Autoencoders

AI+X
INSTITUTE FOR ARTIFICIAL INTELLIGENCE

What is an Autoencoder?

- A type of neural network used for learning compressed encodings of input data.
- There are 2 parts to an autoencoder: (1) Encoder and (2) Decoder
- Encoder (g): Takes the input data (x) and compresses it into a smaller form. Let’s call this smaller form, z = g(x).
- Decoder (f): Takes our compressed data, z = g(x), and tries to decode it to as close as possible as the original input x.
- We’ll call the re-generated input made by the decoder, x'=f(z).

What is an Autoencoder?

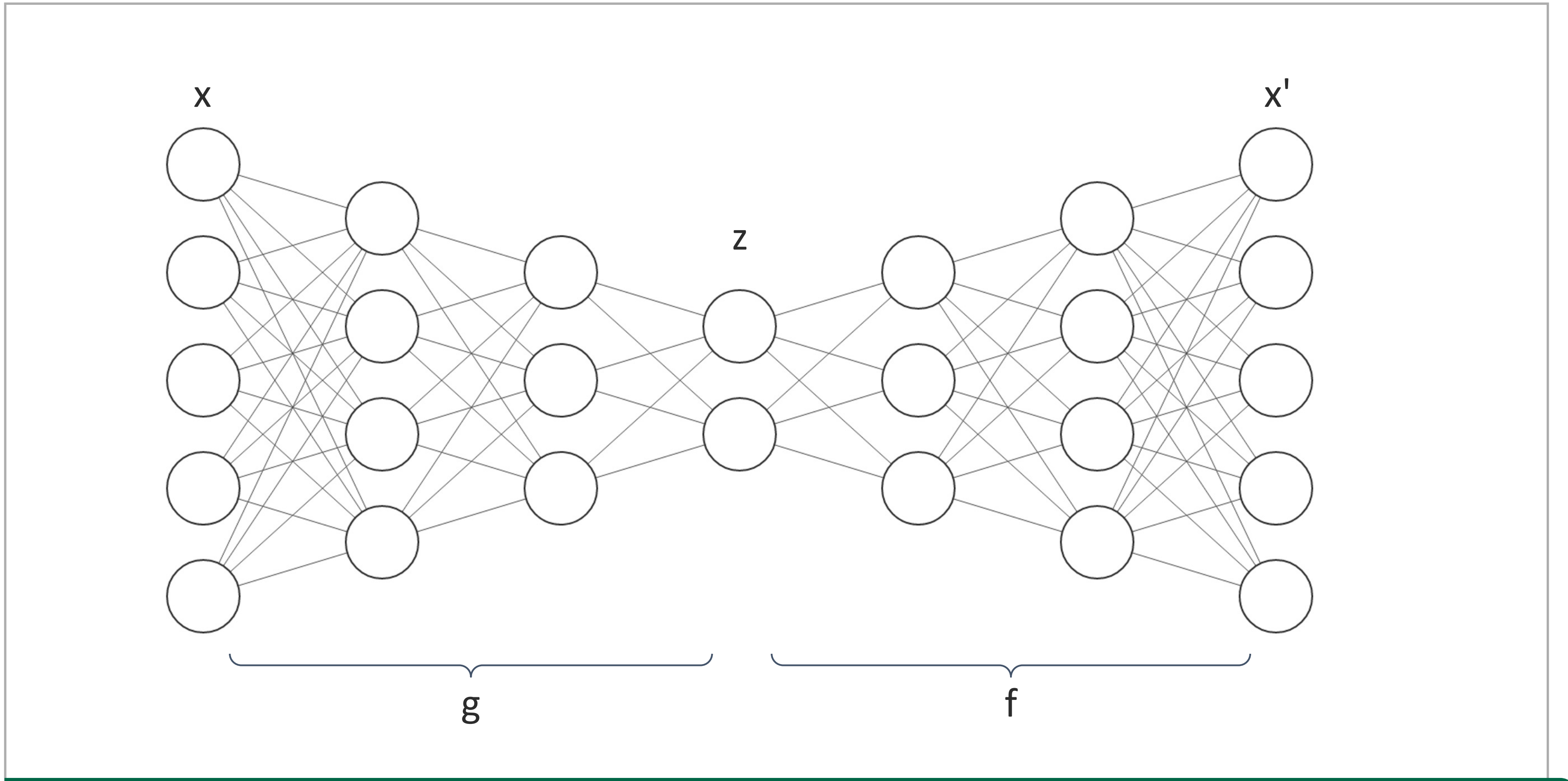


Figure 1. Feed forward autoencoder.

Knowledge Check 1

Which of the following methods can be used for dimensionality reduction?

I) PCA II) LDA III) SVM IV) Autoencoder

A I, II and III

B I, III and IV

C I, II and IV

D II, III and IV

What is an Autoencoder?

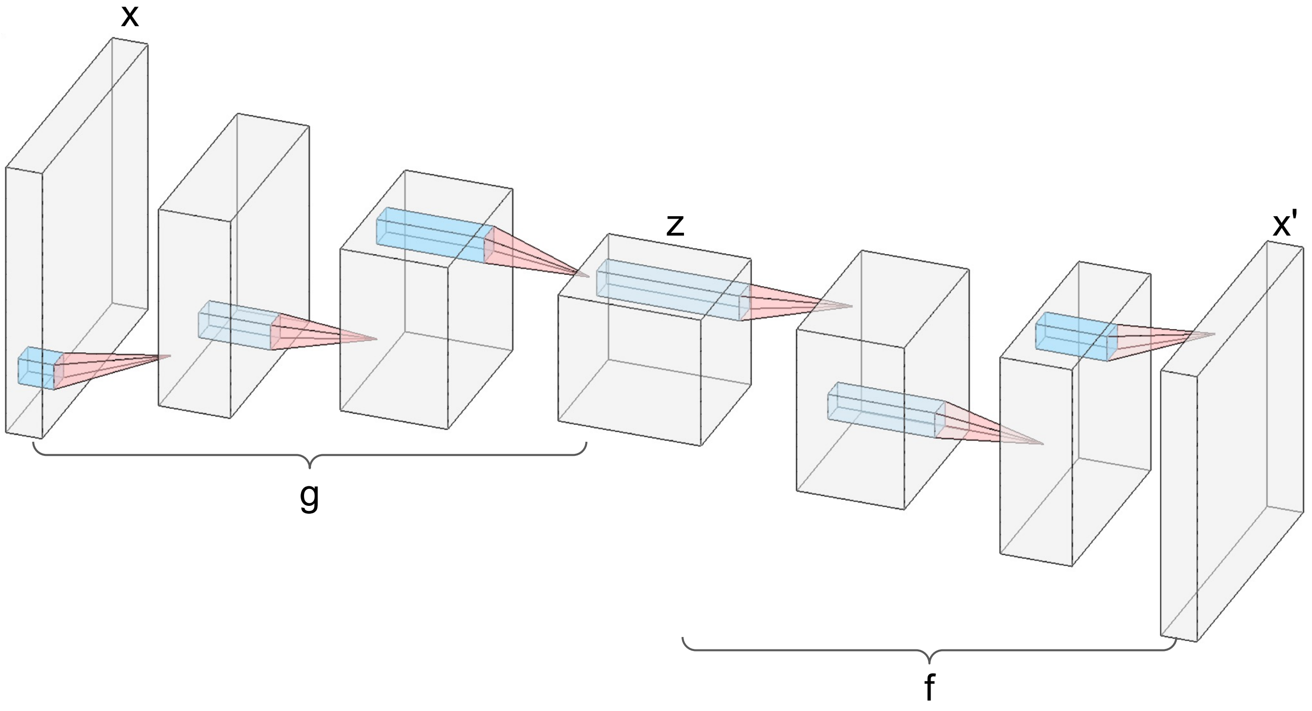


Figure 2. Convolutional autoencoder.

What are the Uses of Autoencoders?

- Data Compression: For example, you can compress images into a smaller format.
- Dimensionality reduction - reducing the number of features from d to $k < d$. You can then use this smaller group of features for your supervised learning task, or you could use them for visualization if $k=1,2$, or 3.
- Denoising: Removing “noise” from data.
- Unsupervised network initialization

Autoencoder Application: Feature Extraction

- You can use the “latent space representation” learned in the autoencoder for a different task such as classification.
- Step 1: Remove the decoder
- Step 2: Put your new layers for classification after the latent space, or “bottleneck” layer.
- Step 3: You can now perform your classification task.

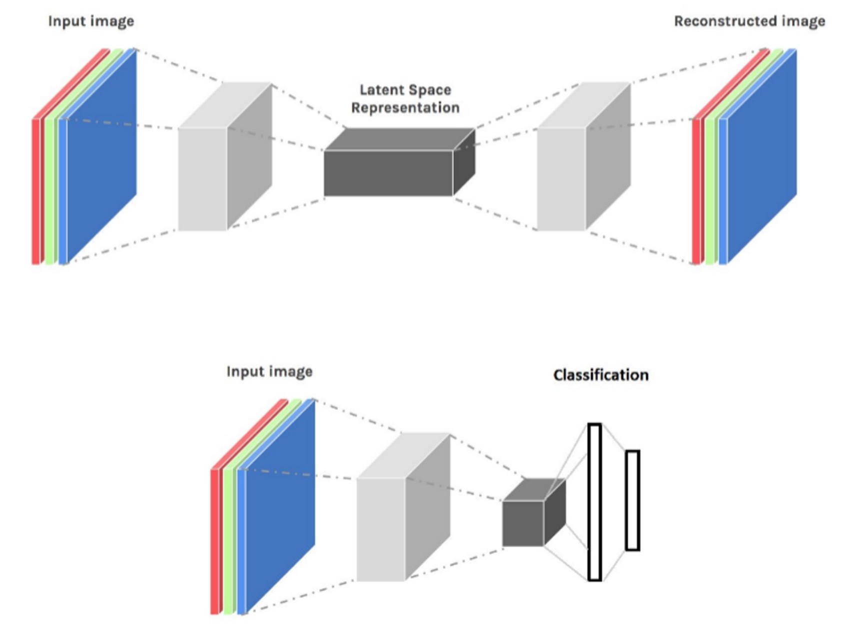


Figure 3. Autoencoder application: Feature extraction.

Face Autoencoder

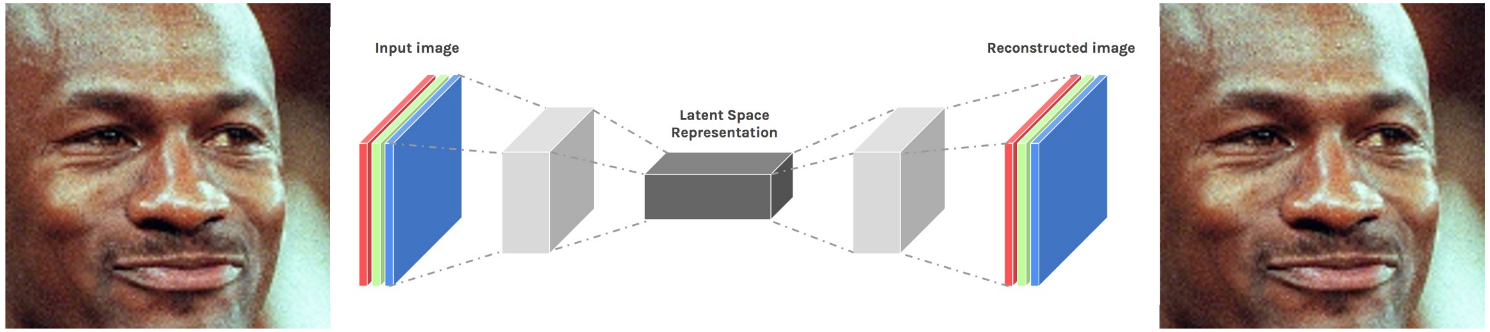
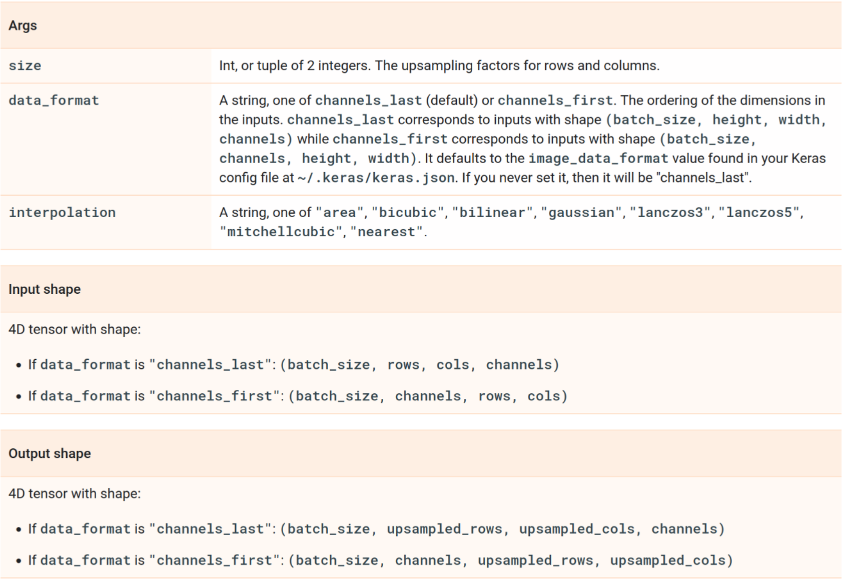


Figure 4. Face autoencoder.

Unsampling in Keras

https://www.tensorflow.org/api_docs/python/tf/keras/layers/UpSampling2D

```
tf.keras.layers.UpSampling2D(
                size=(2, 2),
        data_format=None,
        interpolation='nearest',
        **kwargs
)
```



Args	
`size`	Int, or tuple of 2 integers. The upsampling factors for rows and columns.
`data_format`	A string, one of `channels_last` (default) or `channels_first`. The ordering of the dimensions in the inputs. `channels_last` corresponds to inputs with shape `(batch_size, height, width, channels)` while `channels_first` corresponds to inputs with shape `(batch_size, channels, height, width)`. It defaults to the `image_data_format` value found in your Keras config file at `~/.keras/keras.json`. If you never set it, then it will be "channels_last".
`interpolation`	A string, one of `"area"`, `"bicubic"`, `"bilinear"`, `"gaussian"`, `"lanczos3"`, `"lanczos5"`, `"mitchellcubic"`, `"nearest"`.

Input shape

4D tensor with shape:

- If `data_format` is `"channels_last"`: `(batch_size, rows, cols, channels)`
- If `data_format` is `"channels_first"`: `(batch_size, channels, rows, cols)`

Output shape

4D tensor with shape:

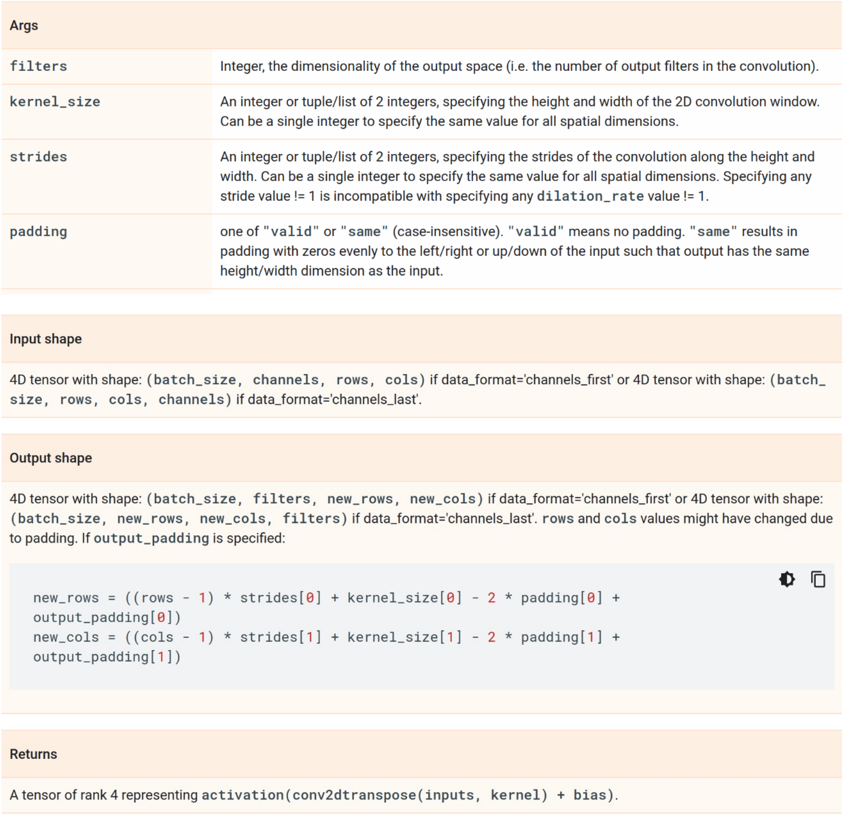
- If `data_format` is `"channels_last"`: `(batch_size, upsampled_rows, upsampled_cols, channels)`
- If `data_format` is `"channels_first"`: `(batch_size, channels, upsampled_rows, upsampled_cols)`

Figure 5. Unsampling in Keras.

Transposed Convolutions in Keras

https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2DTranspose

```
tf.keras.layers.Conv2DTranspose(
        filters,
        kernel_size,
        strides=(1, 1),
        padding='valid',
        output_padding=None,
        data_format=None,
        dilation_rate=(1, 1),
        activation=None,
        use_bias=True,
        kernel_initializer='glorot_uniform',
        bias_initializer='zeros',
        kernel_regularizer=None,
        bias_regularizer=None,
        activity_regularizer=None,
        kernel_constraint=None,
        bias_constraint=None,
        **kwargs
)
```



Args	
`filters`	Integer, the dimensionality of the output space (i.e. the number of output filters in the convolution).
`kernel_size`	An integer or tuple/list of 2 integers, specifying the height and width of the 2D convolution window. Can be a single integer to specify the same value for all spatial dimensions.
`strides`	An integer or tuple/list of 2 integers, specifying the strides of the convolution along the height and width. Can be a single integer to specify the same value for all spatial dimensions. Specifying any stride value != 1 is incompatible with specifying any `dilation_rate` value != 1.
`padding`	one of `"valid"` or `"same"` (case-insensitive). `"valid"` means no padding. `"same"` results in padding with zeros evenly to the left/right or up/down of the input such that output has the same height/width dimension as the input.

Input shape

4D tensor with shape: `(batch_size, channels, rows, cols)` if data_format='channels_first' or 4D tensor with shape: `(batch_size, rows, cols, channels)` if data_format='channels_last'.

Output shape

4D tensor with shape: `(batch_size, filters, new_rows, new_cols)` if data_format='channels_first' or 4D tensor with shape: `(batch_size, new_rows, new_cols, filters)` if data_format='channels_last'. `rows` and `cols` values might have changed due to padding. If `output_padding` is specified:

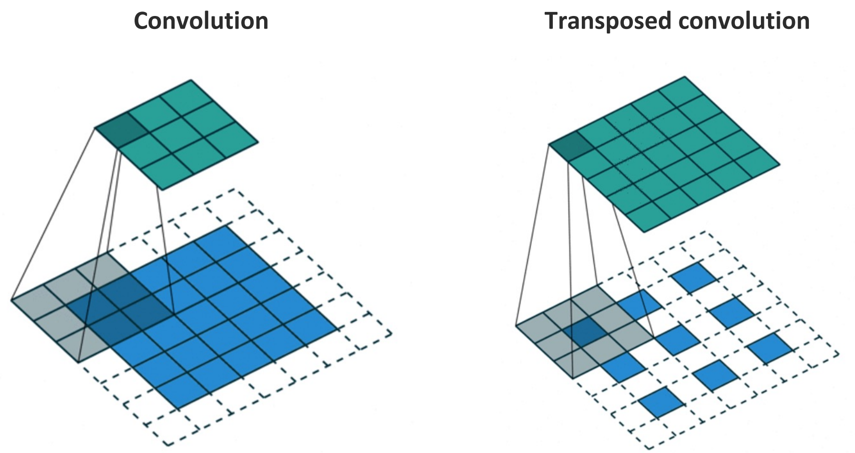
```
new_rows = ((rows - 1) * strides[0] + kernel_size[0] - 2 * padding[0] +
output_padding[0])
new_cols = ((cols - 1) * strides[1] + kernel_size[1] - 2 * padding[1] +
output_padding[1])
```

Returns

A tensor of rank 4 representing `activation(conv2dtranspose(inputs, kernel) + bias)`.

Figure 6. Transposed convolutions in Keras.

How Transposed Convolutions Work?



kernel_size: (3,3)
strides: (2,2)
padding: 'same'

kernel_size: (3,3)
strides: (2,2)
padding: 'same'

Figure 7. Transposed convolutions animations. N.B.: Blue maps are inputs, and cyan maps are outputs.

You have reached the end of the lecture.

Image/Figure References

Figure 3. Autoencoder application: Feature extraction.

Figure 4. Face autoencoder.

Figure 5. Unsampling in Keras. Source: https://www.tensorflow.org/api_docs/python/tf/keras/layers/UpSampling2D

Figure 6. Transposed complications in Keras. Source: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2DTranspose

Figure 7. Transposed convolutions animations. N.B.: Blue maps are inputs, and cyan maps are outputs. Source: https://github.com/vdumoulin/conv_arithmetic

Figure 8. Deepfakes with one-encoder & two-decoders. Source: Nguyen, T., Nguyen, C., Nguyen, D., Nguyen, D. & Nahavandi, S. (2020). Deep Learning for Deepfakes Creation and Detection: A Survey. arXiv abs/1909.11573.

Figure 9. Deepfakes with one-encoder & two-decoders. Source: Nguyen, T., Nguyen, C., Nguyen, D., Nguyen, D. & Nahavandi, S. (2020). Deep Learning for Deepfakes Creation and Detection: A Survey. arXiv abs/1909.11573.

Figure 10. Deepfakes with one-encoder & two-decoders. Source: Nguyen, T., Nguyen, C., Nguyen, D., Nguyen, D. & Nahavandi, S. (2020). Deep Learning for Deepfakes Creation and Detection: A Survey. arXiv abs/1909.11573.

Figure 11. Deepfakes with one-encoder & two-decoders. Source: Nguyen, T., Nguyen, C., Nguyen, D., Nguyen, D. & Nahavandi, S. (2020). Deep Learning for Deepfakes Creation and Detection: A Survey. arXiv abs/1909.11573.