir Kumar Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture – 55
Semantic Segmentation - III

Hello welcome to the NPTEL online certification course on Deep Learning, for last few classes we are talking about the applications of the convolutional neural networks which we have done earlier. And, the particular application that we are discussing for previous two class is the Semantic Segmentation problem, that is the semantic segmentation of the input image input to your deep neural network.

(Refer Slide Time: 00:57)

And for semantic segmentation we have discussed about two different architectures, one of the architecture that you have discussed is a fully convolutional neural network and the other architecture that we have discussed is the deconvolutional neural network.

So, what we have seen in case of fully convolutional neural network is that it is a short of reinterpretation of a deep convolutional neural network, where deep CNN is actually designed for classification purpose; that means, we want to classify an input image whether an input image contains a dog whether or whether an input image contains a car or things like that.

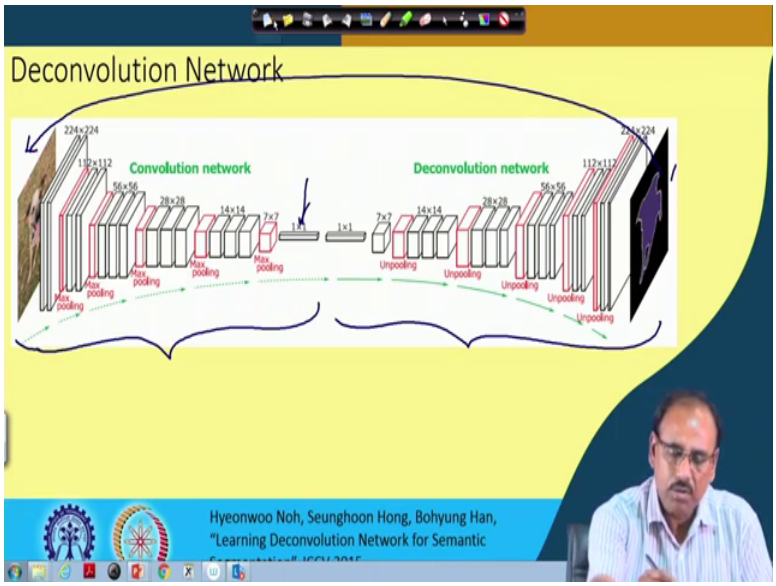
So, for that purpose in the deep convolutional neural network, your initial few layers or the layers in the shallower part of the network which are actually the convolutional layers. There we have the different convolutional layers, then pooling layers which comes one after another, but the final layers in the deep layers few one or more layers which are actually the fully connected layers.

The fully connected layers as we have discussed earlier is something similar to the multi layer perceptron and as the multilayer perceptron is able to classify the input objects or input features. So, the convolutional neural network having fully connected layers at the output side, they are able to classify the input objects or input images to one of the classes to which for which the neural network was trained so, that is what was your fully convolutional neural network.

And the other one was deconvolutional neural network, in deconvolutional neural network what we have seen is the architecture consists of a symmetrical deconvolution part which is mirror symmetry of the convolutional network. Now, in both these cases what you need is your output size have to be same as or the image that is generated at the output or the map that is generated at the output has to be same as the input image size.

Now, in today's lecture what we are going to discuss about is how to train those fully convolutional neural networks or even the deconvolution neural network. So, that those networks are capable of performing semantic segmentation tasks and for this we will talk about two types of loss functions which are to be minimized during training of these networks. One of the loss function that we will talk about is cross entropy loss and the other loss function that we will talk about is dice loss.

(Refer Slide Time: 04:17)



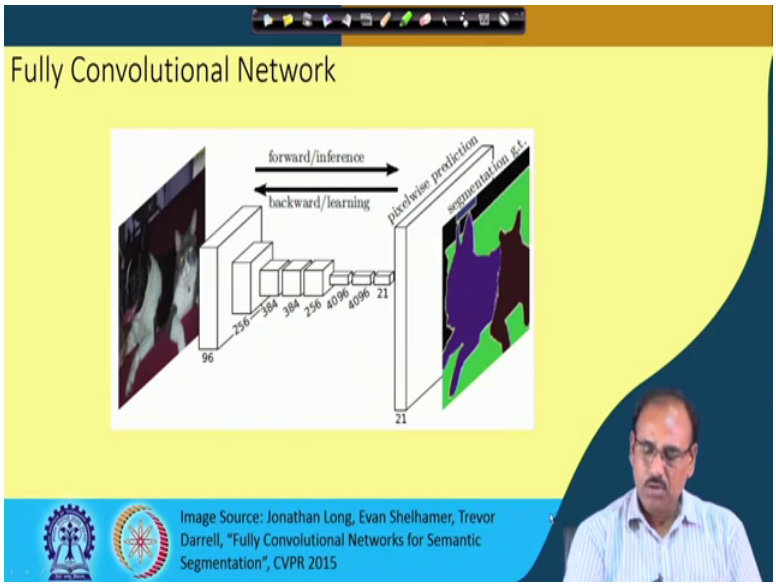
So, you see here that the two architectures that we have discussed in our previous classes as we said that one is the deconvolution network. So, you see in deconvolution network your input side we have a convolutional neural network, where what does this convolution neural network do is it through the convolution operation and the max pooling operation it tries to find out the different features of the input image and the features are extracted at different levels of abstraction. So, as you move inside the deeper layer you have the features extracted at different levels of abstraction.

And, then in the convolutional there it is that deconvolution part along with the unpooling part. So, what it does is in case of convolutional part as well as the pooling part the size of the feature maps goes on reducing because of the max pooling operation which is usually employed and in every layer the size of the feature map goes on reducing one after another. So, as a result say somewhere deep inside the network your size of the feature map that you obtain which is much less than the size of the input image.

So, this deconvolutional network tries to undo this size deduction part. So, there we have an unpooling operation followed by the deconvolution operation using the deconvolution kernel. So, this unpooling and deconvolution gradually tries to bring back your size of the output which is same as the size of the input image. So, if you compare these two your size of the output that you get over here is same as the size of the input image.

So, this is what you get in case of deconvolution network where the increase of size or the up sampling is done gradually whereas, in case of convolutional neural network or fully convolutional network that we have shown before.

(Refer Slide Time: 06:29)



Here the size expansion is done either in one step or in limited number of steps. So, with this fully convolutional network the different modes of operation that you have seen is one of them is FCN 32 where the size is made 32 times in 1 step, then we had seen FCN 16 where the size is increased in 2 steps and we also have seen FCN 8 where the size is increased in 3 steps.

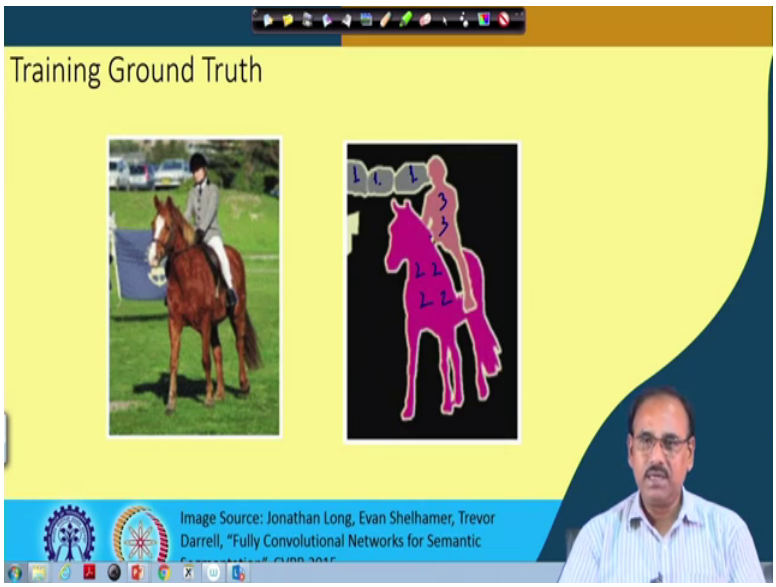
So, by increasing the number of hierarchical steps in which the sizes increased you find that your output has become better and better it was a final output whereas, when the size was increased 32 times in 1 step in that case your output of the segmentation output was very coarse. But whatever we do whether I use deconvolutional network for performing this task or I use a convolutional network for performing this task in both the cases I need an output let us call it an output image whose size is same as the size of the input image or the number of elements in the output array has to be same as the number of pixels in the input image.

And in case of segmentation or semantic segmentation what is what I want is that each of the elements in the output array which corresponds to the corresponding element or corresponding pixel in the input image this output element has to have a particular level

which is associated with the segment to which the input pixel should belong. And using that I have to train the network so, that once the network is trained whether it is deconvolution network or fully convolutional neural network once the network is trained then given an input image the network should be able to perform the segmentation operation of the input image or in other words every pixel of the input image will be classified to one of the categories of objects for which the network has been trained.

So, in case of convolutional neural network it was the classification of the input image the image as a whole was classified to one of the known classes. In case of semantic segmentation what we want is every pixel in the image has to be classified into one of the known classes or this is nothing, but the classes of objects which are present within the image. So, for this classification purpose what I need is; obviously, I need the ground truth images. So, how the ground truth image looks like let us take this particular example.

(Refer Slide Time: 09:27)



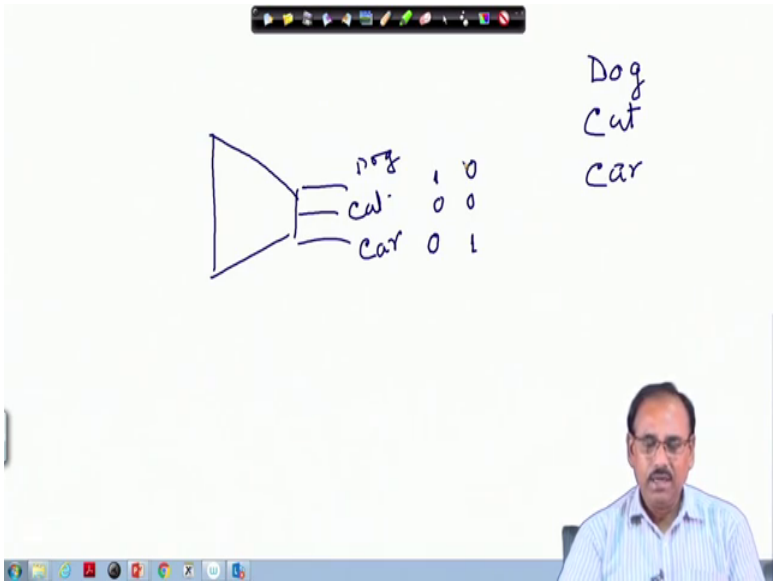
So, here you find that on the left hand side we have original image and on the right hand side this image that we have this is the segmented ground truth image. So, what this segmentation has done is it has identified the pixels belonging to 3 different classes of objects. So, 3 different classes which are considered is the pixels which belong to cars, the pixels which belong to horse and the pixels which belong to a man. So, you find that these are the pixels over here which belongs to cars as shown in this original image,

these are the pixels which belongs to horse and these are the pixels of course, enclosed within this boundary having the same color. So, the all these pixels they belong to the man.

So, what the semantic segmentation output should look like is that, say each of the pixels belonging to belonging to the car all these pixels would we would like to take a value say equal to 1. So, all the pixels belonging to car in the segmented output all these pixels should be labelled with say value equal to 1. Similarly we may want that all the pixels or all the elements which corresponds to a horse all these pixels should be labelled with a value equal to 2. Similarly all the pixels which corresponds to man these pixels should be labelled with a value equal to 3.

So, that I know that within my segment segmented output any pixels having level equal to 3 means these pixels belong to the human. Similarly any pixel having a label equal to 1 this particular element belongs to a car in the original scene. So, this is my ground truth, using this ground truth I have to trained the neural network. Now, we find that when the convolutional neural network is used for the classification purpose, then for training for any given input image for which the class is known the output vector that you get is represented as a one hot vector. So, one what is this one hot vector.

(Refer Slide Time: 12:21)



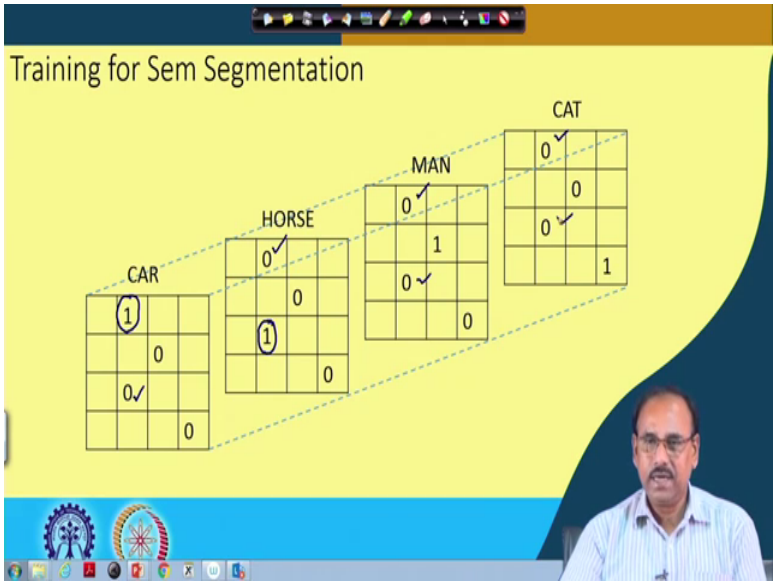
So, the one hot vector is suppose you have 3 different classes say dog, you have a class cat and you have a class say car, say these are the 3 different classes of images 3 different

categories of images that we have. So, this convolutional neural network that we have this neural network will have 3 outputs, one of the output corresponds to dog, the other output corresponds to cat and the other output corresponds to car. So, when I feed an input image which contains a dog then I expect all that my target output should be dog output should be equal to 1 and the rest of the 2 outputs should be 0.

Similarly when I feed an input image containing a car the car output should be equal to 1 and the other 2 output should be equal to 0. So, this is what is and one hot vector. So, for all the training samples depending upon the category of the training sample or the level of the training sample the output is and one hot vector or the target vector. Similarly when I go for segmentation of the an image as the image has got a number of different objects and I want that a pixel should be classified to one of those objects.

So, I want that the every output or every element of the output array has to be represented by a one hot vector depending upon what my ground truth says that whether this element should belong to car or this element should belong to horse or this element should belong to man. So, let us see that how this one hot vector is actually represented.

(Refer Slide Time: 14:25)



So, this is how you represent one hot vector. So, what I have assumed is suppose I have an input image which is of size 5 by 5. So, from the network I also want an output array or say in this case the input image which is say of size 4 by 4. So, from the network I

want an output which will also be of size 4 by 4 and each element of this 4 by 4 array is and one hot vector.

So, I am considering a case where my image is having just 3 categories of objects the categories are car, horse, man and cat. So, if a pixel at location say 0 1 I have a car present at that particular pixel location then my one hot vector should be that for car it will be 1, for horse it will be 0, for man it will be 0, for cat it will be 0. So, that is what has been shown over here. So, the corresponding pixel in my input image that if that contains a car then I have to have this component of the one hot vector to be equal to 1, this component 0, this component 0, this component is also 0.

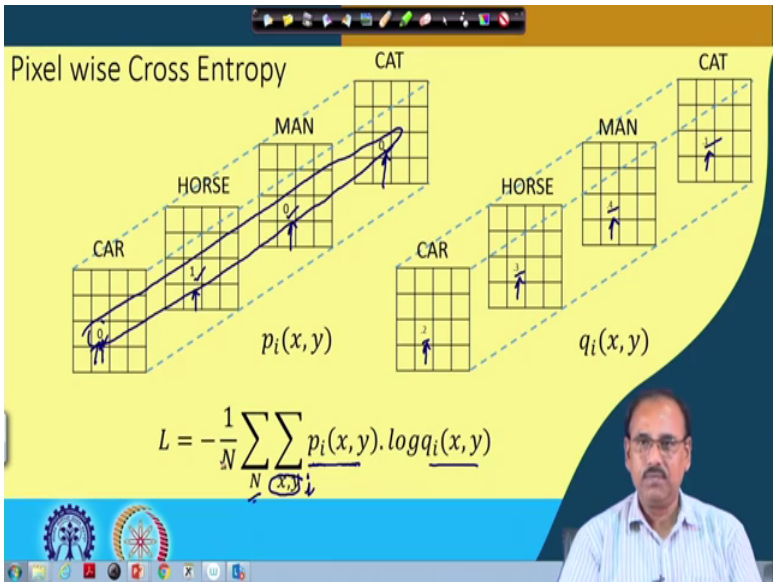
Similarly, at this location at location say 0 1 2 to 1 at that corresponding pixel if a horse is present or if that pixel is part of a horse then in the one hot vector the component corresponding to hot should be equal to 1 and other components corresponding to car, corresponding to man, corresponding into cat all these components it will have to be equal to 0.

So, this one hot vector that every element represents this one hot vector has got 4 different components or it is a 4 dimensional vector, because I am assuming that I will have these 4 categories of object present in my image. So, this is how you generate your training samples that is I have an image and original image for which I have a ground truth and that ground truth is represented in the form of an array of one hot vectors, where the array size is same as your original input image size and every element in that output array is one hot vector corresponding to different classes that are present in my training samples or the training images.

So, this one hot vectors actually become my target vectors now what do I do using this one hot vectors or how do I train the neural network using this one hot vectors, for doing that what I have to do is I have to define a loss function. So, how do you define a loss function, why do I need to define a loss function, for one when you feed one of the training input image you know what is the array of one hot vectors that you would like to have that is your ground truth. But, your network may give you something else, network will also give you an array of vectors, but the contents of those vectors may be different from the from what is your target vector.

So, you compute the difference and this difference actually gives you the loss function or a function of the difference gives you the loss function and then using the back propagation learning algorithm following the gradient descent procedure you try to update the parameters of the network in such a way that the loss that has been computed the loss goes on reducing or I want to have a set of parameters network parameters for the loss for which the loss is minimum. So, I need to define a loss function.

(Refer Slide Time: 18:29)



So, how do I define that loss function? So, here is an example. So, you find that for a particular pixel location say my one hot vector which is the target is and suppose that particular corresponding pixel in the input image is in the region occupied by a horse. So, my one hot vector is 0 as shown over here, the one hot vector is 0 1 0 0, but maybe while training the neural network whether it is the fully convolutional network or a deconvolution network whatever it is.

The network gives me a vector at the corresponding location which is 0.2, 0.3, 0.4 and 0.1 I will tell you just after explaining this that how do you get this 0.2 0.3 0.4 and 0.1 and these are the figures which actually tells you that what is the probability that this pixel belongs to car. So, that probability is 0.2 this pixel belongs to a horse, with probability 0.3 the pixel belongs to man, with point with probability 0.4 and the pixel belongs to cat with point probability 0.1. So, this is the actual vector output which your

network is giving and this is what is my target vector and this is the actual vector that I get.

So, once I have my target vector and the actual vector which is given by the network what I can do is, now I can find out what is the difference between these two or what is the error that has been encountered. So, one of the ways in which I can compute the error is just find out the vector difference and then you take the square of that square of the mode of that vector difference sum it over all the elements within the output array. So, that is what is your sum of squared error loss function which we have talked about earlier and we have also seen that, what is the limitation of that sum of squared error being taken as a loss function.

So, we have discussed earlier that instead of using the sum of squared error as your loss function if you use a cross entropy loss that gives you much more advantage in terms of convergence of your training algorithm. So, here if I compute the cross entropy; so, if I take a pixel location x, y suppose my target value at location x, y is p x, y and the actual value that you get is q i x, y. Then the cross entropy is defined as minus log of p i x, y, q i x, y take the summation over all x, y sorry here it will not be x y take the summation over all I because, we are representing each of the components of this one hot vector as the ith component.

So, if I take p i x, y and x, y represents what is your pixel location. So, this location of the pixel is actually your x, y. So, you take p i x, y then log of q i x, y, where p i x, y is the target value value of the target vector at the in the ith plane or ith object and q i is the actual value that you are getting. So, over here your p 1 x y is equal to 0, here p 2 x y is equal to 1, p 3 x y is equal to 0 and p 4 x y is equal to 0. Similarly here I have q 1 x y which is 0.2, q 2 x, y which is 0.3, q 3 x y which is 0; 4 and q 4 x y which is 0.1.

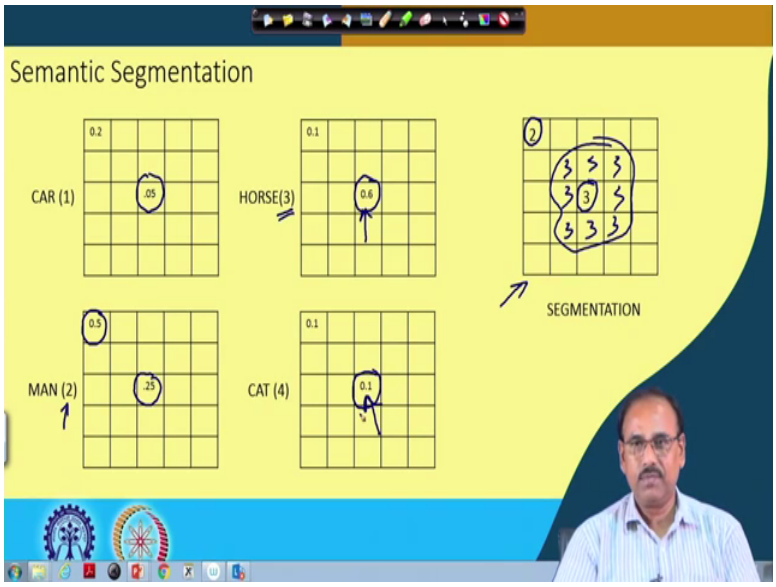
So, using these two the definition says that the cross entropy is simply given by p i x y into q i x y where you take the summation p i x y into log of q i x y and you take the summation over all i. So, that is what is giving you a pixel cross entropy or an element cross entropy and I get the overall cross entropy, if I take the average of this cross entropy over all the elements that I have.

So, what I will do is, I will sum it up over all the elements that I have within this output array if I have total N number of elements. So, this is where I can put that this is over i

and this is over x y and divide it by 1 over N, where N is the total number of elements. So, this is the average cross entropy loss or average pixel wise cross entropy loss that I get and my training algorithm or the gradient descent approach should be that following back propagation gradient descent back propagation you try to update the parameters of your network such that this pixel wise cross entropy loss will be minimized.

So, this is one of the loss functions that can be used for training your neural network whether it is a fully convolutional neural network or it is a deconvolution neural network, this training algorithm applies equally. So, this is one of the loss functions that can be used for training your FCN fully convolution neural network or the deconvolution neural network which will be used for the semantic segmentation purpose. Now, there is another loss function that can also be used for the training purpose.

(Refer Slide Time: 25:01)



That particular loss function before coming to that loss function let us say how do you actually get your segmentation. So, for this semantic segmentation once your network is trained you get an output array where the array size is same as your input image size and every element of the array is a vector where every component of that vector actually tells you that what is the probability that the corresponding input pixel belongs to one of the classes for which the network has been trained.

Say for example, as given over here, if you consider this particular pixel at the center of this array this is a 5 by 5 array assuming that your input image is also of size 5 by 5 and

this is the center pixel in that image there is a center pixel in that image. So, it says that the probability that the center pixel belongs to car is 0.05, the probability that the center pixel belongs to man is 0.25, the probability that the center pixel belongs to horse is 0.6 and the probability that the center pixel belongs to cat is 0.1. And then for semantic segmentation when I want to get the segmented output what I need to do is, I need to give a level to this center element in my output array.

And, here you find that because the probability of occurrence to horse is maximum which is 0.6 and assuming that this horse has an index 3 in my one hot vector in the vector representation of the output for that particular element. Then to this corresponding location I will assign a label 3, similarly in the other case say this pixel the probability of occurrence to man is maximum and assuming that man has an index 2 in that one hot vector representation of this output then at this location the level assigned will be 2.

So, this final output array or the segmented output array is an array of integers where a number at any location tells you to which of the classes that particular pixel has been classified and that is how you get the semantic segmented output. So, if I have all these pixels which belong to say horse then all these elements will get a value equal to 3. So, that is how you get a semantic segmentation.

Now, as I said that how do you get this probability estimate, you know that every node in the neural network computes an activation value right. So, at the output of these networks when I have a one hot vector of in this case of dimension 4 then what I can do is, I can perform soft max operation over those 4 elements. And, if you perform a soft max operation assuming that I have a soft max layer which performs a soft max operation over those 4 components.

Then all these 4 components will be normalized to have a value between 0 and 1 and that is what is being done here. And, this value that you get tells you gives you some indication of what is the probability to which probability that this pixel belongs to the corresponding class corresponding to it is index in that one hot vector ok.

So, let me stop here today, in my next class I will talk about the other loss function which is dice loss that can be used for training the neural network performing semantic segmentation. And, I will also try to talk about another application which is denoising of input image using the same deconvolutional neural network.

Thank you.