5. Encoder-Decoder Networks (EDNs)

INTRODUCTION

Decoding consists in mapping an encoded quantity, usually described into a specific domain, towards a different domain, by means of one or more a set of transforms. The encoded quantity is usually described as a vector whose elements are extracted from a predefined "alphabet" of symbols. For instance, a binary vector of size *N*, regarded as an integer number encoded by *N* bits, will be suitable to describe a set of 2^{N-1} encoded quantities. For example, a vector of 3 bits might be used to encode 8 different shapes (circle, triangle, square, rhombus, …). The transformation of one encoded vector back to the corresponding shape is just the decoding process. A decoding transform therefore might be regarded as a process that makes the information readable/interpretable to a higher level. The next picture depicts the mapping from an encoded quantity spanning a low scale input domain into a decoded quantity spanning a more complex output domain. Increasing the complexity of the decoded quantity impacts on the decoding model in terms of accuracy and reliability, demanding appropriate transforms (some a priori knowledge should be required for example) to ensure quality of the result. Additionally, decoding may be implemented by progressive steps addressing the overall complexity split in steps of less reduced complexity in a progressive decoding process.

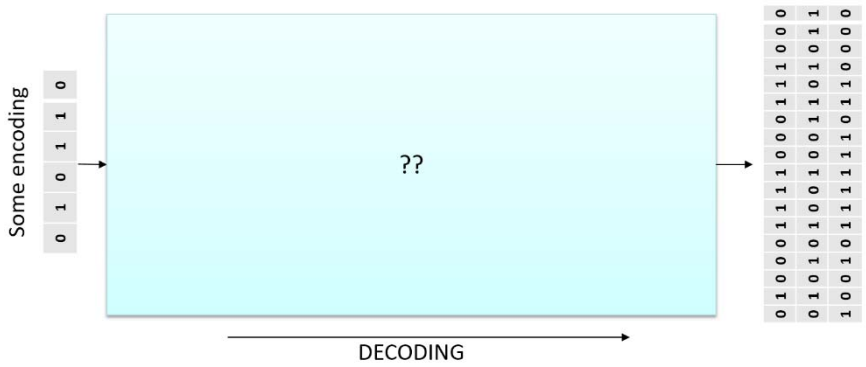


Figure 1. General decoding schema: from an encoded input quantity to a decoded output quantity usually spanning a more complex domain than that one of the input.

Information decompression and data up-sampling can be regarded a specific types of decoding. Up-sampling in particular is the operation which increases the size of a vector quantity, which can span an arbitrary number of dimensions (1D: signal, 2D: image, 3D: volume, 4D: volume in time,…). In case of the familiar image domain, up-sampling means to move from a lower spatial scale to a higher spatial scale, and is based on computing each pixel y_{ij} in the output image by an interpolation map (nearest neighbor, linear, bi-linear,..) applied to the input image.

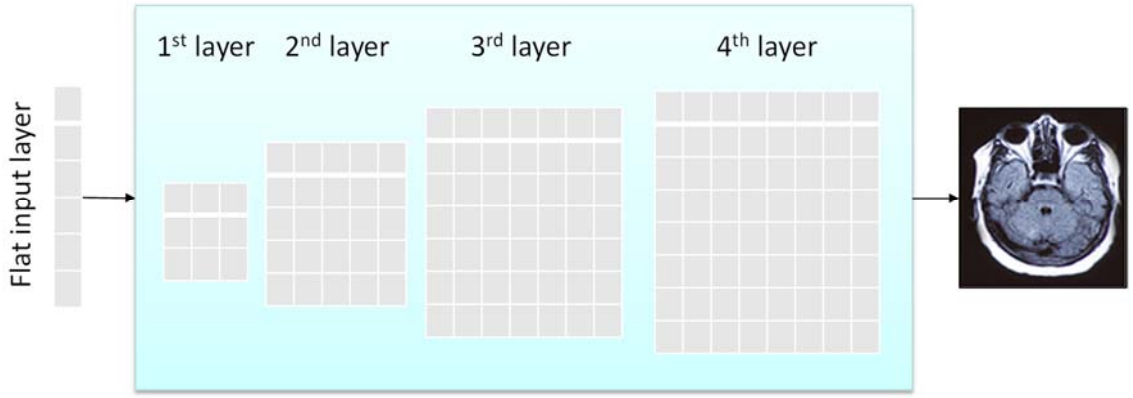


Figure 2. Simplification of a decoding deep network.

This map replicates a one-to-many relationship, say one pixel in the input image will be responsible of many pixels in the output image. The result of the interpolation will depend on the selected map (global vs local, linear vs non-linear, …) and on the reciprocal positions of the input and output corresponding image regions. Moving up-sampling in the field of neural networks means to exploit feed-forward networks that have to be tailored the perform up-sampling by means of a particular type of convolution called deconvolution. The NN approach to optimal deconvolution avoid the use a predefined interpolation method and implements the deconvolution by means of proper convolution filtering, whose weights are computed during the training, so that is the target task that will determine the best deconvolution. The advantage of using deep networks is that the deconvolution process can be spread in more steps for progressive and smoother up-sampling (Fig. 2). From an architectural point of view, enabling sequential deconvolution processing corresponds to define a multi-layer deconvolution network where each layer is devoted to up-sampling its input by a specific factor. Assuming a factor of 2, every deconvolution layer will double the spatial size of its input in the output.

Deconvolution implementation

There exist two main approaches to deconvolution implementation:

1. inverse convolution, also known as transposed convolution, which is based on with pixel-wise multiplication/summation avoiding explicit zero-padding (Fig. 3).
2. direct convolution by zero-padding the input

The example in (Fig. 3) illustrates the first approach. One 4×4 input image must be deconvoluted by means of a 3×3 filter featuring a stride of 1 in both x and y. The process requires first the pixel-wise masking by means of the filter for all the input image pixels to obtain the contributions to the output image of each pixel. Second, the contribution are locally mixed by pixel-wise summation in the field of the output image. This overall operation of deconvolution can be explained in terms of transposed convolution. To the aim, let us take a step back to the convolution operation. Assuming a 4×4 input image and a 3×3 filter with stride 1 (no zero-padding) the output image, of size 2×2, is obtained by shift and matching the filter onto the image step-b-step to cover the entire input image. This convolution operation can be solved nonetheless into one single step by exploiting matrix convolution as depicted in Fig. 4. This is based on one product matrix, obtained by rearranging the filter data in agreement with horizontal f_x and vertical f_y sizes, horizontal L_x and vertical L_y strides and the input image size (I_x, I_y), and one vector of I_x*I_y elements obtained by flattening the image . In detail, in order to make consistent the matrix convolution, the matrix should have a horizontal size equal to the flattened image size (I_x*I_y) and a vertical size equal to the flattened output image size (s_x*s_y), considering that

$$s_x = \frac{(I_x - f_x)}{L_x} + 1$$

$$s_y = \frac{(I_y - f_y)}{L_y} + 1$$

hold (assuming no padding). The product matrix is filled with rows of the 2D filter map taking into account the stride.

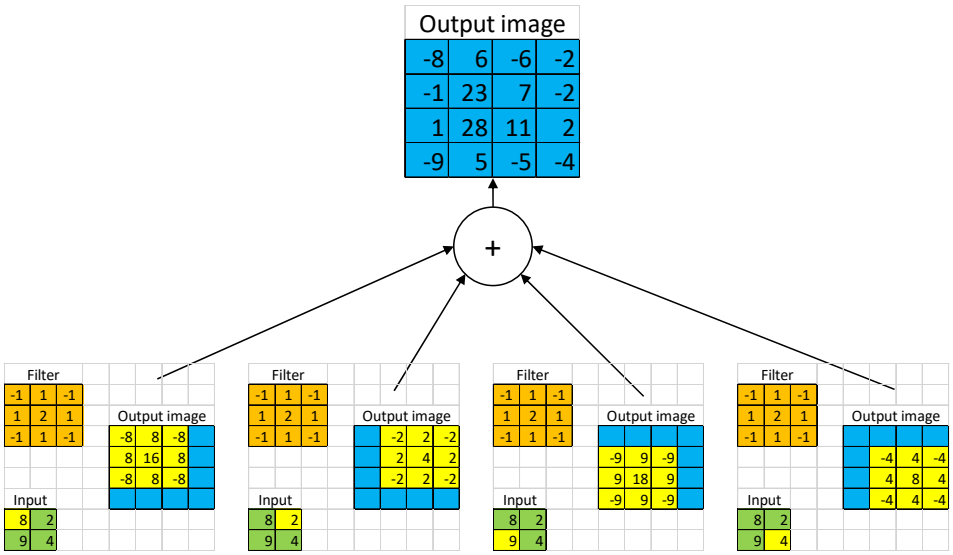


Figure 3. Process of up-sampling implemented by inverse convolution.

Given that we expressed the convolution by means of a matrix product, the deconvolution can be therefore implemented by simply setting the product matrix of the corresponding filter and then transpose it (Fig. 5). Flattening the input allows to reframe a consistent matrix multiplication obtaining an output vector which can be reshaped into the final output image.

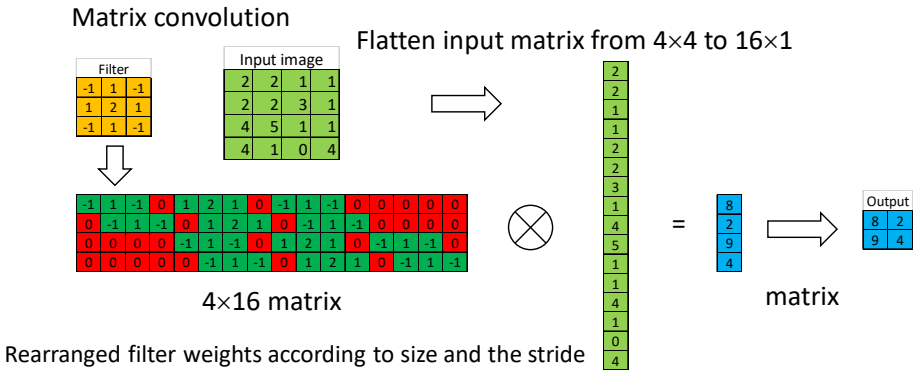


Figure 4. Convolution remapped into a matrix operation.

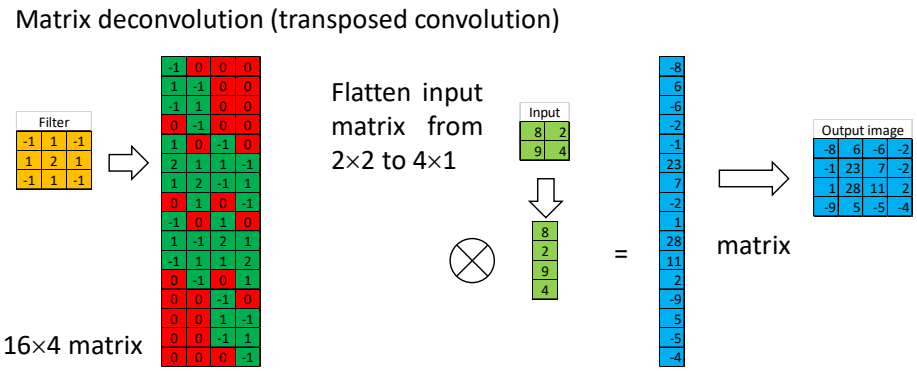


Figure 5. Deconvolution implemented by means of a transposed convolution.

In general, the paradigm of transposed convolution forms the same connectivity as the normal convolution but in the backward direction (one-to-many rather than many-to-one association). As such, the transposed convolution is not a 'direct' convolution.

The second approach emulates the transposed convolution by using a direct convolution exploiting zero-padding in the input image. For particular conditions, the direct convolution produces the same effect as the transposed convolution. Along with classical zero-padding, inter-pixel padding allows image up-sampling with a direct convolution. This is usually called fractional strided convolution where the size of the resulting output image must be computed as:

$$s_x = \frac{(I_x + 4z_x - f_x)}{L_x} + 1 \; s_y = \frac{(I_y + 4z_y - f_y)}{L_y} + 1$$

Figure 6. Traditional upsampling by zero-padded convolution and fractional strided convolution, where inter-pixel padding.

Encoder-decoder networks

As it has been said, the deconvolution network splits the up-sampling into two or more transposed convolutional layers, each accounting for doubling the spatial resolution. In agreement with the methodology traditionally exploited in the encoding convolutional layers, where the number of feature maps is progressively increased from the higher spatial scales up to lower ones in the deeper encoding layers, the deconvolution layered network involves the decreasing of the number of feature maps as the spatial scale increases, thus exactly mirroring the process in the encoding layer.

We are going to see now that encoding and decoding processing architectures can be joined together to express more challenging tasks. We have already seen in the previous class that the Autoencoder unit can be regarded as the simplest encoder-decoder network. As a matter of fact, it is composed by one encoder layer and one decoder layer, this last corresponding to the output layer. In this network, it has been understood that the decoding is mapping the encoding performed by the hidden layer into an output domain identical to the domain of the input. This simple architecture can be extended to a more general architecture for compression-decompression of information (Fig. 7), which is represented by two sequential paths, namely the encoder (yellow layers) and decoder (blue layers) and one output layer (purple) customized to the specific task the network has to be address. This kind of network, just called encoder-decoder network, not only might generate decoding into different domains with respect to that one of the input but also may exploit multiple layers in both encoding and decoding paths. In such a case, the encoding path will perform progressive compression of the input scale and extraction of the relevant features, whereas the decoding path will perform progressive decompression of the spatial scale, and integration of the features. The final encoding layer, called bottleneck of the network (green block in the Fig. 7), is the interface between encoded features and task-dependent decoding maps. This means that the ED network gets trained according to one specific task using supervised backpropagation. The network will learn the optimal weights of the encoding layers and the optimal weights in the decoding layers to fit the task (e.g. regression, classification, …).

In 2015, Ronneberger et al. proposed, in the field of image segmentation, an evolution of the encoder-decoder network they called Unet. The graph of the network presented in the original paper is shown in figure 8, where it is clearly visible why the model was named after the letter “U”. This model is takes its root from the previous E-D architecture implemented as an end-to-end fully convolutional network, where just convolutional and pooling layers are present, without any dense layer. The first specificity of the Unet proposal consists of two 3x3 convolutions embedded one after the other one, each followed by ReLU activation function. Then, a 2x2 max pooling operation with strides of 2 is added to halve the size of the input.

At each down-sampling step, the number of filters used to perform the convolutions is doubled, to enlarge the space of features to be extracted from the data. The up-sampling path performs the opposite operations: transposed convolutions are applied to enlarge the spatial representation of data and gradually increase image size, in order to recover the original dimension of the input. Feature maps coming from deep layers represent abstract attributes and high-level information. These are up sampled step after step, and concatenated with feature maps generated at the same level in the down sampling branch. The second specificity of the Unet consists just of concatenation of layers in the decoding path exploiting activation maps coming from the encoding layers. Graphically, skip connections put in relation the corresponding levels in the encoding and decoding paths. The rationale of this solution is to add consistent information to the up-sampling to better constraint the deconvolution operation. In fact, the skip connections from encoding to decoding path allow to merge global contextual information, coming from deeper layers, with the spatial high resolution information, present in the decoding path at each level. This enables the network to produce fine-grained segmentation maps at the output. After concatenation, two 3x3 convolutions are performed to unify the information. The network output is a Softmax layer according to the number of labels segmented into the image (in the original work this was two corresponding to a binary segmentation i.e. label and background).

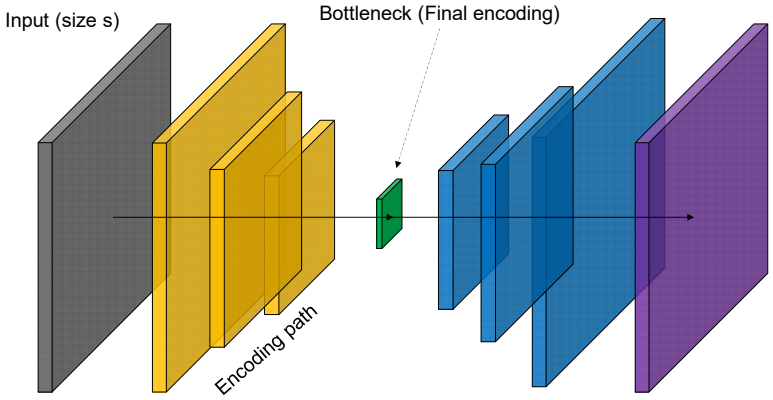


Figure 7. Diagram of basic encoder-decoder architecture. The final encoding layer is usually called bottleneck.

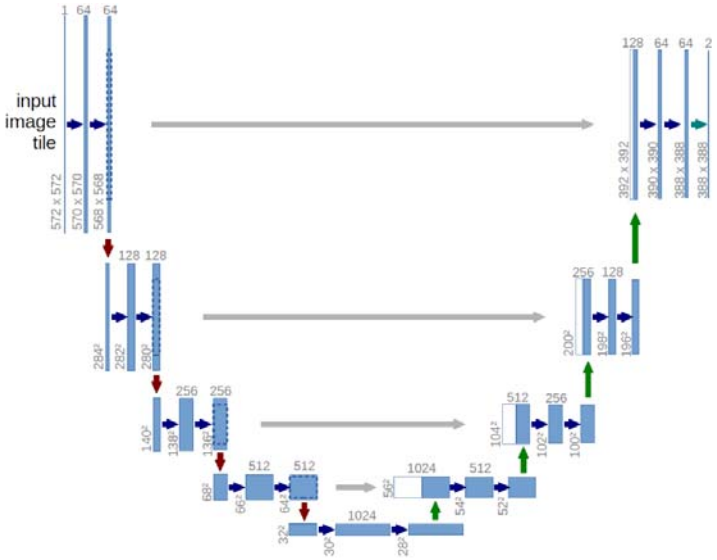
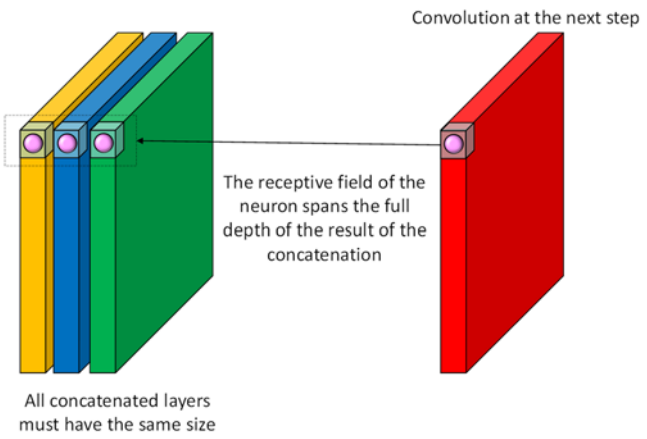


Figure 8. Diagram of Unet architecture as presented in the original paper (Ronneberger et al. 2015). Blue horizontal arrows stand for 3x3 convolutions + ReLU activations. Downward and upward arrows represent respectively 2x2 max pooling and 2x2 transposed convolutions. Grey long horizontal arrows represent skip connections to integrate feature maps from the encoding branch to the decoding branch. The overall network has four encoding and four decoding layers connected by the bottleneck, embedding 1024 feature maps. The number of feature maps in the encoding layers is: 64, 128, 256 and 512, mirrored in the decoding layer.

In the next picture, the concatenation is graphically represented where the receptive field of one neuron in the convolutional layer (red) spans the overall depth of the concatenated layers, which are all characterized by the same size (each layer can actually have however a different number of feature maps. The Unet architecture can be extended to process 1D, 2D and 3D signals. In this last case, full 3D set of images (e.g. a clinical volumetric scan) may be the input to the network and the Unet will be characterized by 3D convolutions. According to the task to be address by the network (e.g. segmentation, super-resolution, noise-removal, ..) the output layer of the network is to be set properly (Fig. 9). 2D and 3D Unet models have been largely proposed to segment both anatomical regions and lesion such as tumors in diagnostic images.



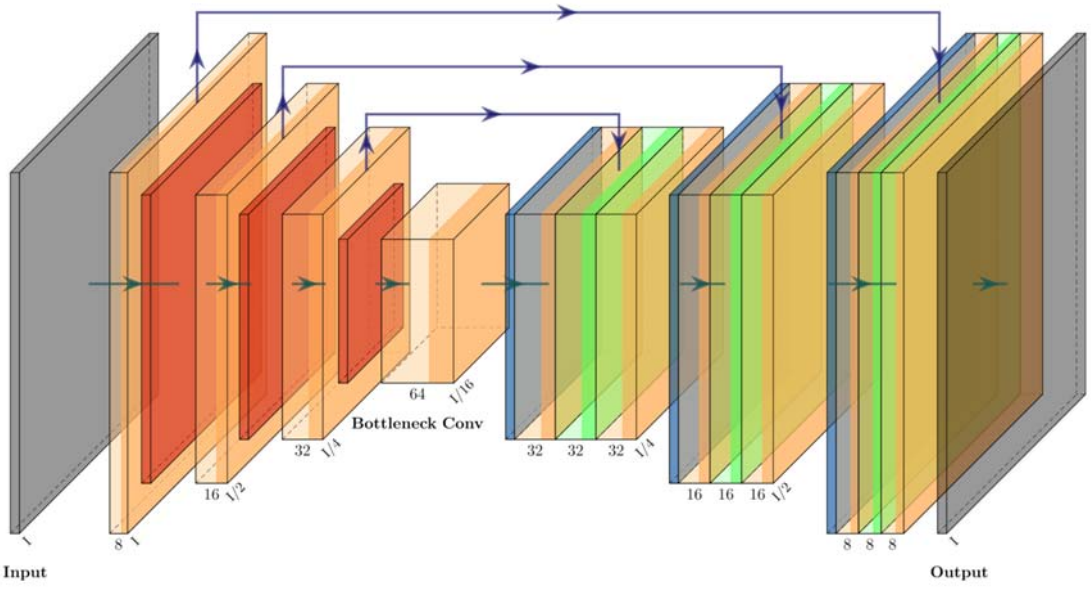


Figure 9. Architecture of one 3D Unet to process an overall 3D volume in the input. The first level of the encoder is composed by one convolutional (light orange) with 8 feature maps, one Relu activation (dark orange) and one max pooling layer (red) for down-sampling. Progressively the encoding scale down by a factor of 2 the spatial scale while doubling the number of feature maps. In the decoding layer, afterwards the deconvolution (blue) layer, the output is concatenated to the layer coming from the encoding path (skip connection) and then further filtered by a following convolutional layer.

An example is the segmentation of bony region in the knee to reconstruct femur and tibial surfaces, usable in the surgical planning for joint replacement. In this case the input is a 3D volume corresponding to a CT scans of the knee and the output is the segmented volume extracting distal femur and proximal tibia (Fig. 10). Technical, the segmentation is obtained by enabling a Softmax layer in the output so that the segmentation is reframed as a classification task. Considering a Unet for image segmentation, the Softmax layer will be conceived as a regular grid of neurons, with the same size of the input image, one grid for each label to be segmented. Assuming to detecting one label in the $m \times n$ image, the Softmax will have size of $m \times n \times 2$ where the first dimension will encode the membership of each pixel to the background (no label) while the second one will encode the membership of each pixel to the label. One pixel therefore will have a probability to belong to class background and one probability to belong to class label, being clearly the sum of the two probabilities equal to one. This can be generalized when dealing with multiple labels. Similarly, for 3D volume segmentation, the Softmax layer will be characterized by a size of $m \times n \times s \times c$, with m, n, and s are the number of rows, columns and slices of the volume, whereas c is the number of different aspects (organs,

lension, …) to segment into the image. The network in Fig. 9, devoted to knee bone segmentation in CT volumes, consists of a feature map schema of 8-16-32-64-32-16-8 (filter size: 3×3×3), distributed over three main symmetric encoding-decoding layers, and has 351435 parameters. Assuming a volume of 128×128×128 the network output will be 128×128×128×3, the first label being the background, the second one being the femur and the third one being the tibia. A detail of 8 feature maps from the first and the last convolutional layers inside a Unet architecture is presented in Fig. 11.

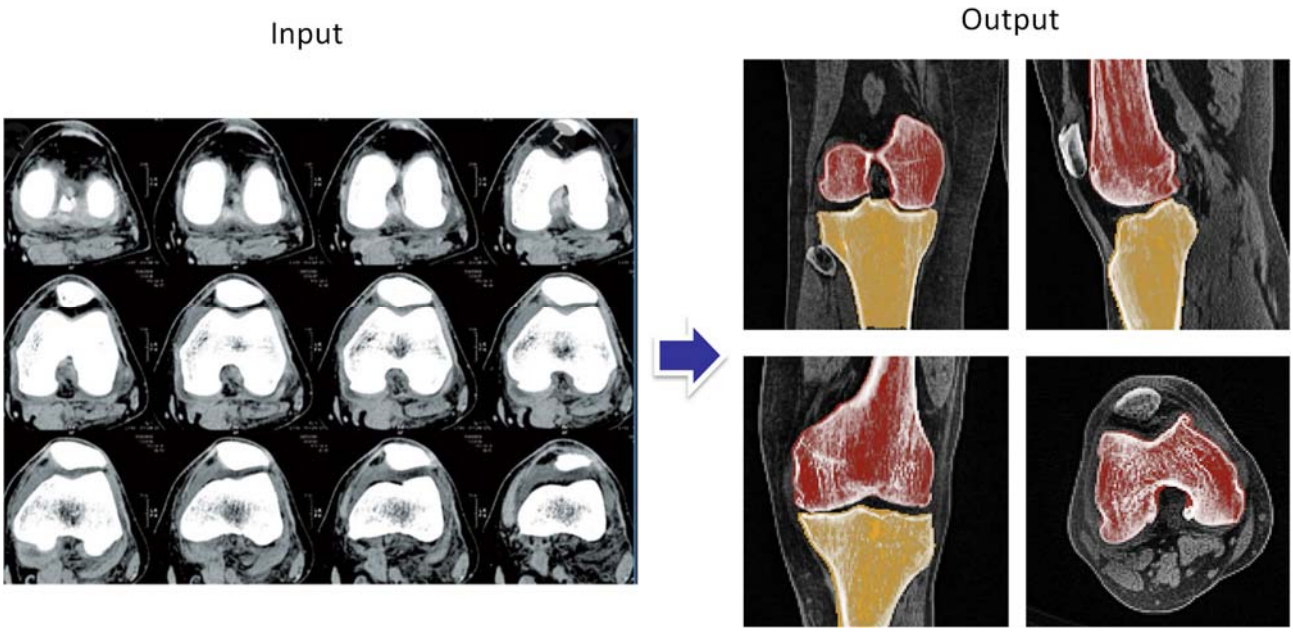


Figure 10. CT scan in the input to the 3D Unet is progressively encoded and then decoded to semantically segment the distal femur and the proximal tibia.

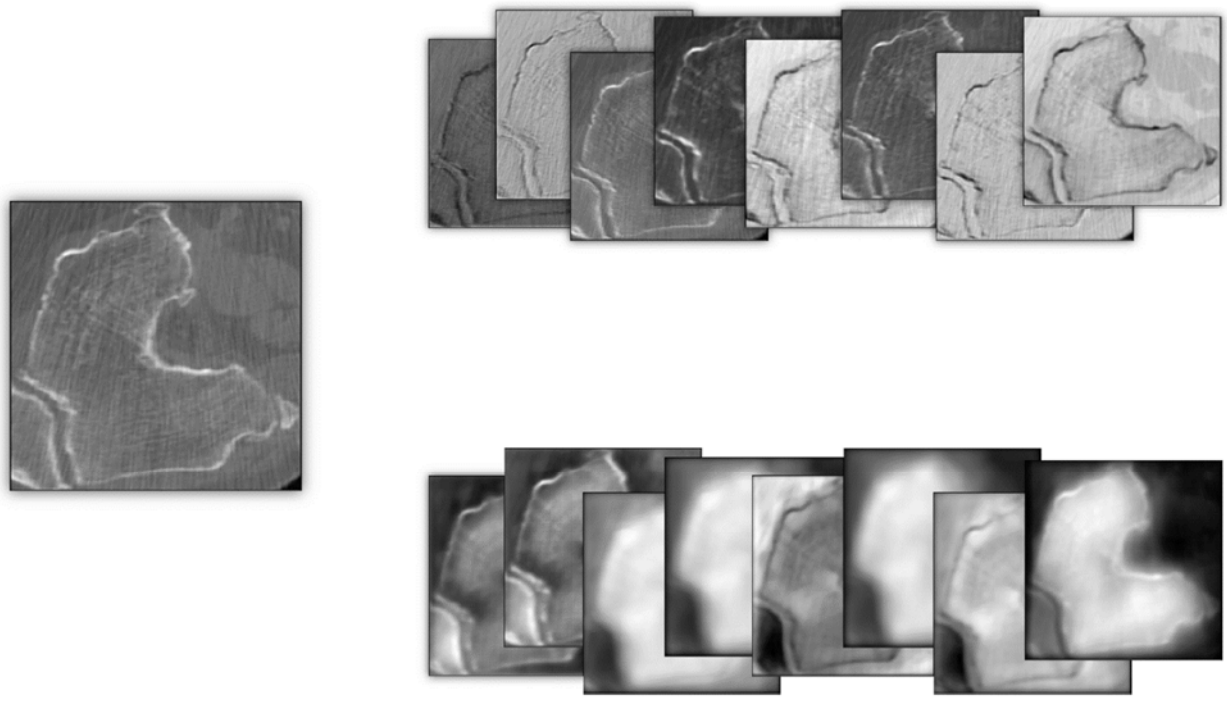


Figure 11. Example of 8 feature maps from the first and the last convolutional layers inside a Unet architecture.