ConvNet Lab for DD2427

Miquel Marti Rabadan
921019-1459
miquelmr@kth.se

May 9, 2016

1 ConvNet building blocks

1.1 Convolution

1.1.1 Convolution by a single filter

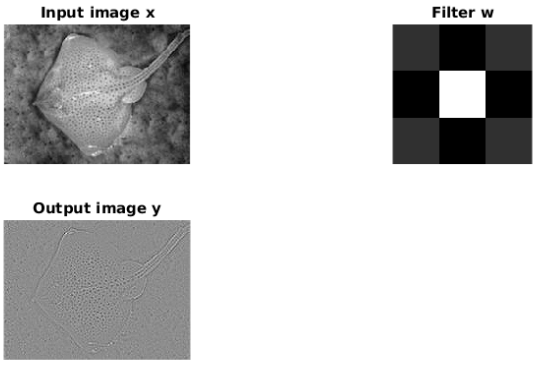

Figure 1: Convolution with a single filter.

Question 1: If H x W is the size of the input image, H′ x W′ the size of the filter, what is the size H″ x W″ of the output image? Why? Different strategies apply giving different results. If only using the valid part of the

image, no zero-padding, the resulting size is H''=H-(H'-1) x W''=W-(W'-1) because the value cannot be computed in the edges as there is no value to use when applying the kernel.

Question 2: The filter w given above is a discretized Laplacian operator. Which type of visual structures (corners, bars, ...) do you think may excite this filter the most? It will excite the most the edges, colour level discontinuities in the image.

Task 1: Suggest a way that you can make the output size be same as the input image size? Apply your suggestion using vl_nnconv. Using zero padding around the image so the image edges have values around it that allow to compute the operation. y = vl_nnconv(x, w, [],'Pad',1) ; Adds padding around the pixels of the input image before operating.

1.1.2 Convolution by a filter bank

Question 3: What is the number of feature channels K in this example? Why? The number of feature channels is 3, one for each of the filters in the filter bank.

Task 2: Run the code above and visualize the individual feature channels in the tensor y by using the provided function showFeatureChannels(). Do the channel responses make sense given the filter used to generate them? Feature channel 1 shows the same result as in previous section. Feature channel 2 enhances more the vertical edges while feature channel 3 the horizontal.

Task 3: Implement the Sobel kernel as another channel and compare the results. Do you see any difference? Explain. The difference is difficult to see but computing it and showing the result shows that there is a difference, mainly in the center of the edges which seem to have higher value.

Question 4: If the input tensor x has C feature channels, what size should be the third dimension of w? It should have size C, one slice for each channel.

Question 5: In the code above, the command wbank = cat(4, w1, w2, w3) concatenates the tensors w1, w2, and w3 along the fourth dimension. Why are we interested in filters with 3 dimensions? Because as said in the previous question each filter should have as many 2D arrays as channels in the input tensor.

1.1.3 Convolving a batch of images

Task 4: Run the code above and visualize the result. Convince yourself that each filter is applied to each image. Explain the procedure you used to check this? First using `size(y)` gives: 148x198x5x2, being 5 the number of channels and 2 the number of images that were concatenated in the 4th dimension of the input tensor x. Using `showFeatureChannels(y(:,:,:,1)` the resulting feature channels of the first image are shown, same can be done for the second. See Figure 2.

Question 6: Why are we interested in working on (mini) batches of images instead of single images? Give all the reasons that you know. We are interested in doing that because we can re-use a filter bank and apply it at the same time against a larger number of images. Also it is needed for performing mini-batch stochastic gradient descent (SGD).

1.2 Non-linear activation: ReLU

Question 7: What happens if all layers are linear? That they can only represent linear functions as the sum of linear layers is linear.

Task 5: Run this code and visualize images x, y, and z. The result can be seen in Figure 3.

Question 8: Which kind of image structures are preferred by this filter? Isolated dots with high value.

Question 9: Why did we negate the Laplacian? So that opposite to the first time it was used dots have high value and edges low and when the non-linear ReLU is applied only the dots survive, everything else has the same value of 0.

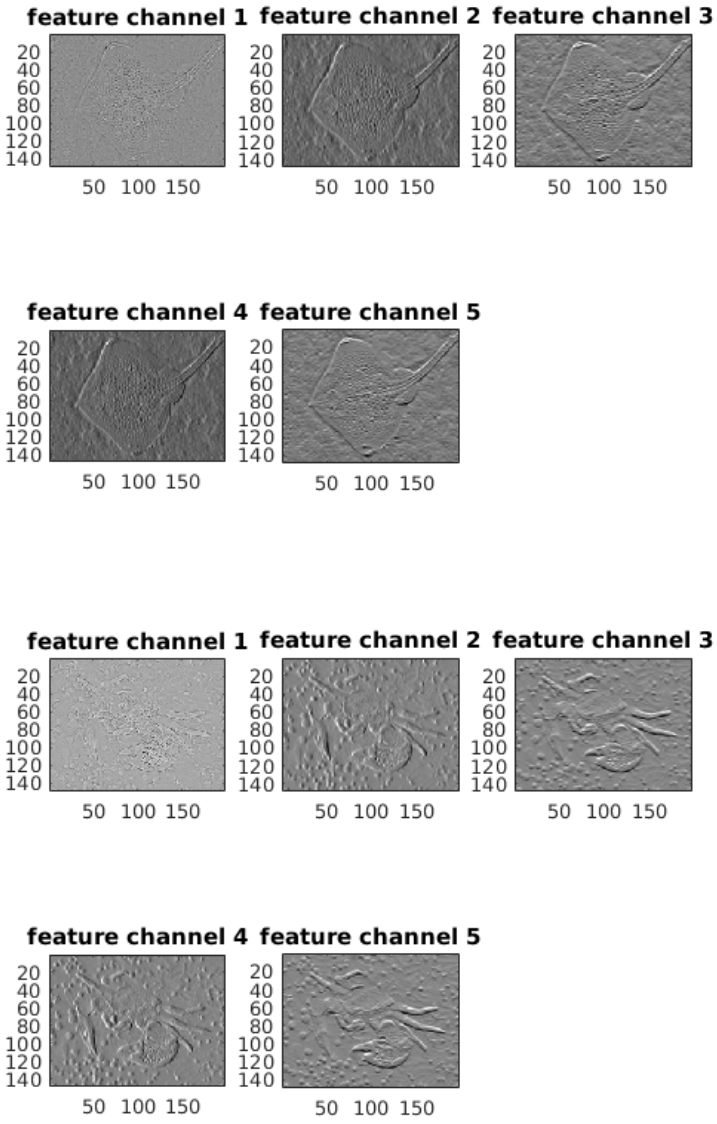

Figure 2: Convolution of batches with filter bank.

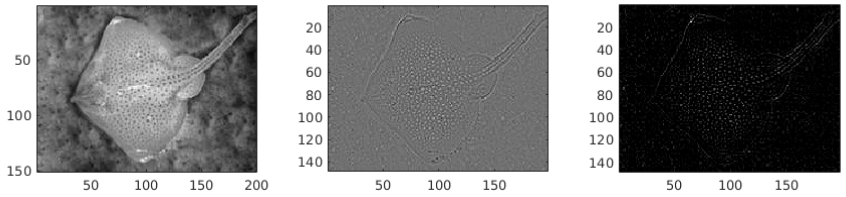

Figure 3: Left to right: original, filtered image and through non-linearity.

Task 6: Repeat the same procedure and replace the non-linearity function with sigmoid and tanh. Explain the differences you see between the results of these three operations. Particularly, how is the distribution of the values out of these three activation functions different? Figure 4 shwos the different activation functions after the negated Laplacian filter is applied. The tanh and the sigmoid functions behave in a similar way, compressing to high and low values, while the ReLU sets all values below 0 to that value. Compressing only the lower end.

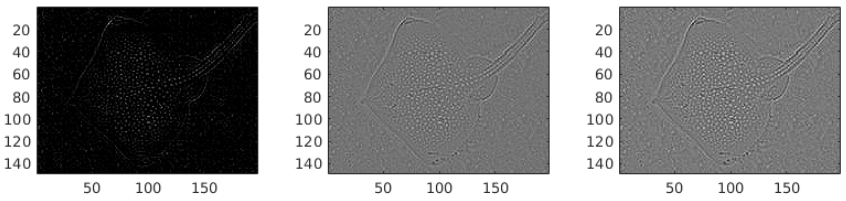

Figure 4: Different activation functions. Left to right: ReLU, sigmoid and tanh.

Task 7: Run this code and visualize images x, y, and z. Figure 5 shows the result.

Question 10: Is the response now more selective? It is as now fewer pixels are over the zero value.

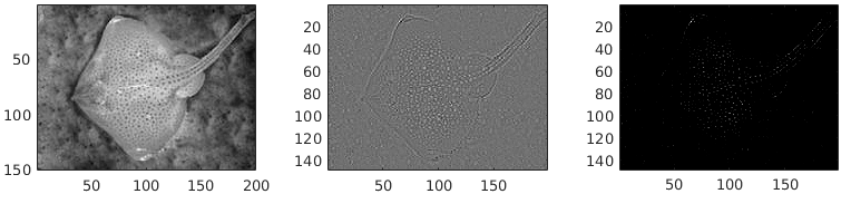

Figure 5: ReLU with bias of -0.2

Task 8: Take a look at the available functionalities provided by MatConvNet. Find the functionality that can do both max pooling and average pooling. Apply both max and average pooling with a couple of different sizes on both the input image and the output of the convolution. Visualize the results. Discuss the results. Figure 6 shows the result of applying max and average pooling to the input image and to the output image with different sizes. With average pooling the image gets blurred while still being recognizable. The larger the size of the pooling filter the more blur the image gets. With max pooling, the images get divided in super-pixels and the contrast seems to be enhanced. This can be better seen in the images after the filter is applied. This suggest doing subsampling.

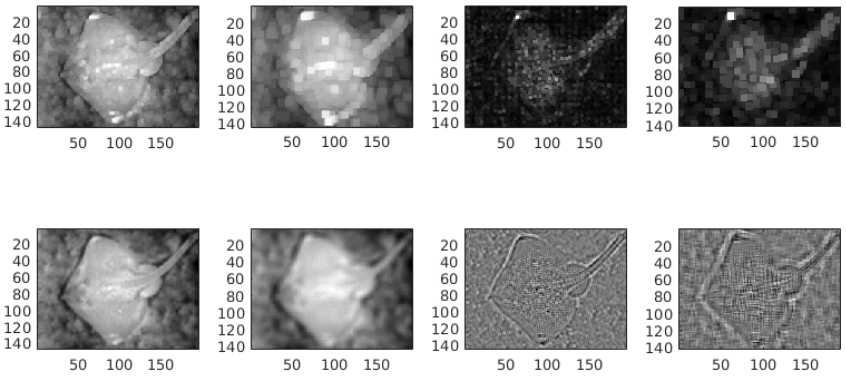

Figure 6: Max (top) and average (bottom) pooling examples.

2 Backpropagation

Question 10: The output derivatives have the same size as the parameters in the network. Why? They have the same size because, applying the chain rule, each of the derivatives of the output is computed as a number of derivatives as well depending on the connections of the net which are in turn also computed in the same way according to the connections and functions to previous layers or input.

Question 11: From the procedure above what kind of derivative is the vector p_l representing at each step l? It is representing the partial derivatives of layer l respect to x.

2.1 Backward mode verification

Question 12: Open the file checkDerivativeNumerically.m. What is it trying to do? Identify the lines in which the above expression should be evaluated. Takes the scalar function F, its tensor input X and its derivative DX at X and compares DX to a numerical approximation of the derivative returing their difference ERR.

Question 13: Note that checkDerivativeNumerically() is applied to the function @(x) proj(p,vl nnconv(x, w, [])). This syntax defines a function on the fly (an anonymous closure to be more precise). In this case, the purpose of the closure is to evaluate the expression for a variable x and a fixed value of w. Furthermore, the closure projects the output of vl_nnconv() onto p by calling the proj() function. Why? Because the numerical derivatives computed in the forward mode have to be compared with the projected derivatives computed with the backward mode with the same projection vector.

Task 9: Complete the code in checkDerivativeNumerically() to compute the approximate derivative of f w.r.t. x. Run the code, visualizing the results. Convince yourself that the numerical and analytical derivatives are nearly identical. Figure 7 shows the result of checkDerivativeNumerically(). It can be seen that the difference between both approaches to compute the derivatives is almost null.

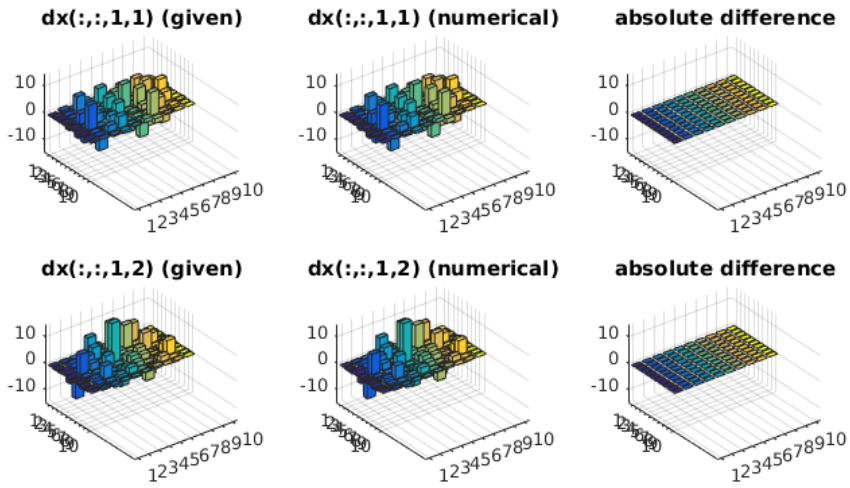

Figure 7: Comparison between backward mode and forward mode with numerical derivative.

Task 10: Modify the code to compute the derivative of the first element of the output tensor y with respect to all the elements of the input tensor x. Hint: it suffices to change the value of p.

Task 11: Modify the code to compute the derivative with respect to the convolution parameters w instead of the convolution input x.

Question 14: In the code above, in backward mode the projection p is fed to the vl nnrelu operator. However, the vl nnconv operator now receives dy as projection. Why? Because dy is already the projected derivative and thus includes the projection.

Task 12: Run the code and use checkDerivativeNumerically() to compare the analytical and numerical derivatives. Do they differ? Figure 10 shows the result of implementing backprop in a network of two layers. They are mostly equal but a small difference can be seen in the second image in the batch around (6,8).

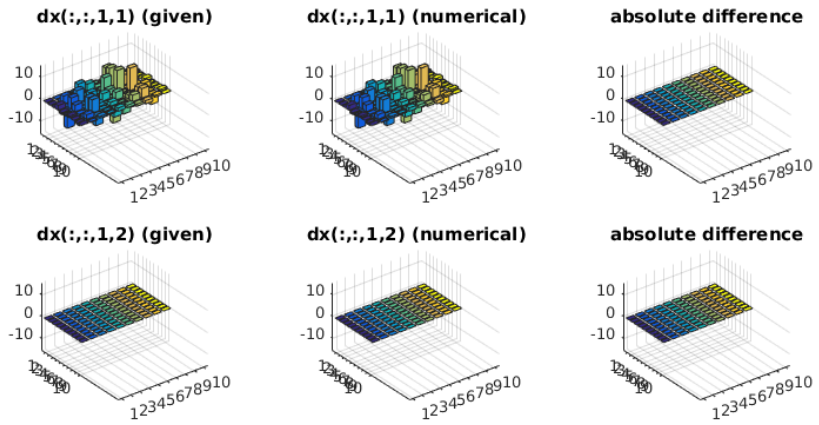

Figure 8: Comparison between backward mode and forward mode with numerical derivative for first element of output tensor y.

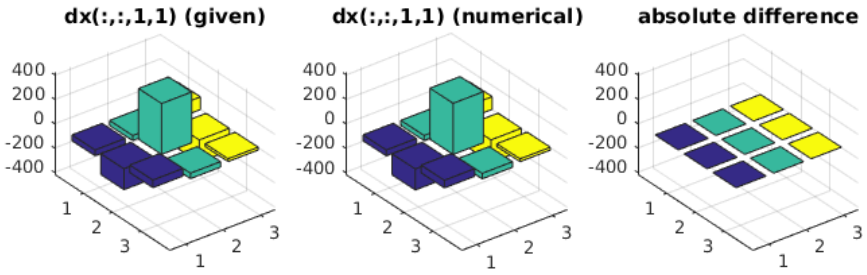

Figure 9: Comparison between backward mode and forward mode with numerical derivative for first element of output tensor y respect convolution parameters w.

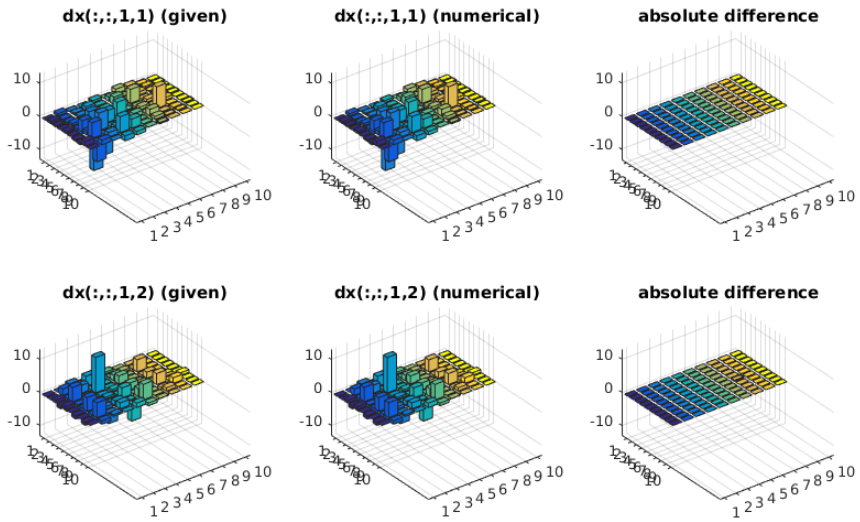

Figure 10: Projected derivatives with forward and backward mode in a two layer network.

2.2 Design and verify your own layer

Task 13: Verify that the forward and backward functions are correct by computing the derivatives numerically using checkDerivativeNumerically(). Figure 11 shows the correctness of the forward and bacward functions.

Task 14: Implement the l1LossForward.m and l1LossBackward.m to compute the L1 distance (sum of absolute differences). Make sure that both the forward and backward modes are correctly modified by verifying the result numerically once more. What happens for the components of x that are zero or very close to zero? Figure 12 shows the resulting derivatives computed with backward and forward mode for the L1Loss function. When the derivatives are very small some differences actually appear between both ways of computing the derivatives.

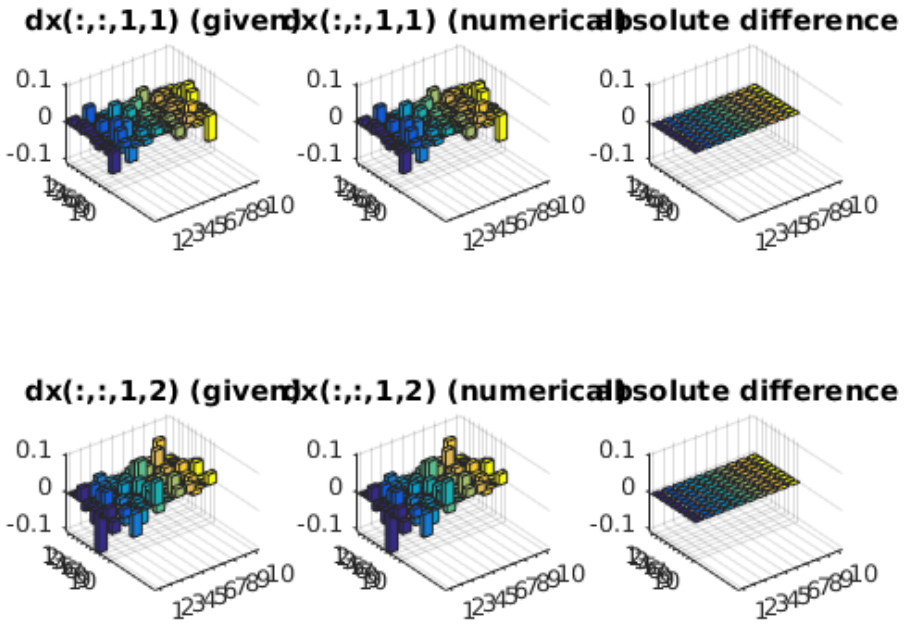

Figure 11: Derivatives of backward and forward functions for l2Loss function.

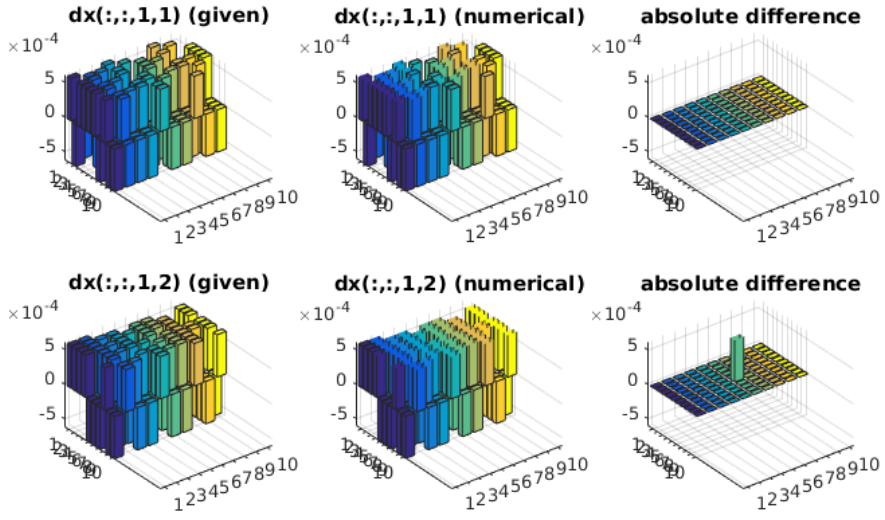

Figure 12: Derivatives of backward and forward functions for l1Loss function.