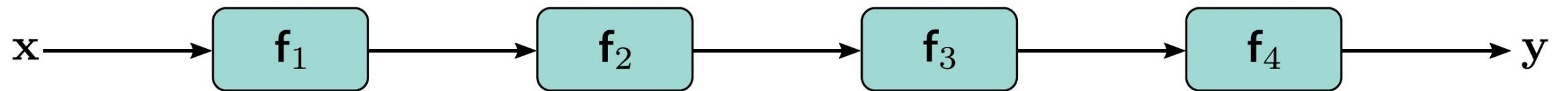
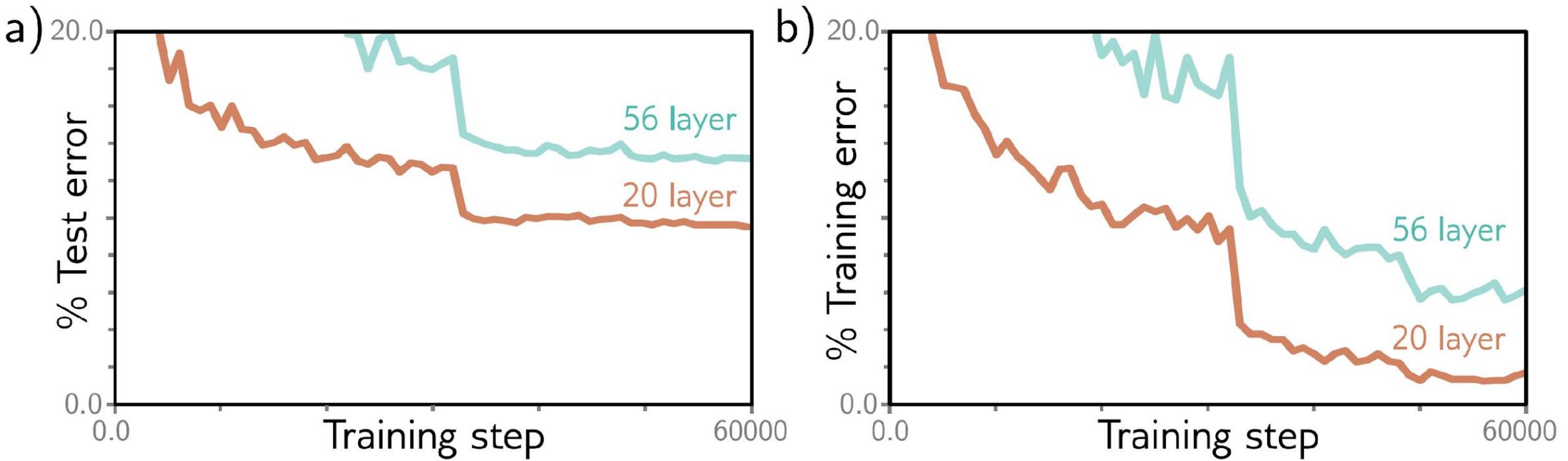


# Understanding Deep Learning

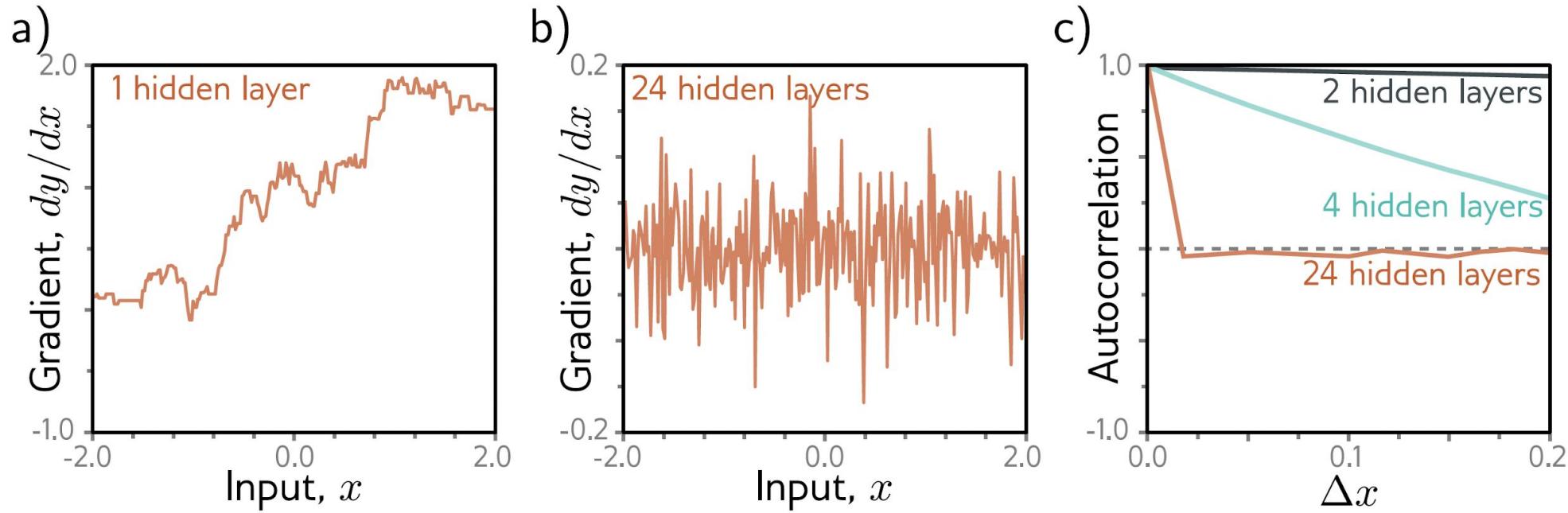
Chapter 9: Residual Networks



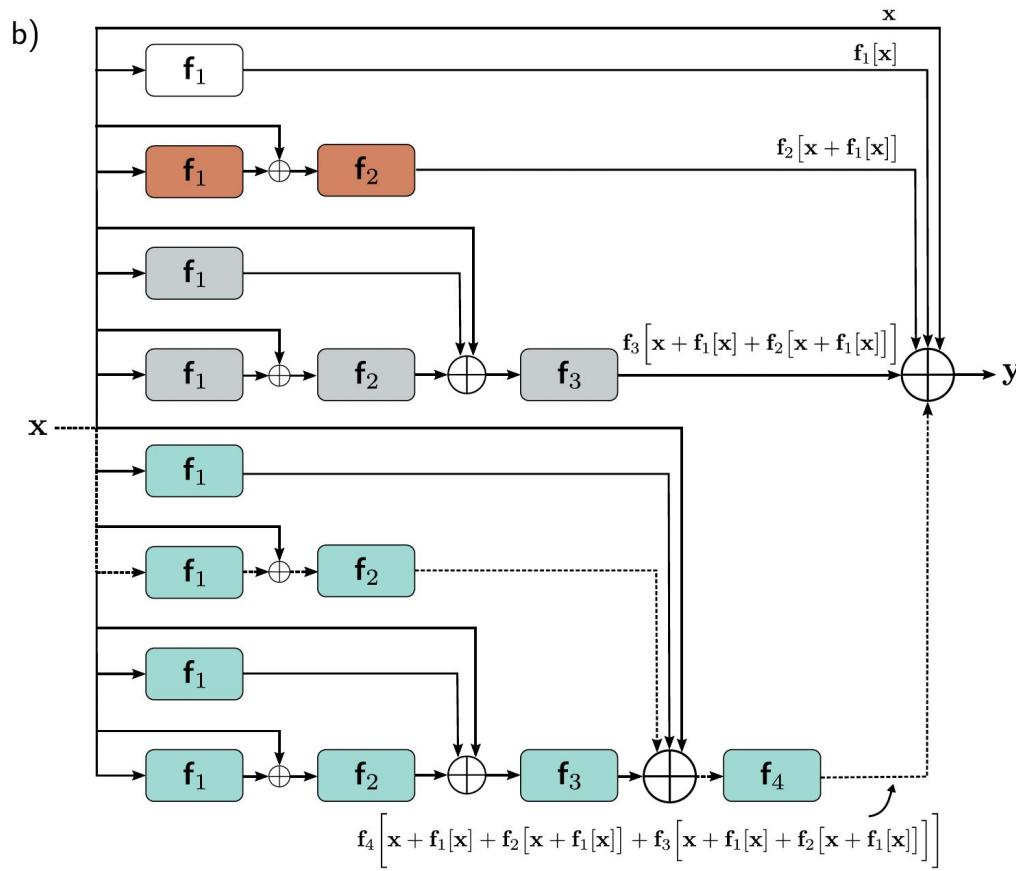
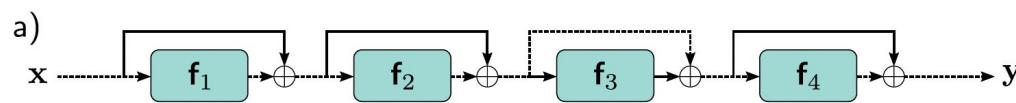
**Figure 11.1** Sequential processing. Standard neural networks pass the output of each layer directly into the next layer.



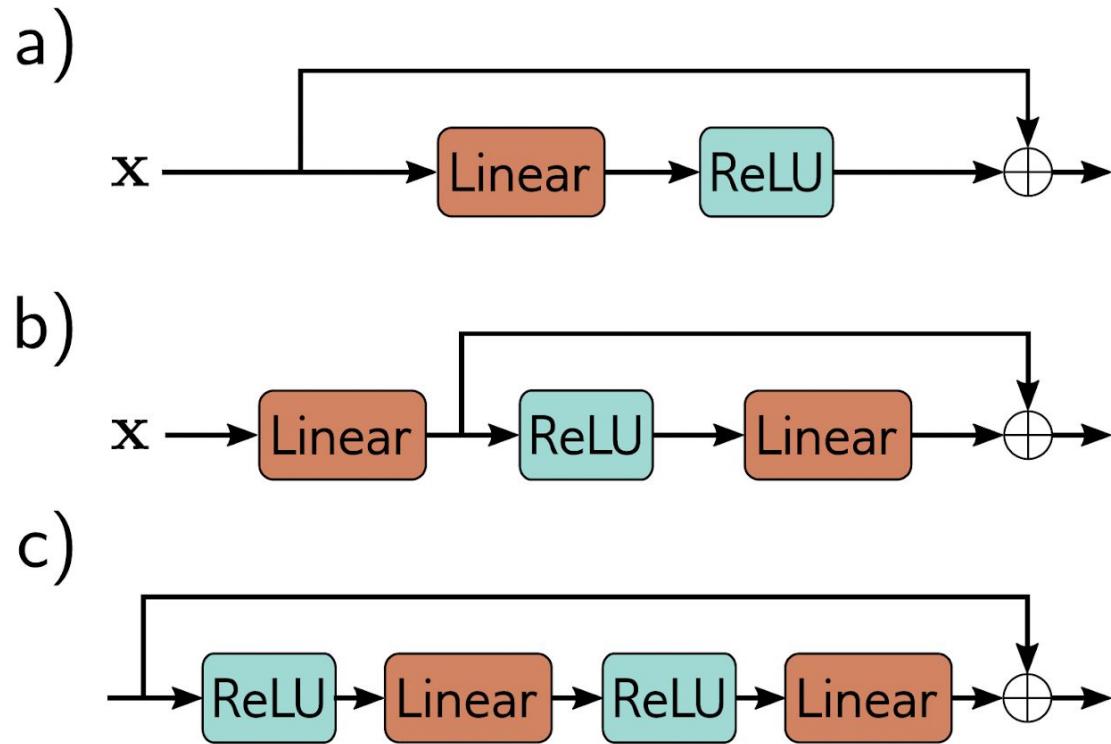
**Figure 11.2** Decrease in performance when adding more convolutional layers. a) A 20-layer convolutional network outperforms a 56-layer neural network for image classification on the test set of the CIFAR-10 dataset (Krizhevsky & Hinton, 2009). b) This is also true for the training set, which suggests that the problem relates to training the original network rather than a failure to generalize to new data. Adapted from He et al. (2016a).



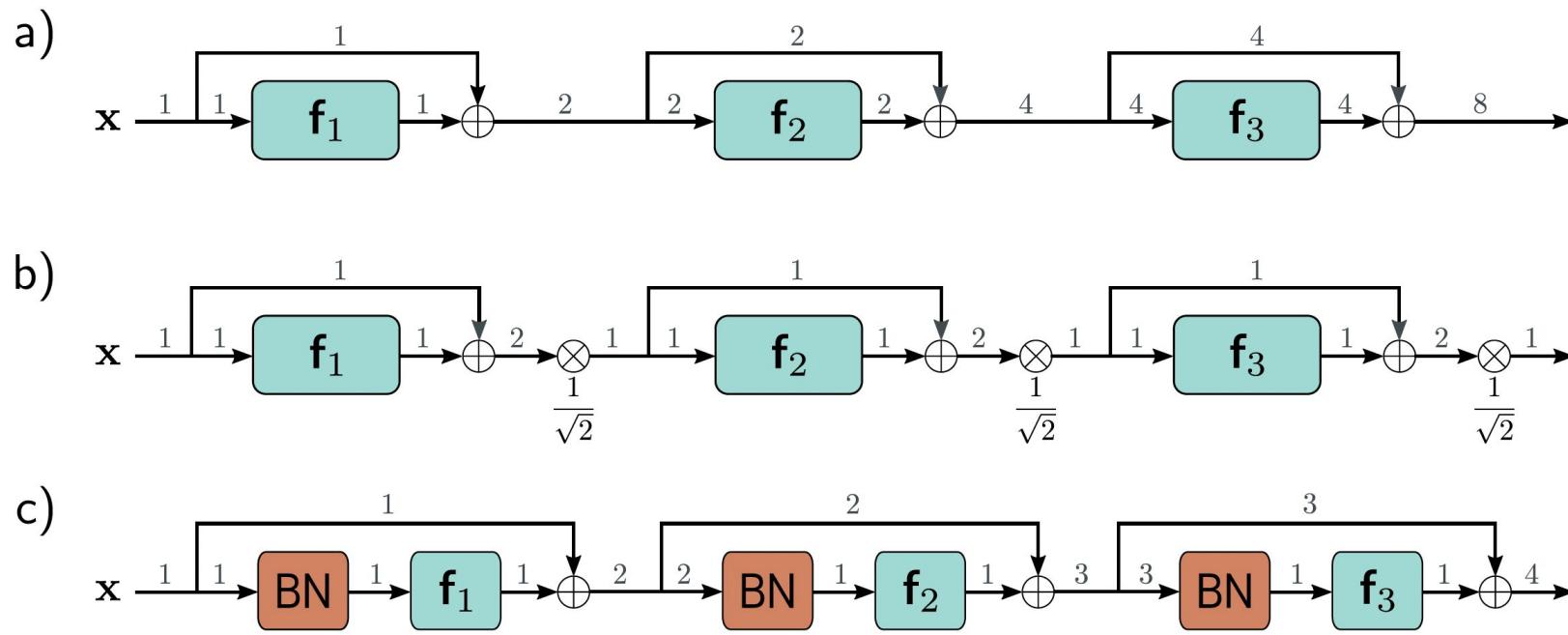
**Figure 11.3** Shattered gradients. a) Consider a shallow network with 200 hidden units and Glorot initialization (He initialization without the factor of two) for both the weights and biases. The gradient  $\partial y / \partial x$  of the scalar network output  $y$  with respect to the scalar input  $x$  changes relatively slowly as we change the input  $x$ . b) For a deep network with 24 layers and 200 hidden units per layer, this gradient changes very quickly and unpredictably. c) The autocorrelation function of the gradient shows that nearby gradients become unrelated (have autocorrelation close to zero) for deep networks. This *shattered gradients* phenomenon may explain why it is hard to train deep networks. Gradient descent algorithms rely on the loss surface being relatively smooth, so the gradients should be related before and after each update step. Adapted from Balduzzi et al. (2017).



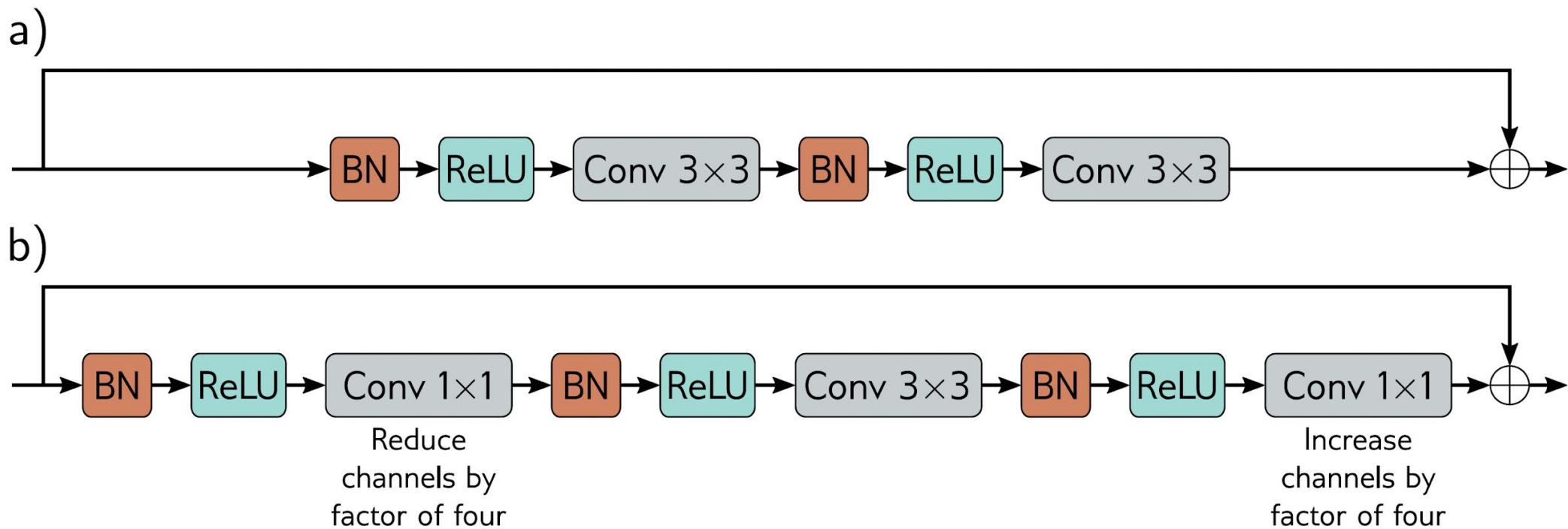
**Figure 11.4** Residual connections. a) The output of each function  $f_k[\mathbf{x}, \phi_k]$  is added back to its input, which is passed via a parallel computational path called a residual or skip connection. Hence, the function computes an additive change to the representation. b) Upon expanding (unraveling) the network equations, we find that the output is the sum of the input plus four smaller networks (depicted in white, orange, gray, and cyan, respectively, and corresponding to terms in equation 11.5); we can think of this as an ensemble of networks. Moreover, the output from the cyan network is itself a transformation  $f_4[\bullet, \phi_4]$  of another ensemble, and so on. Alternatively, we can consider the network as a combination of 16 different paths through the computational graph. One example is the dashed path from input  $\mathbf{x}$  to output  $\mathbf{y}$ , which is the same in panels (a) and (b).



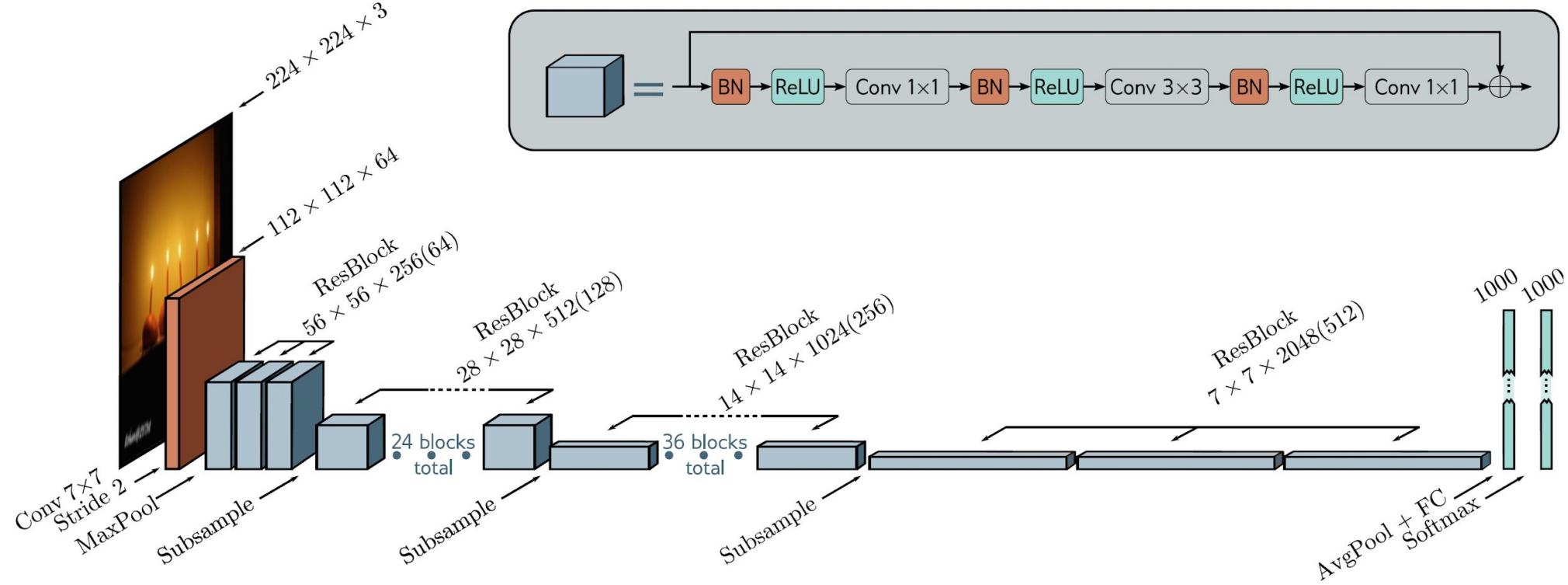
**Figure 11.5** Order of operations in residual blocks. a) The usual order of linear transformation or convolution followed by a ReLU nonlinearity means that each residual block can only add non-negative quantities. b) With the reverse order, both positive and negative quantities can be added. However, we must add a linear transformation at the start of the network in case the input is all negative. c) In practice, it's common for a residual block to contain several network layers.



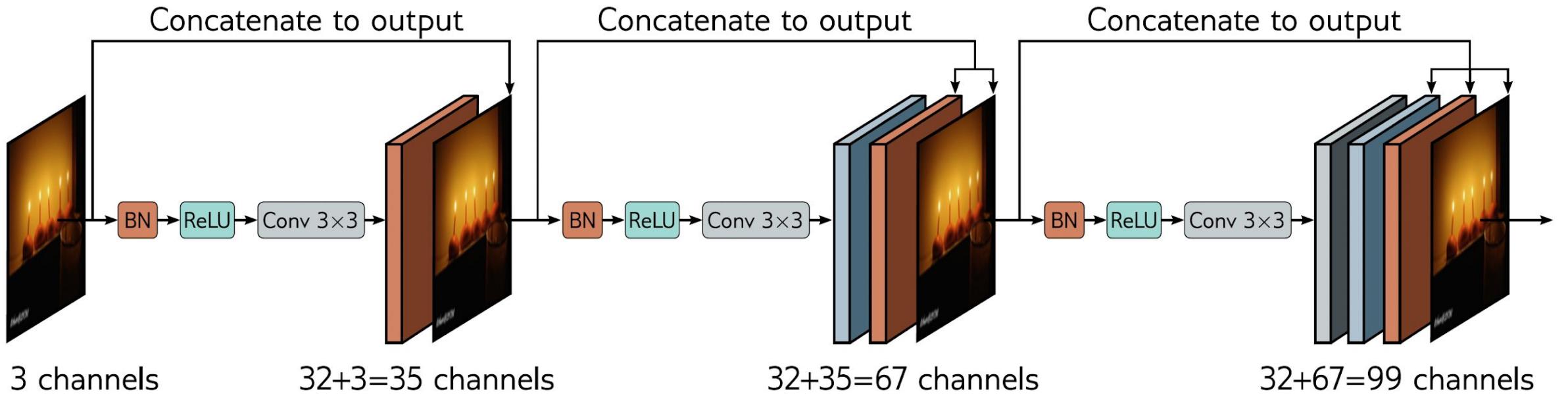
**Figure 11.6** Variance in residual networks. a) He initialization ensures that the expected variance remains unchanged after a linear plus ReLU layer  $f_k$ . Unfortunately, in residual networks, the input of each block is added back to the output, so the variance doubles at each layer (gray numbers indicate variance) and grows exponentially. b) One approach would be to rescale the signal by  $1/\sqrt{2}$  between each residual block. c) A second method uses batch normalization (BN) as the first step in the residual block and initializes the associated offset  $\delta$  to zero and scale  $\gamma$  to one. This transforms the input to each layer to have unit variance, and with He initialization, the output variance will also be one. Now the variance increases linearly with the number of residual blocks. A side-effect is that, at initialization, later network layers are dominated by the residual connection and are hence close to computing the identity.



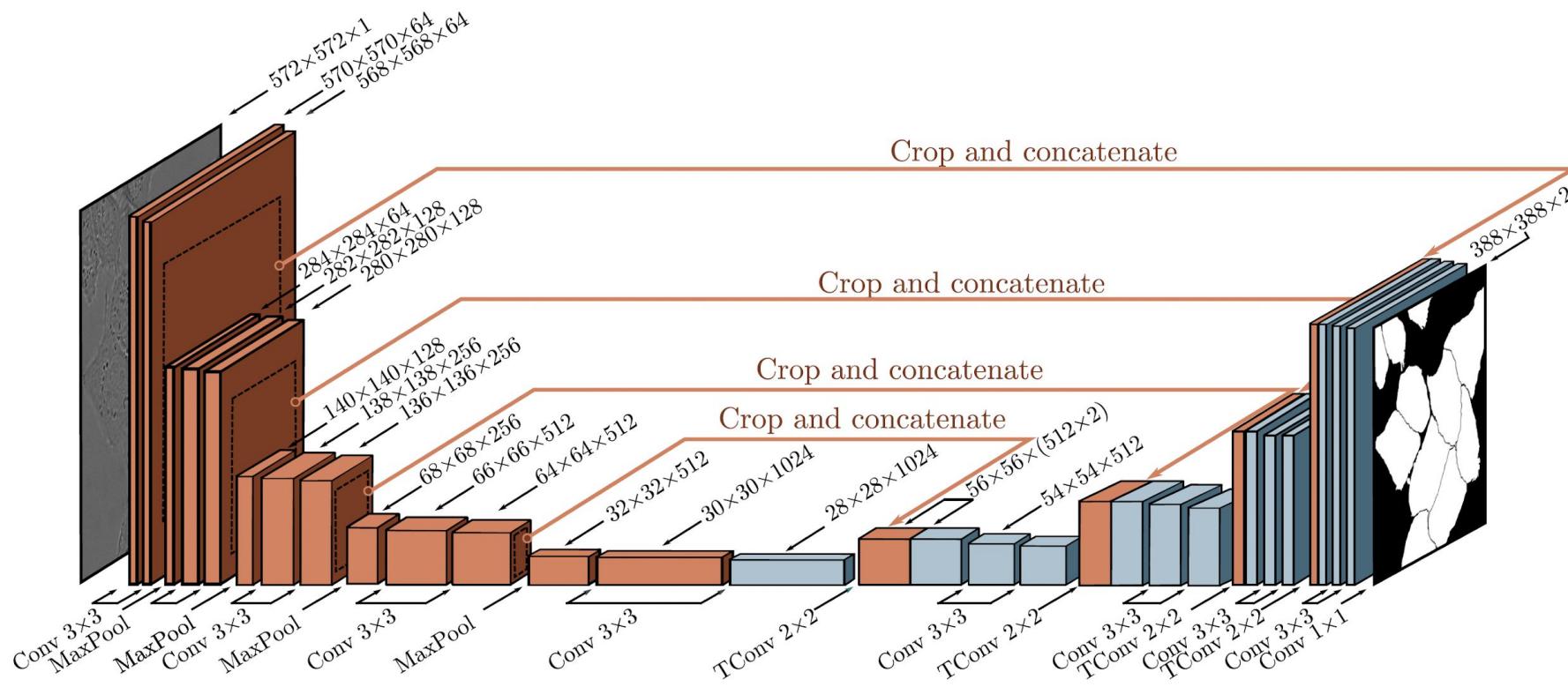
**Figure 11.7** ResNet blocks. a) A standard block in the ResNet architecture contains a batch normalization operation, followed by an activation function, and a  $3 \times 3$  convolutional layer. Then, this sequence is repeated. b). A bottleneck ResNet block still integrates information over a  $3 \times 3$  region but uses fewer parameters. It contains three convolutions. The first  $1 \times 1$  convolution reduces the number of channels. The second  $3 \times 3$  convolution is applied to the smaller representation. A final  $1 \times 1$  convolution increases the number of channels again so that it can be added back to the input.



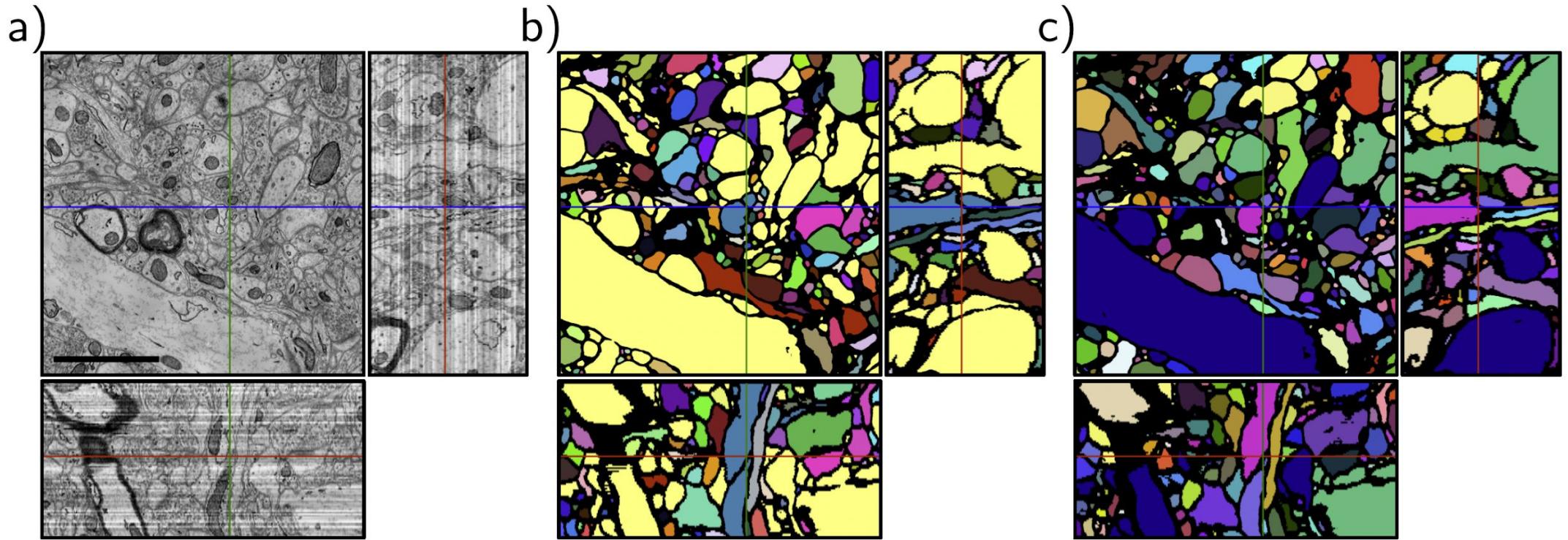
**Figure 11.8** ResNet-200 model. A standard  $7 \times 7$  convolutional layer with stride two is applied, followed by a MaxPool operation. A series of bottleneck residual blocks follow (number in brackets is channels after first  $1 \times 1$  convolution), with periodic downsampling and accompanying increases in the number of channels. The network concludes with average pooling across all spatial positions and a fully connected layer that maps to pre-softmax activations.



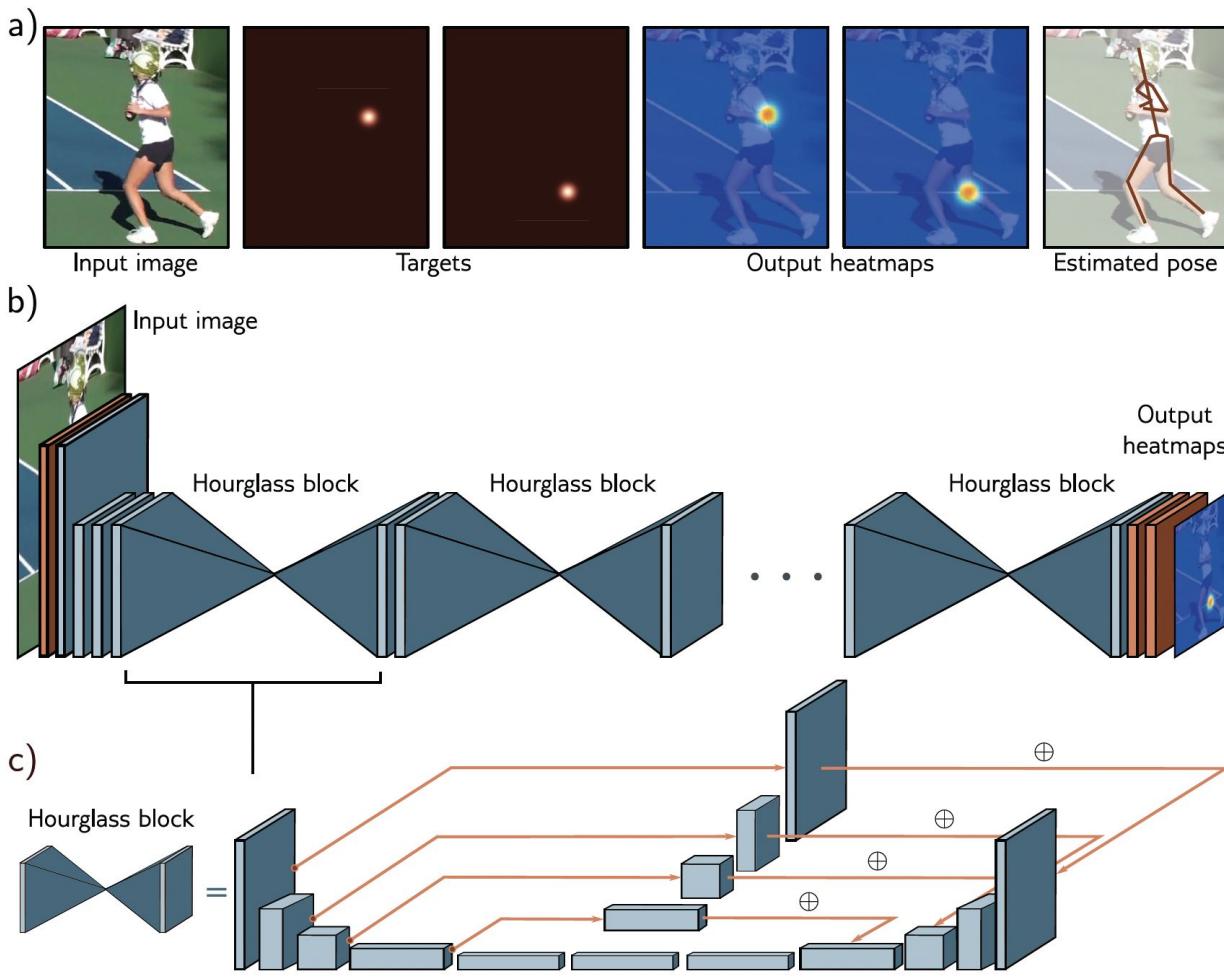
**Figure 11.9** DenseNet. This architecture uses residual connections to concatenate the outputs of earlier layers to later ones. Here, the three-channel input image is processed to form a 32-channel representation. The input image is concatenated to this to give a total of 35 channels. This combined representation is processed to create another 32-channel representation, and both earlier representations are concatenated to this to create a total of 67 channels and so on.



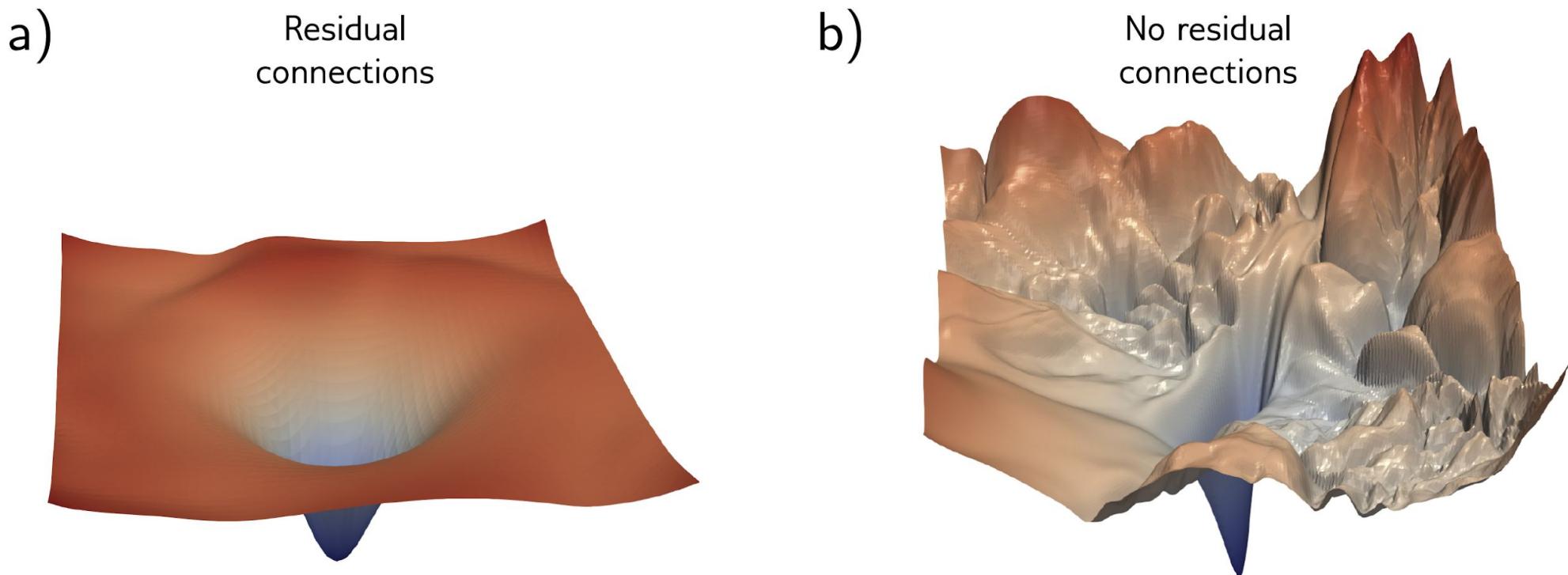
**Figure 11.10** U-Net for segmenting HeLa cells. The U-Net has an encoder-decoder structure, in which the representation is downsampled (orange blocks) and then re-upsampled (blue blocks). The encoder uses regular convolutions, and the decoder uses transposed convolutions. Residual connections append the last representation at each scale in the encoder to the first representation at the same scale in the decoder (orange arrows). The original U-Net used “valid” convolutions, so the size decreased slightly with each layer, even without downsampling. Hence, the representations from the encoder were cropped (dashed squares) before appending to the decoder. Adapted from Ronneberger et al. (2015).



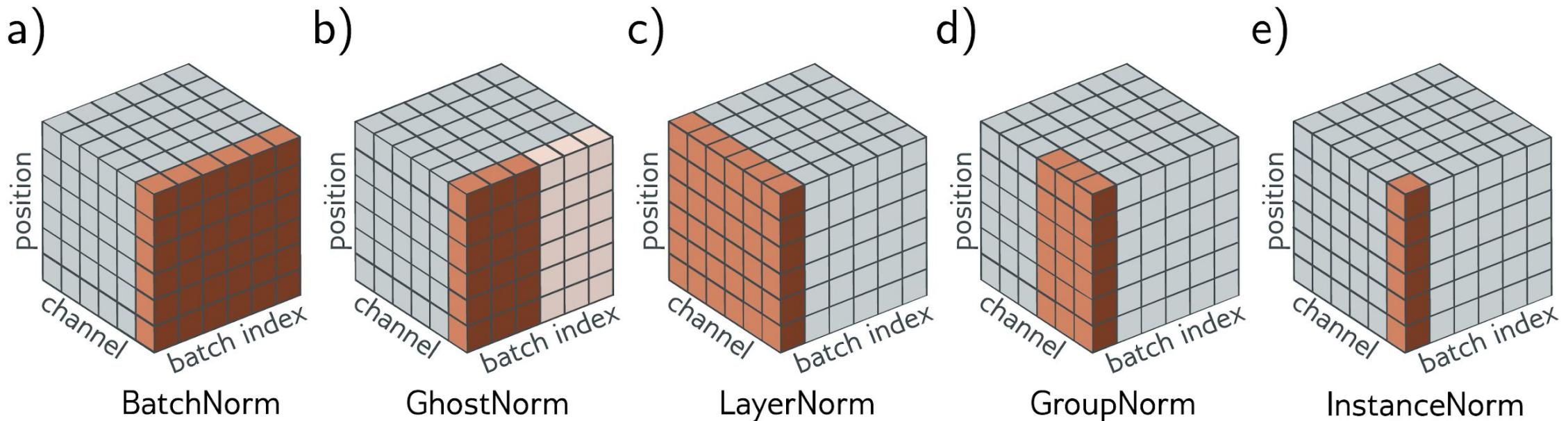
**Figure 11.11** Segmentation using U-Net in 3D. a) Three slices through a 3D volume of mouse cortex taken by scanning electron microscope. b) A single U-Net is used to classify voxels as being inside or outside neurites. Connected regions are identified with different colors. c) For a better result, an ensemble of five U-Nets is trained, and a voxel is only classified as belonging to the cell if all five networks agree. Adapted from Falk et al. (2019).



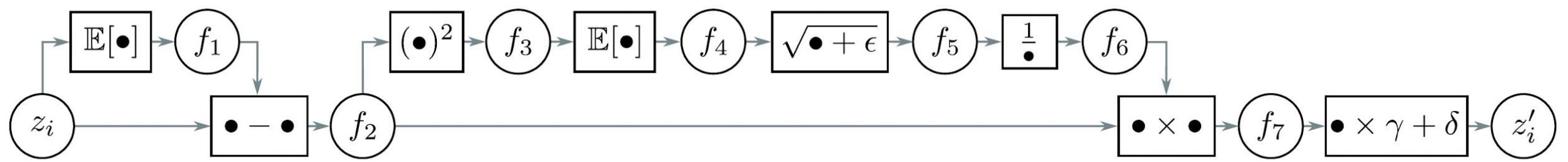
**Figure 11.12** Stacked hourglass networks for pose estimation. a) The network input is an image containing a person, and the output is a set of heatmaps, with one heatmap for each joint. This is formulated as a regression problem where the targets are heatmap images with small, highlighted regions at the ground-truth joint positions. The peak of the estimated heatmap is used to establish each final joint position. b) The architecture consists of initial convolutional and residual layers followed by a series of hourglass blocks. c) Each hourglass block consists of an encoder-decoder network similar to the U-Net except that the convolutions use zero padding, some further processing is done in the residual links, and these links add this processed representation rather than concatenate it. Each blue cuboid is itself a bottleneck residual block (figure 11.7b). Adapted from Newell et al. (2016).



**Figure 11.13** Visualizing neural network loss surfaces. Each plot shows the loss surface in two random directions in parameter space around the minimum found by SGD for an image classification task on the CIFAR-10 dataset. These directions are normalized to facilitate side-by-side comparison. a) Residual net with 56 layers. b) Results from the same network without skip connections. The surface is smoother with the skip connections. This facilitates learning and makes the final network performance more robust to minor errors in the parameters, so it will likely generalize better. Adapted from Li et al. (2018b).



**Figure 11.14** Normalization schemes. BatchNorm modifies each channel separately but adjusts each batch member in the same way based on statistics gathered across the batch and spatial position. Ghost BatchNorm computes these statistics from only part of the batch to make them more variable. LayerNorm computes statistics for each batch member separately, based on statistics gathered across the channels and spatial position. It retains a separate learned scaling factor for each channel. GroupNorm normalizes within each group of channels and also retains a separate scale and offset parameter for each channel. InstanceNorm normalizes within each channel separately, computing the statistics only across spatial position. Adapted from Wu & He (2018).



**Figure 11.15** Computational graph for batch normalization (see problem 11.10).