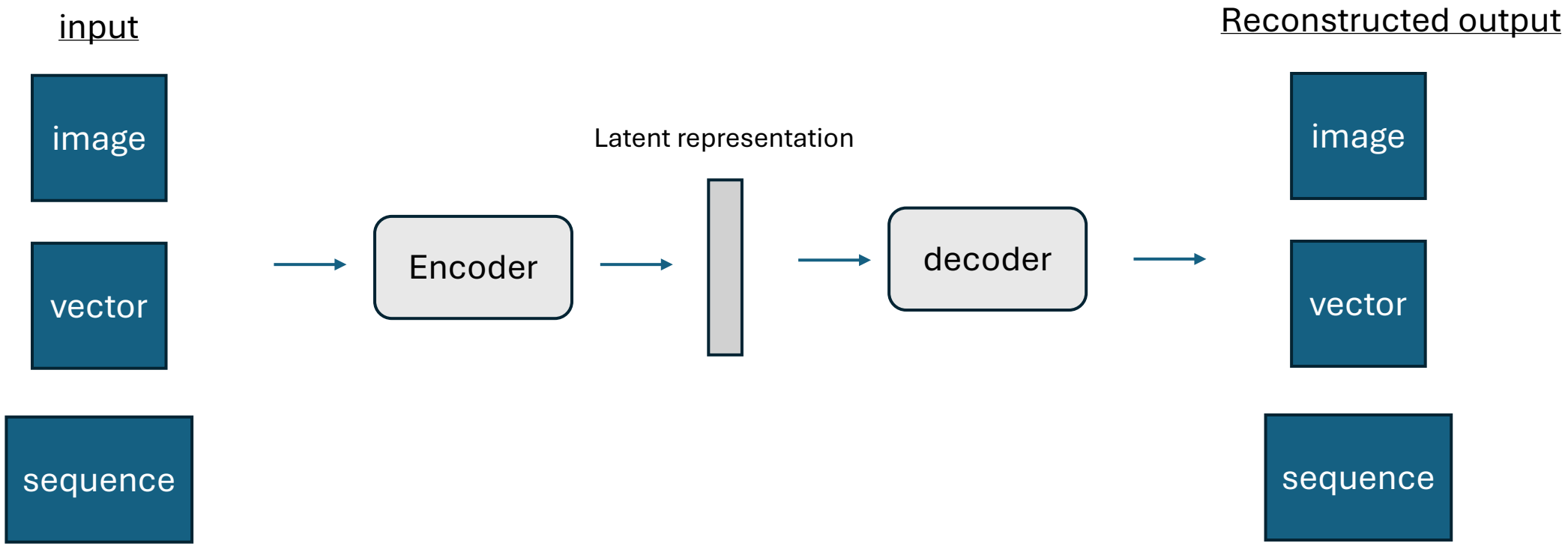
Machine Learning for Robotics: **Encoder-Decoder Architectures**

Prof. Navid Dadkhah Tehrani

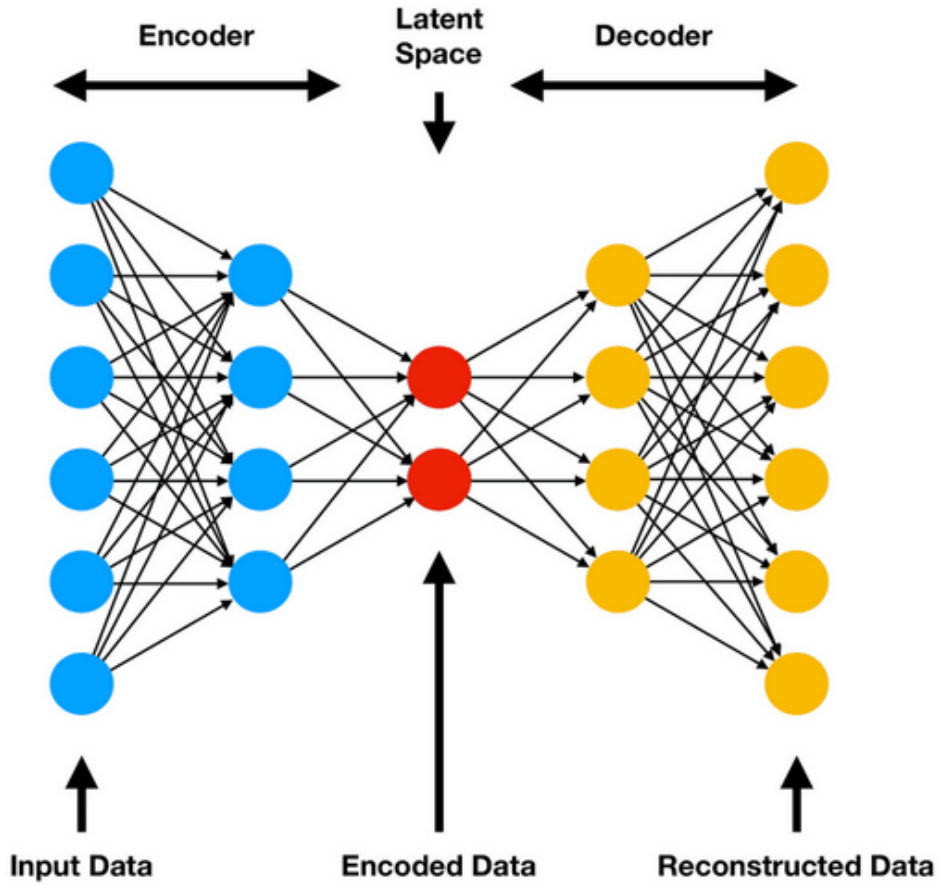
Autoencoders- main idea



MLP autoencoders

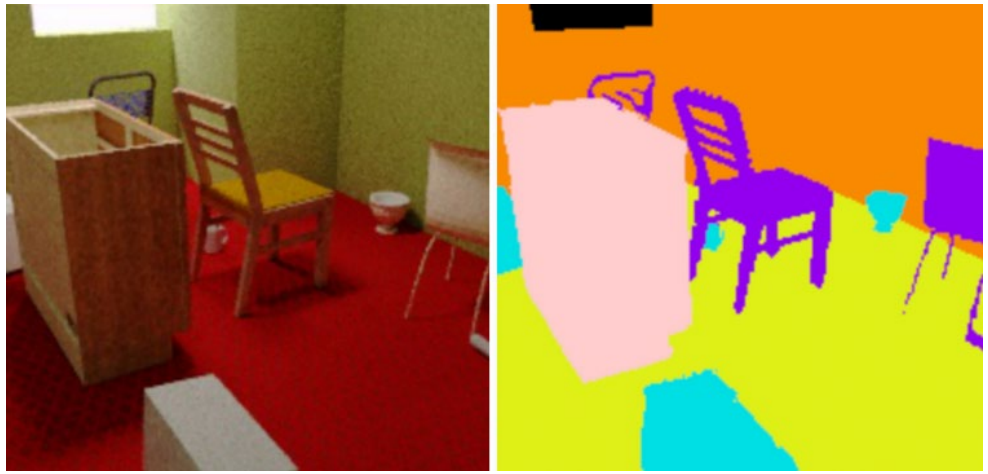
Some applications:

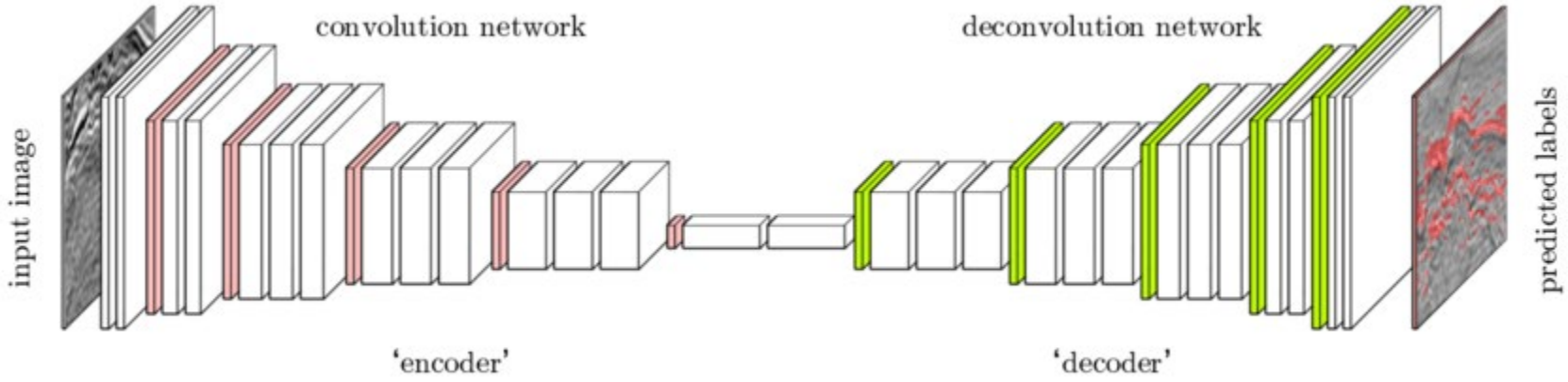
- Dimensionality reduction.
- Anomaly detection: when the outlier is entered the model, it results in high reconstruction error.
- Control allocation



CNN autoencoders- Semantic Segmentation

Sematic segmentation: To label each pixel in an image with a class. As opposed to label the entire image.





[1] V. Badrinarayanan, et. al, SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation, 2015
[2] O. Ronneberger, P, Fischer, T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation", 2015

Classes:

1. Pedestrian
2. Car
3. Traffic light
4. road
5. Train
6. building

Ground truth →

Q: How is the output of the NN is generated?

A: the output of the NN is a tensor of B*N*H*W where B is batch size, N is the number of classes.
each pixel value in the tensor corresponds to a class probability.
For example, assuming B=1:

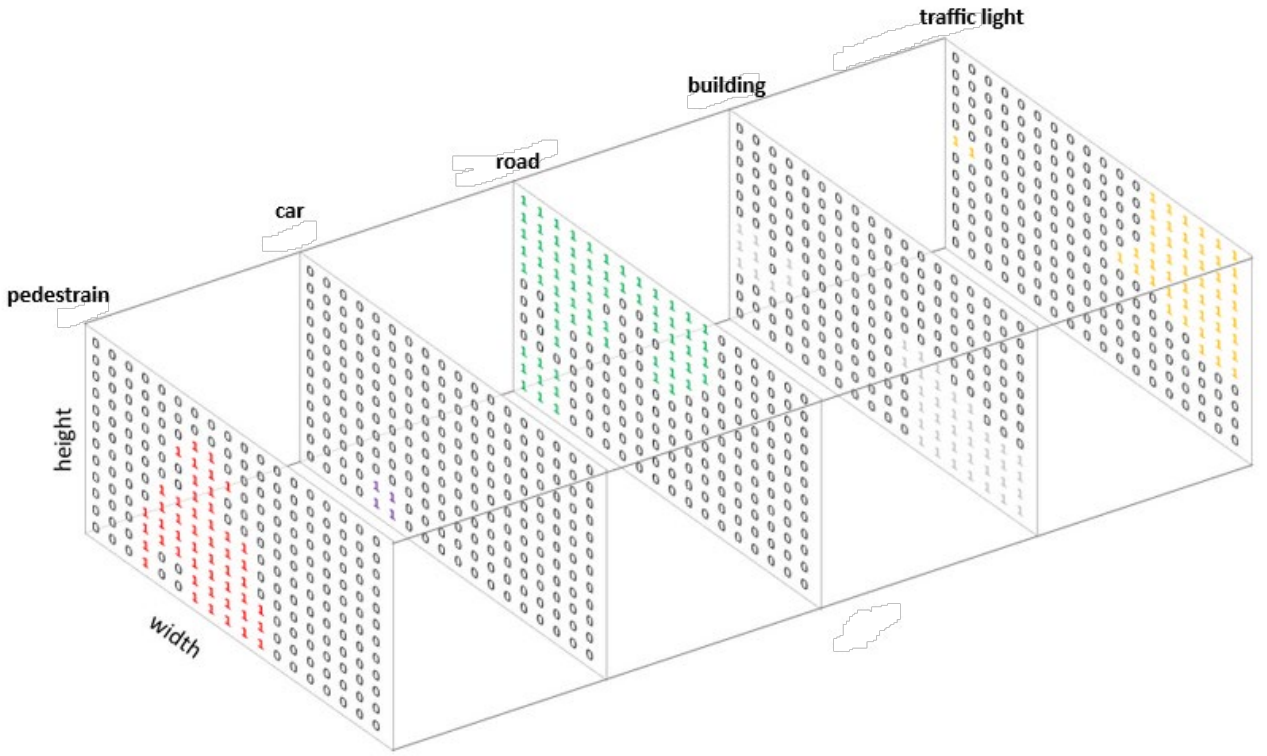
pixel (1,1,i,j)→ probability of pixel belong to pedestrian→ 0.5 -------- ---------- 1
pixel (1,2,i,j)→ probability of pixel belong to car→ 0.1 ---------- 0
pixel (1,3,i,j)→ probability of pixel belong to traffic light→0.05 ---------- 0
pixel (1,4,i,j)→ probability of pixel belong to road→ 0.1 ----------0
pixel (1,5,i,j)→ probability of pixel belong to train→ 0.2 ----------0
pixel (1,6,i,j)→ probability of pixel belong to building→ 0.05 ----------0

Apply argmax

Q: How is the loss function?

A: Softmax Cross-Entropy loss averaged over all the pixels.

This is what we get after applying argmax for each channel.



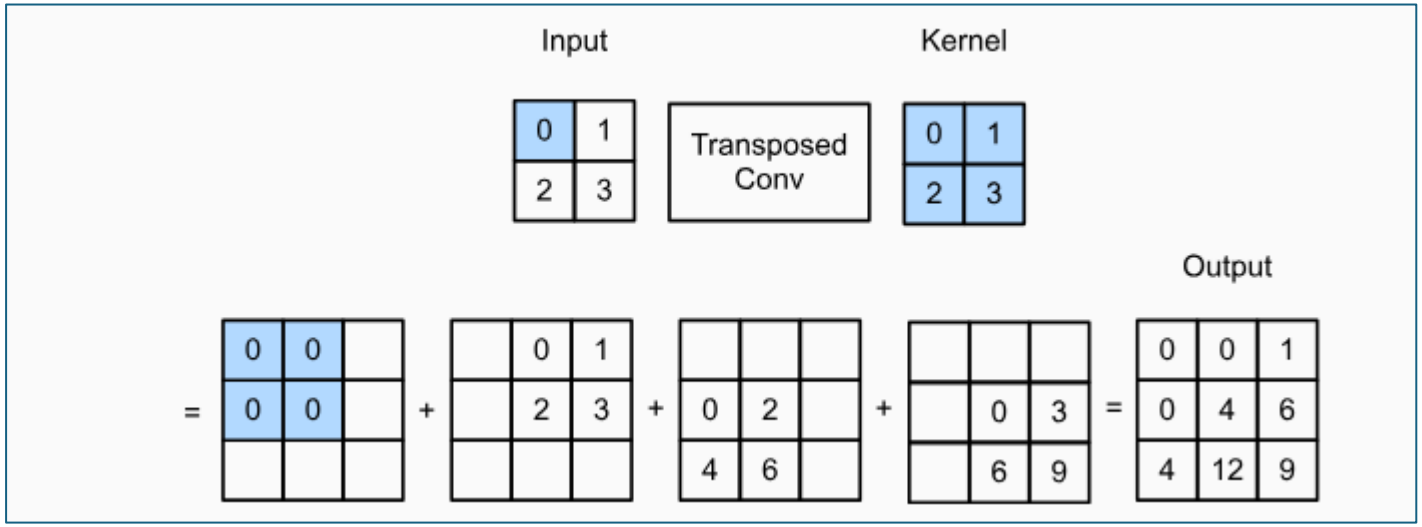
Transposed Convolution/ Deconvolution/Inverse convolution

We saw that in a CNN, convolutional layers (with stride>=2) reduce the height and width of the output image/feature map.

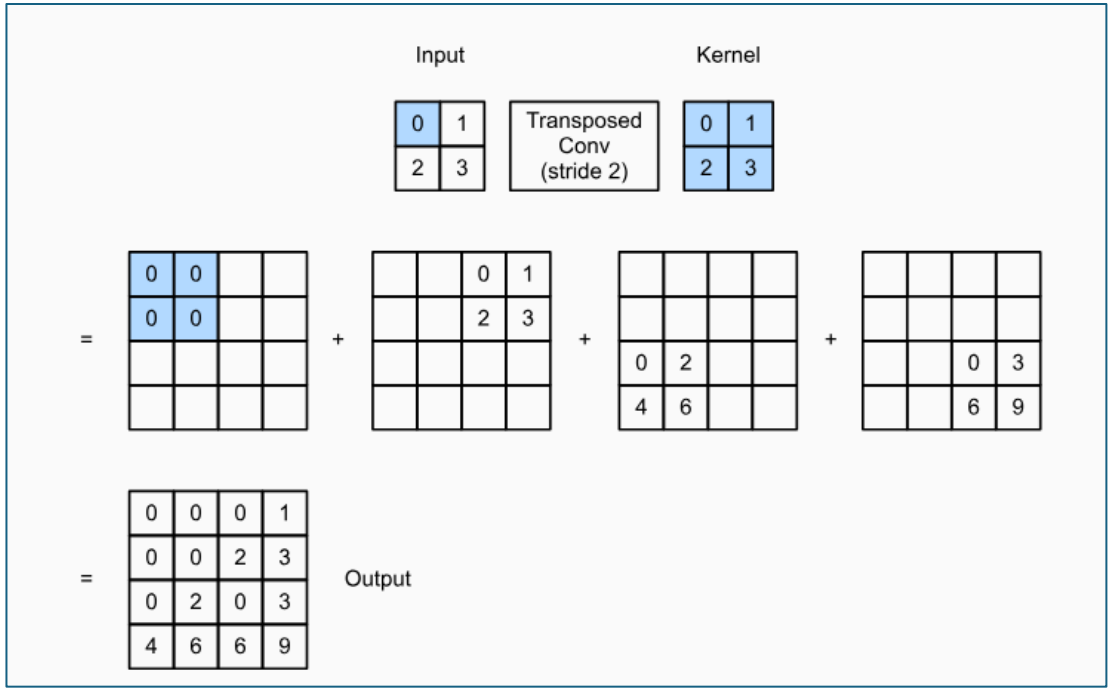
To reverse that, i.e. to go from smaller size image/feature map to larger one, we use transposed convolution.

Transpose conv. with 2-by-2 kernel

Stride = 1 →



Stride = 2 →



$$output\_size = s * (input\_size - 1) + k - 2 * p$$

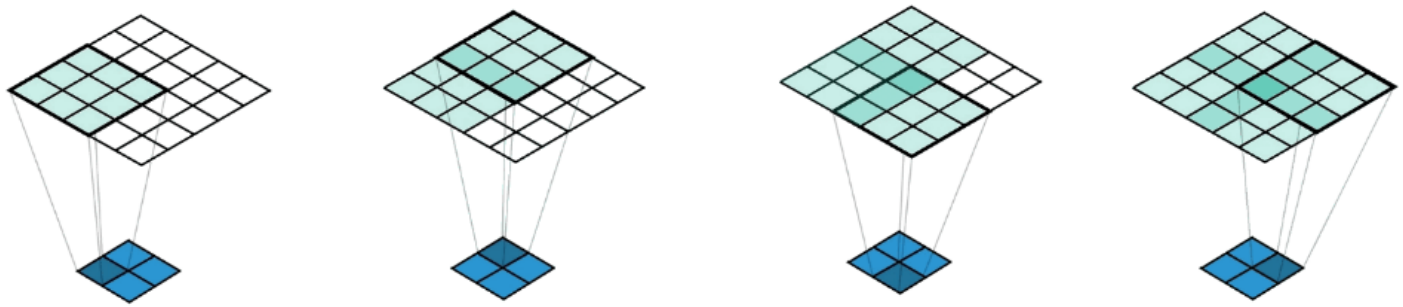
s: $stride$

k: $kernel\ size$

p: $padding\ size$

Beware of the checkerboard artifacts in transpose convolutions

Example: Transposed Conv. With 3-by-3 kernel and stride =2



To avoid the checkerboard here, we should use kernel size of 2-by-2.

In pytorch:

ConvTranspose2d

CLASS torch.nn.ConvTranspose2d(*in_channels*, *out_channels*, *kernel_size*, *stride=1*, *padding=0*, *output_padding=0*, *groups=1*, *bias=True*, *dilation=1*, *padding_mode='zeros'*, *device=None*, *dtype=None*) [SOURCE]

Applies a 2D transposed convolution operator over an input image composed of several input planes.

Up-sampling

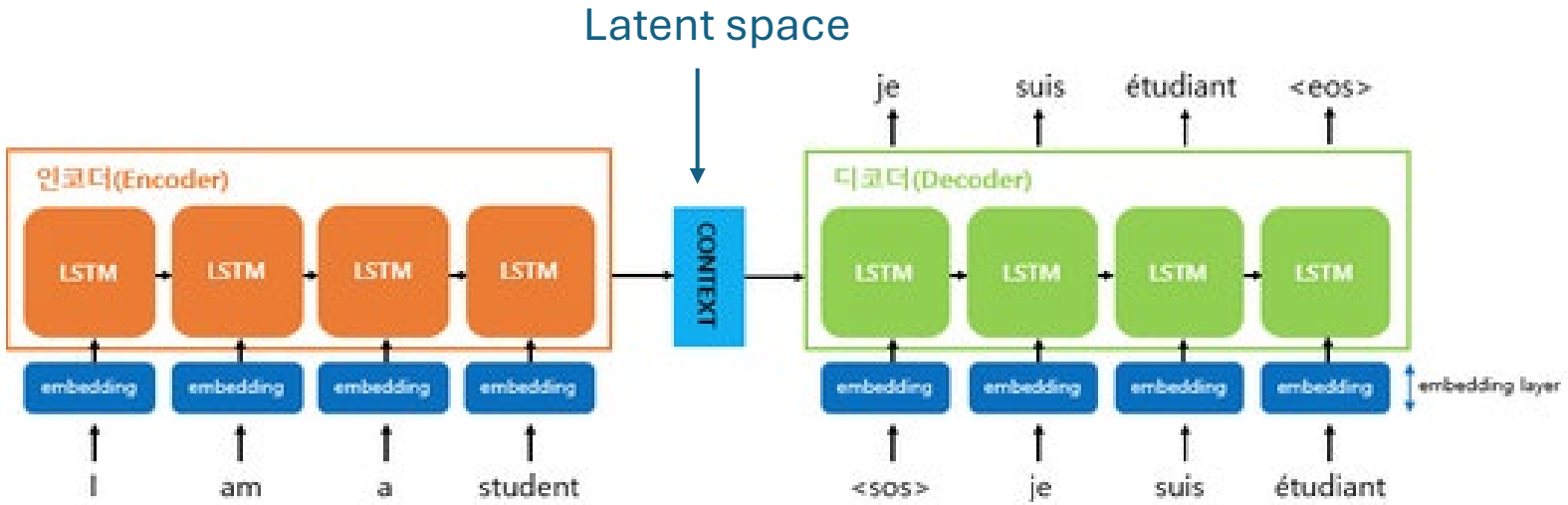
Up-sampling is another way to increase the H/W of the channel.
Up-sampling can be thought of as the inverse of pooling.
Like pooling, up-sampling does not have learnable weights. So transposed convolution might be more useful.

To add learnable weights to up-sampling we can
add a conv layer after up-sampling operation (up-sampling followed by a conv layer).

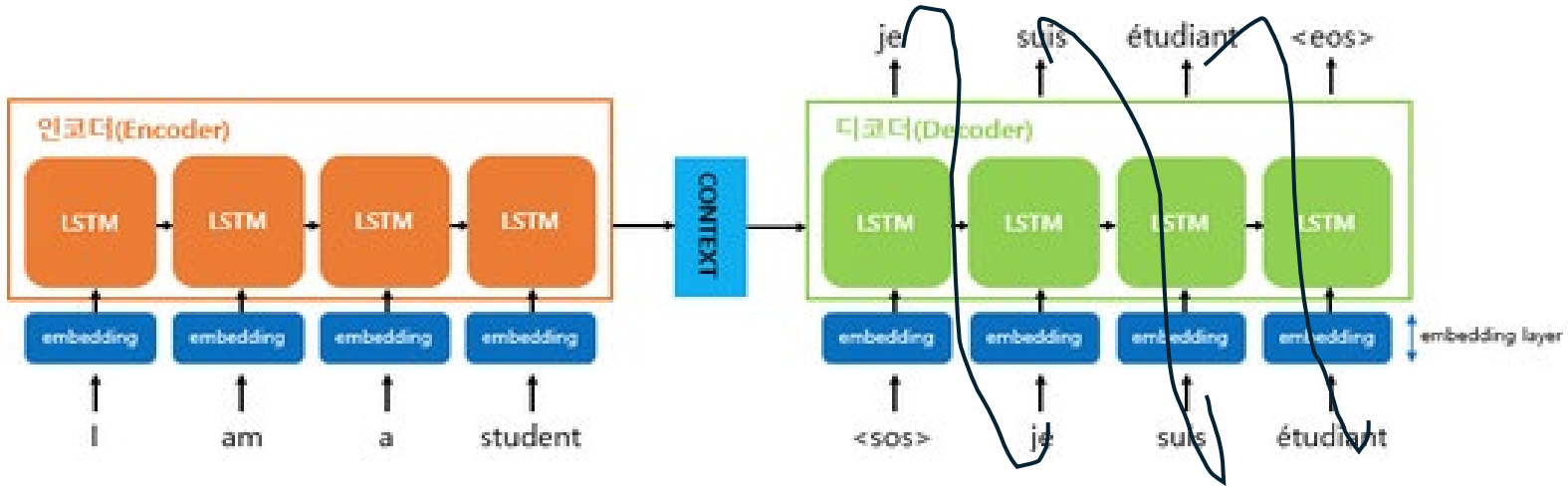
```
>>> input = torch.arange(1, 5, dtype=torch.float32).view(1, 1, 2, 2)
>>> input
tensor([[[[1., 2.],
          [3., 4.]]]])

>>> m = nn.Upsample(scale_factor=2, mode='nearest')
>>> m(input)
tensor([[[[1., 1., 2., 2.],
          [1., 1., 2., 2.],
          [3., 3., 4., 4.],
          [3., 3., 4., 4.]]]])

>>> m = nn.Upsample(scale_factor=2, mode='bilinear')  # align_corners=False
>>> m(input)
tensor([[[[1.0000, 1.2500, 1.7500, 2.0000],
          [1.5000, 1.7500, 2.2500, 2.5000],
          [2.5000, 2.7500, 3.2500, 3.5000],
          [3.0000, 3.2500, 3.7500, 4.0000]]]])
```

RNN encoder-decoder for text translation

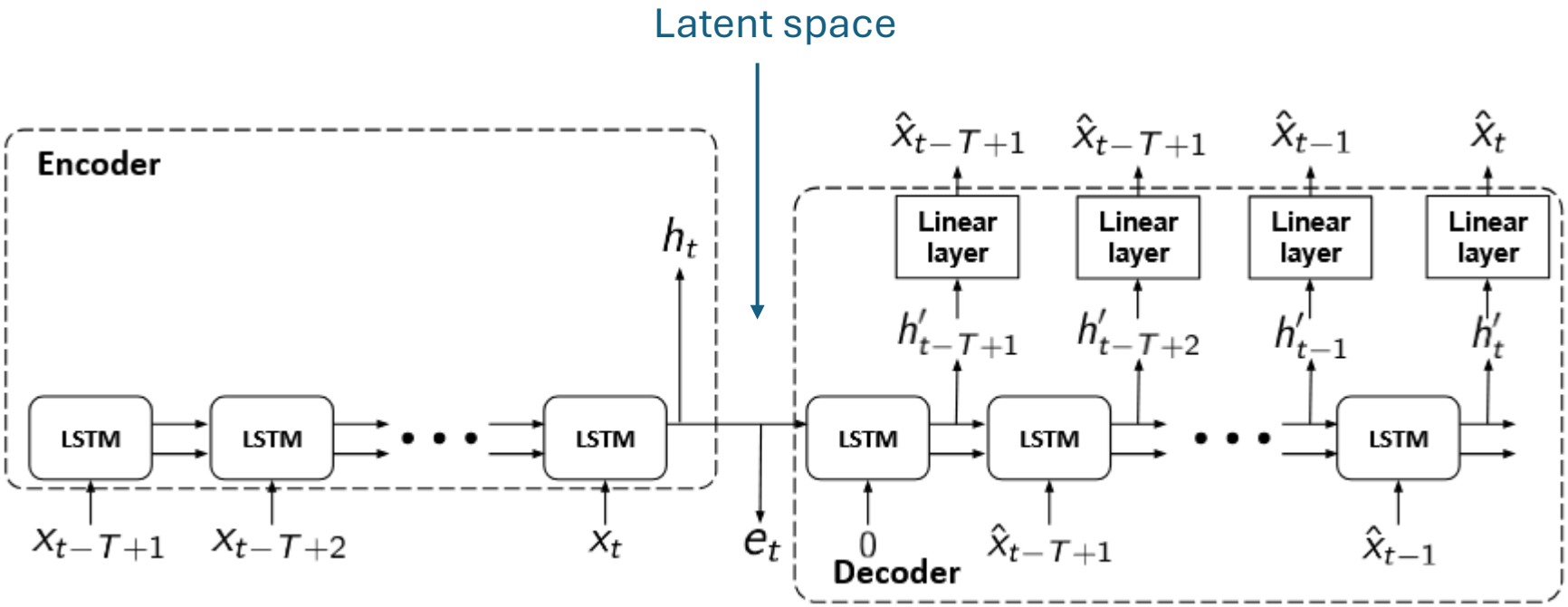
During training



During evaluation



RNN autoencoders



Ref: X. Lyu et. al., Improving Clinical Predictions through Unsupervised Time Series Representation Learning, 2018

Other Applications of Autoencoders

- Image denoising: Denoising autoencoders are trained to reconstruct clean images from noisy inputs.
- Image generation:
  can generate new data that is similar to the input data by sampling from the learned latent space.
  This concept it called Variational Autoencoder (VAE) :
  D. Kingma, M. Welling, “An Introduction to Variational Autoencoders”, 2019.
  (excellent 89-page article!)
  VAE is also an important component of stable diffusion models that’s used in generative AI.
- Monocular depth estimation