Part 1:

1. You will have $H_2xW_2=(H-(H_1-1))x(W-(W_1-1))$ because no zero padding
2. The Laplacian operator detects changes in intensity in the image, so edges
3. (Task) Use the 'pad' option in vl_nnconv and pass in 1 to surround the image with one layer of zeros.
4. Since we apply 3 filters, we should have 3 feature channels
5. (Task) Yes, the channels make sense, channel 2 looks like the change in x-axis and channel 3 as the change in y-axis
6. (Task) The sobel kernel takes both the x and y axis into consideration
7. The third dimension of w should by a fraction of C (C should be divide-able by it)
8. Because the image is represented by 3 dimensions and we might want to treat the feature channels in x differently
9. (Task) I looked at the 3 feature channels in y for the entirety of the 4^{th} dimension and saw that the images were stored there
10. It is needed for batch SGD and batch normalization, which both help to give a smother convergence (smoother as in less noisy). Sometimes batches are also necessary for processing continuous data. (Although noisier data makes it more difficult to get stuck in local minima and online training allows for larger step size and faster convergence)
11. If all layers are linear, it's the same thing as only having one layer
12. For Laplacian filter, edges are preferred since it highlights intensity changes in the image.
13. Laplacian responds with negative values when we find increasing intensity changes, so we negate it to get positive values for the positive changes.
14. The distribution of the values for the ReLu are very close to 0, only above for positive responses, while they for sigmoid are a lot higher. I could not find any function for applying tanh in the library, but it would look fairly similar to the sigmoid, only centered around 0 instead of 0.5.
15. The response becomes more selective.
16. The average makes a blur, good for smoothening the image to remove noise, also makes it easier for the network to extract features. The max gives translational invariance, good for image recognition.

Part 2:

1. Because you are deriving with respect to the parameters, and a derivative won't change the dimensions.
2. The chain of partial derivatives in x up to the point "l", much like δ in classic back-prop.
3. On lines 16-18, it is trying to fill a tensor with approximated errors using the definition of the derivative so that it can compare that with MatConvNets faster way.
4. That way we instantly get p'*y(x,w).

5. Because vl_nnrelu(z,p) computes the derivative of the ReLU layer and then projects it onto p, so now our dy is our p going into the conv from the ReLU (backwards).
6. They differ quite a lot.
7. We get a spike in error when a x becomes close to 0

Part 3:

1. There are 2550 training images and 902 validation images.
2. White is usually the highest intensity, which in this case would correspond to 0 because of the shift. The conv operation uses padding of zeros around the image. The reason that interval was chosen is probably because of most of the edges are white (it's a white background for the text) and the 0 padding does not do any damage if white also is 0.
3. If you applied a 3x3 filter without using padding, the output would be 2 smaller in both width and height.
4. Because we want the output to become similar to the labels, and added nonlinearity in this case does not help with that. Not to mention that it will be very difficult to get the interval [-1,0] with a ReLU just before the output.
5. 5 layers (if you don't count the calculation of the loss, which you usually don't).
6. In order from left to right (not counting ReLU since always support 1 and n/a "num filts" (should be one since only 1 in depth)):
   support: 3, 3, 3
   depth: 32, 32, 1
7. Filter dim in the table is the number of feature channels in the input to the filter.
8. It's:
   outX*outY*FeatureChannels*(FilterSizeX*FilterSizeY*PrevoiusLayersFeatureChannels+1)
9. The receptive field for one output pixel is 7x7 in the input pixel. A large enough receptive field to encompass the majority of a letter is probably to be preferred in this network. You can get a larger receptive field by either increasing one of the filters in a conv layer, or by adding another conv layer. For example, increasing the any layer's filter to a 5x5 would increase the receptive field to 9x9.
10. Increasing the batch leads to faster training time (same hz but less iterations per epoch), but also worse performance since it does not converge as fast, although the convergence does become a lot smoother.
11. The network did a decent job, but it's still somewhat difficult to see what it says.
12. Barely, the training performance might be slightly better, but it's so little that it is difficult to see.