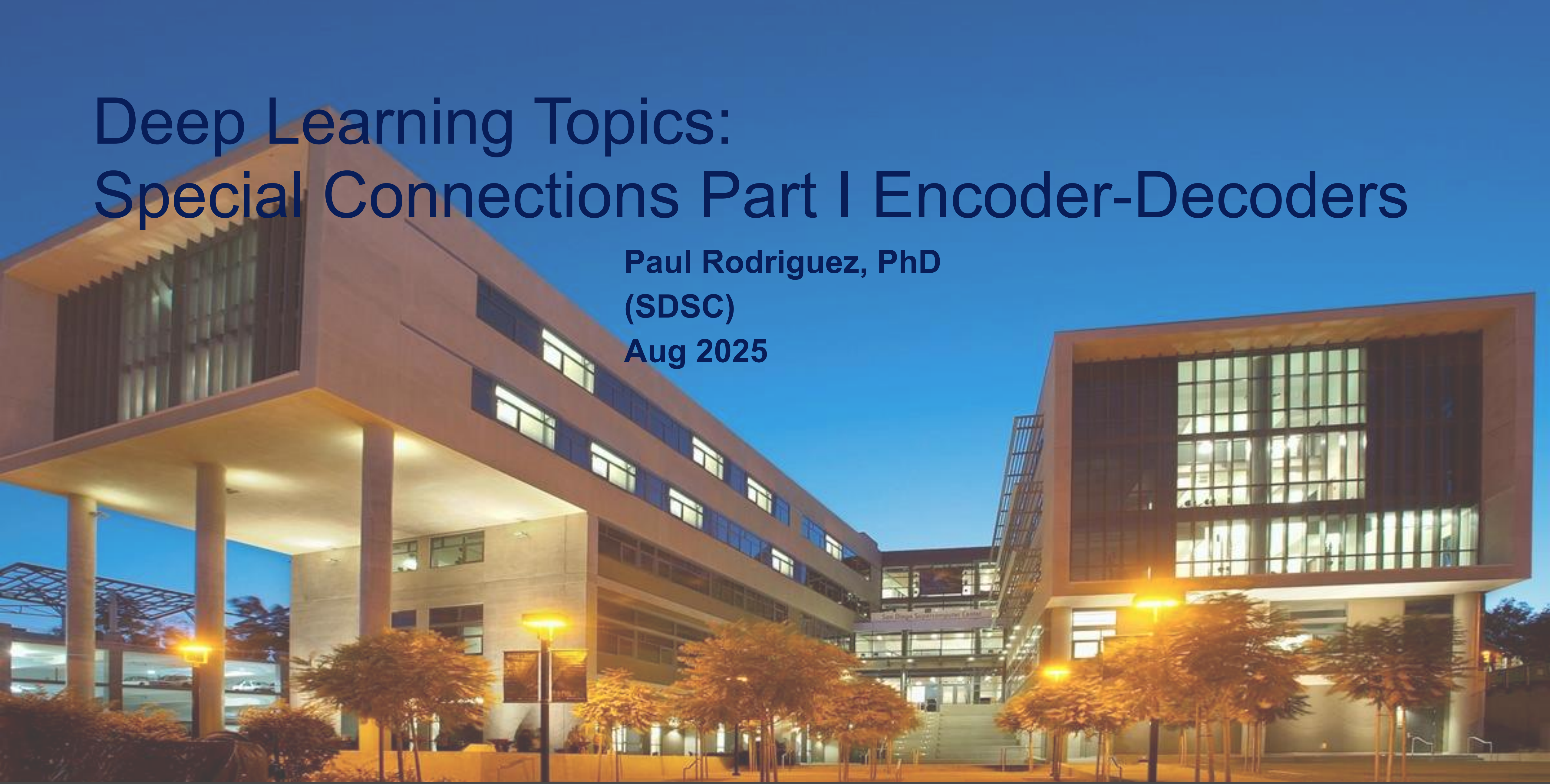
Deep Learning Topics: Special Connections Part I Encoder-Decoders

Paul Rodriguez, PhD
(SDSC)

Aug 2025



Outline

- **Part I**

Gate connection idea

Skip and Residual connections

Programing connections and Keras Model API

Encoder-Decoder (Autoencoder)

Exercise MNIST Autoencoder

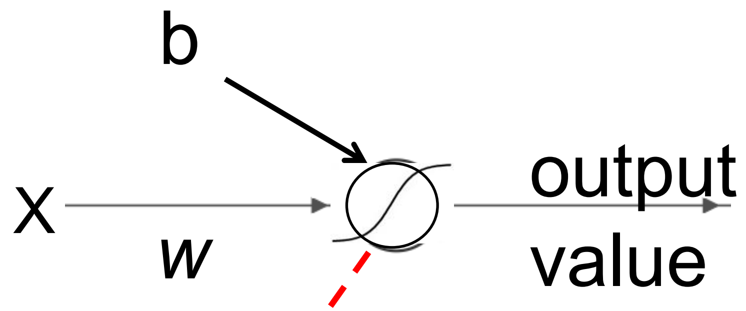
Autoencoder with Stable Diffusion

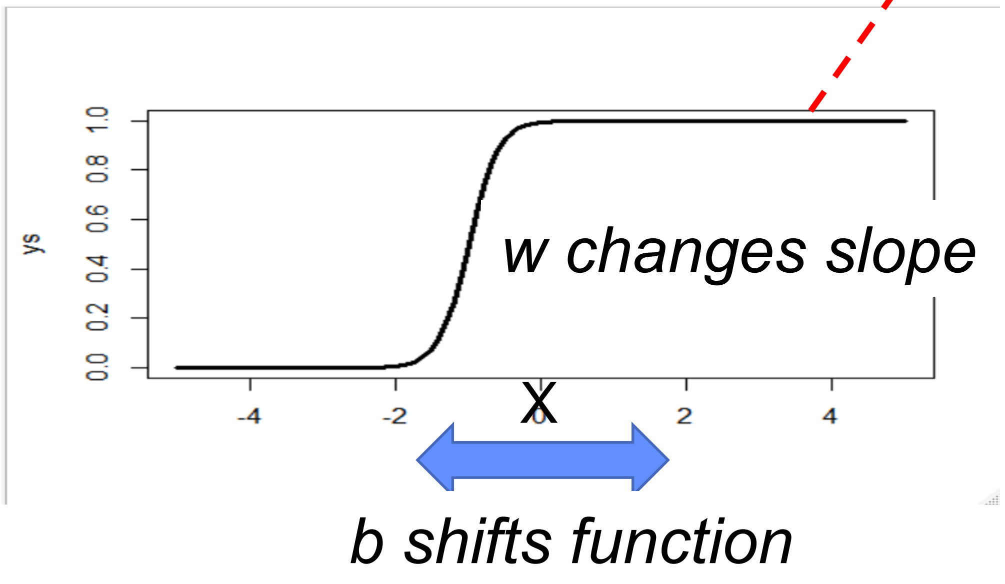
- **Part II**

Attention Head and Transformers,

1. Gate connection: Recall the logistic unit

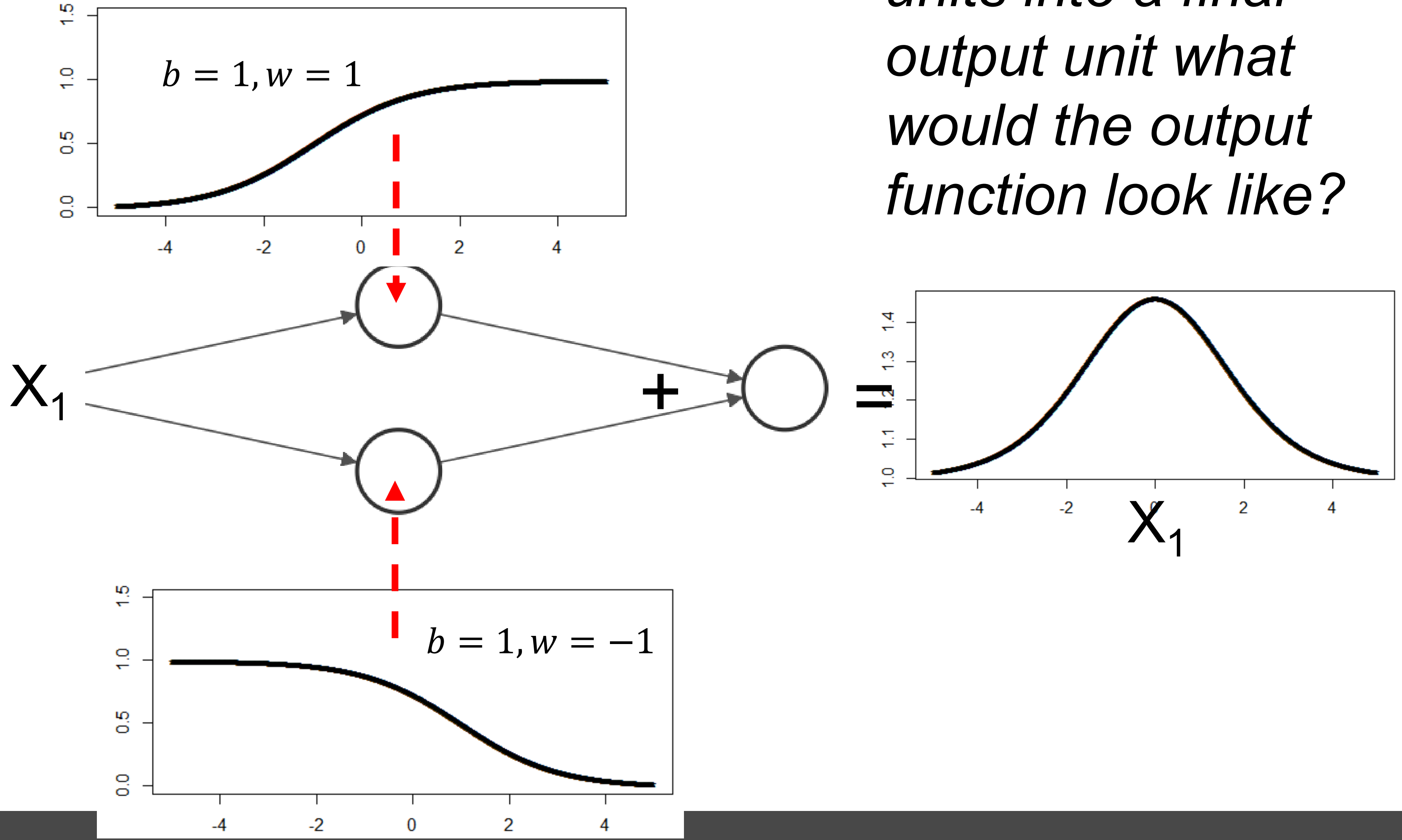
$$f(x) = \frac{1}{1 + exp^{(-(b+wx))}} \Leftrightarrow$$



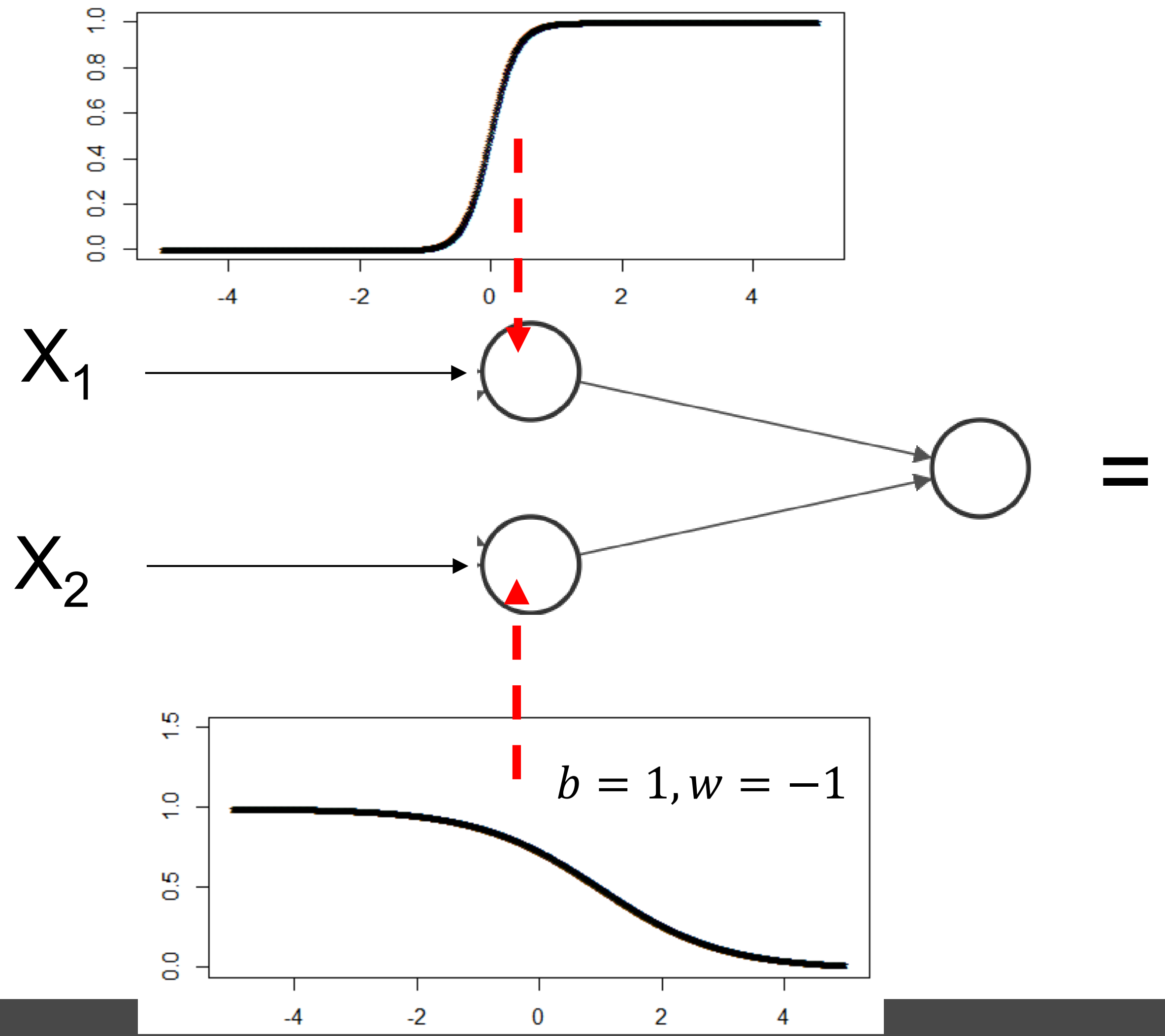


Example: 1 input into 2 logistic units with these activations

If you add these 2 units into a final output unit what would the output function look like?

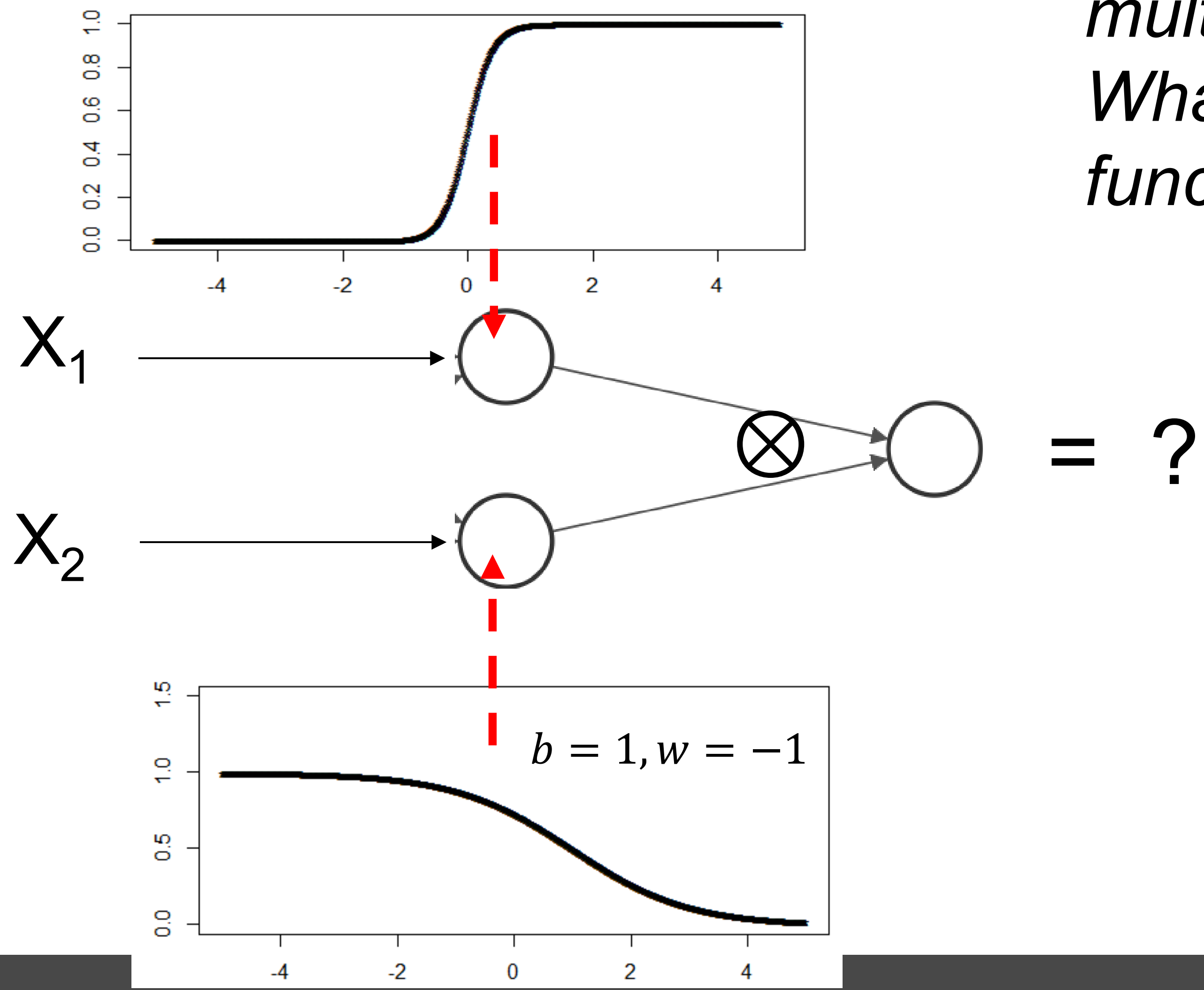


Example: 2 input into 2 logistic units with these activations

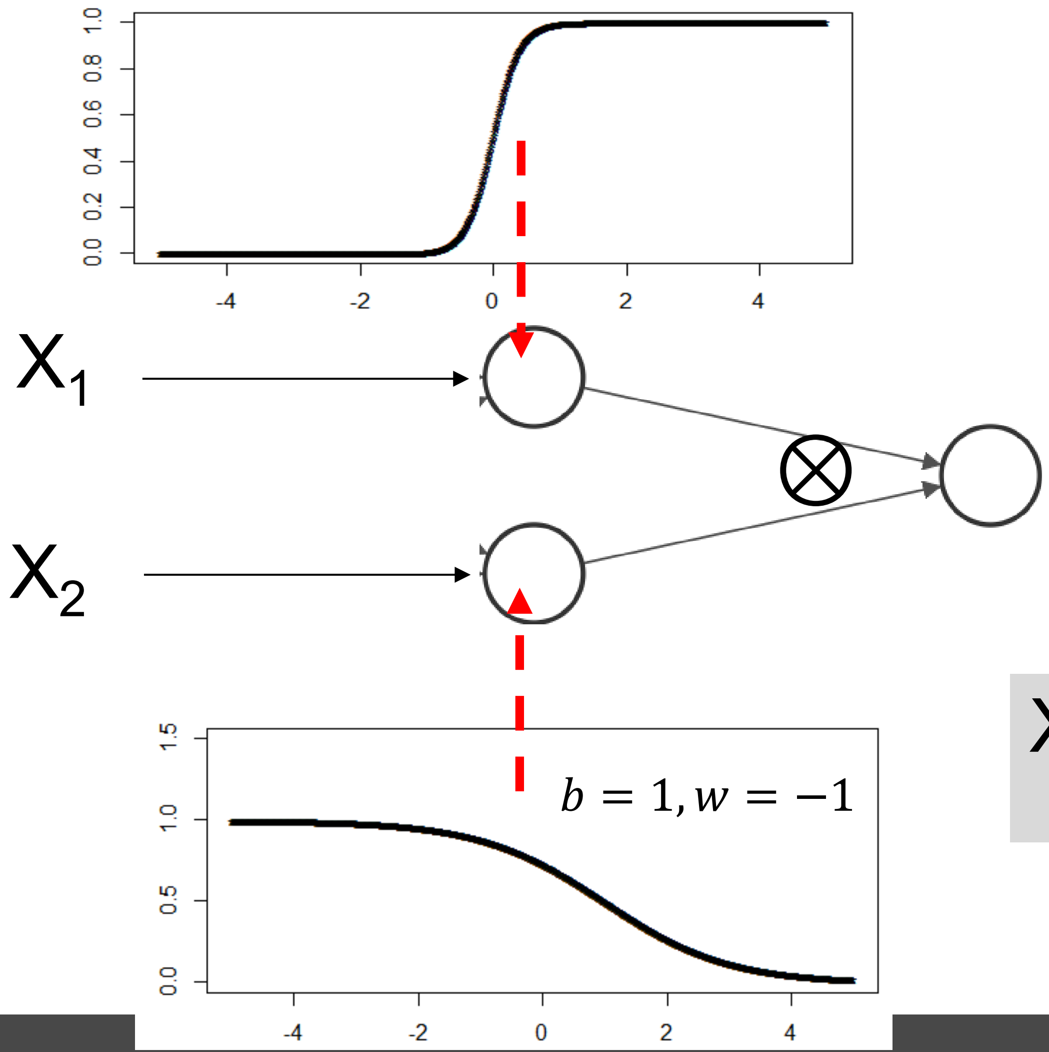


Example: 2 input into 2 logistic units with these activations

What if you multiply these? What is the output function doing?



Example: 2 input into 2 logistic units with these activations

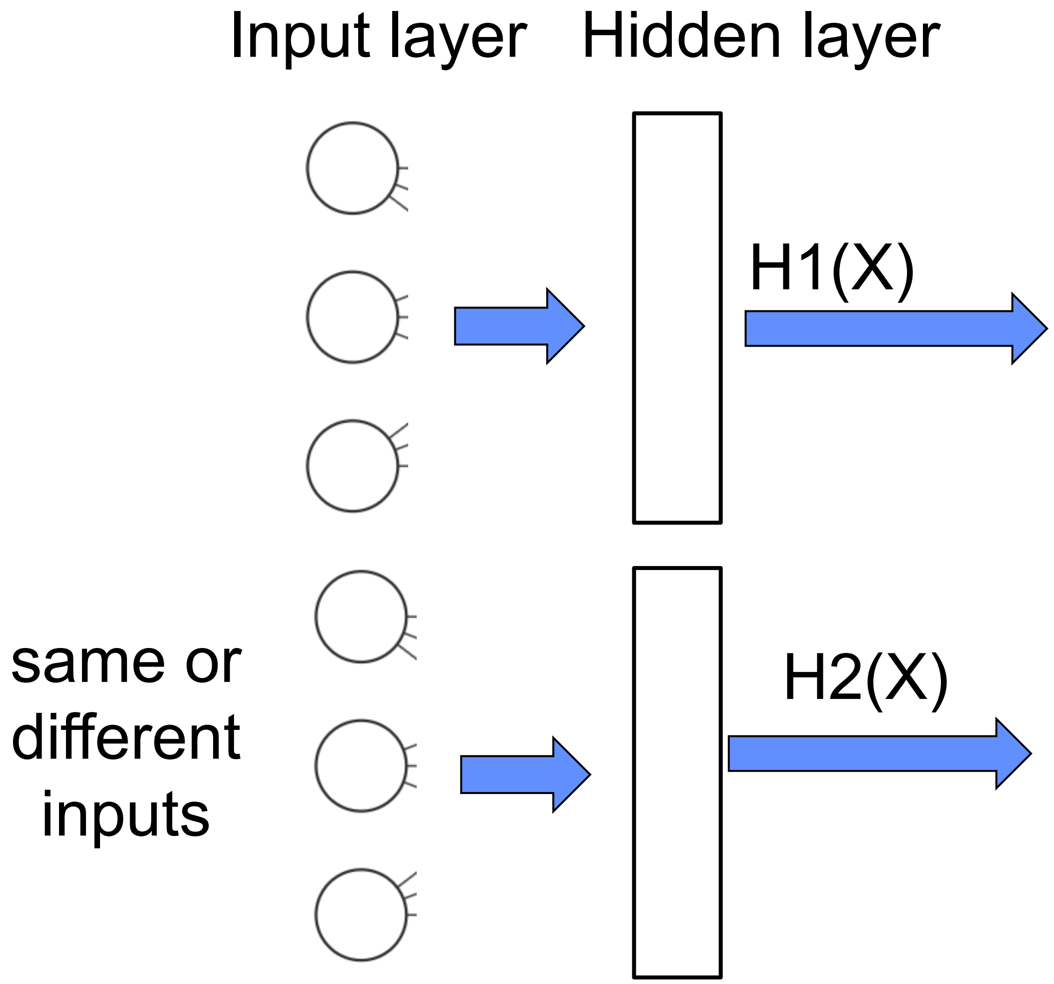


What if you multiply these?
For linear activation, what is the output function doing ?

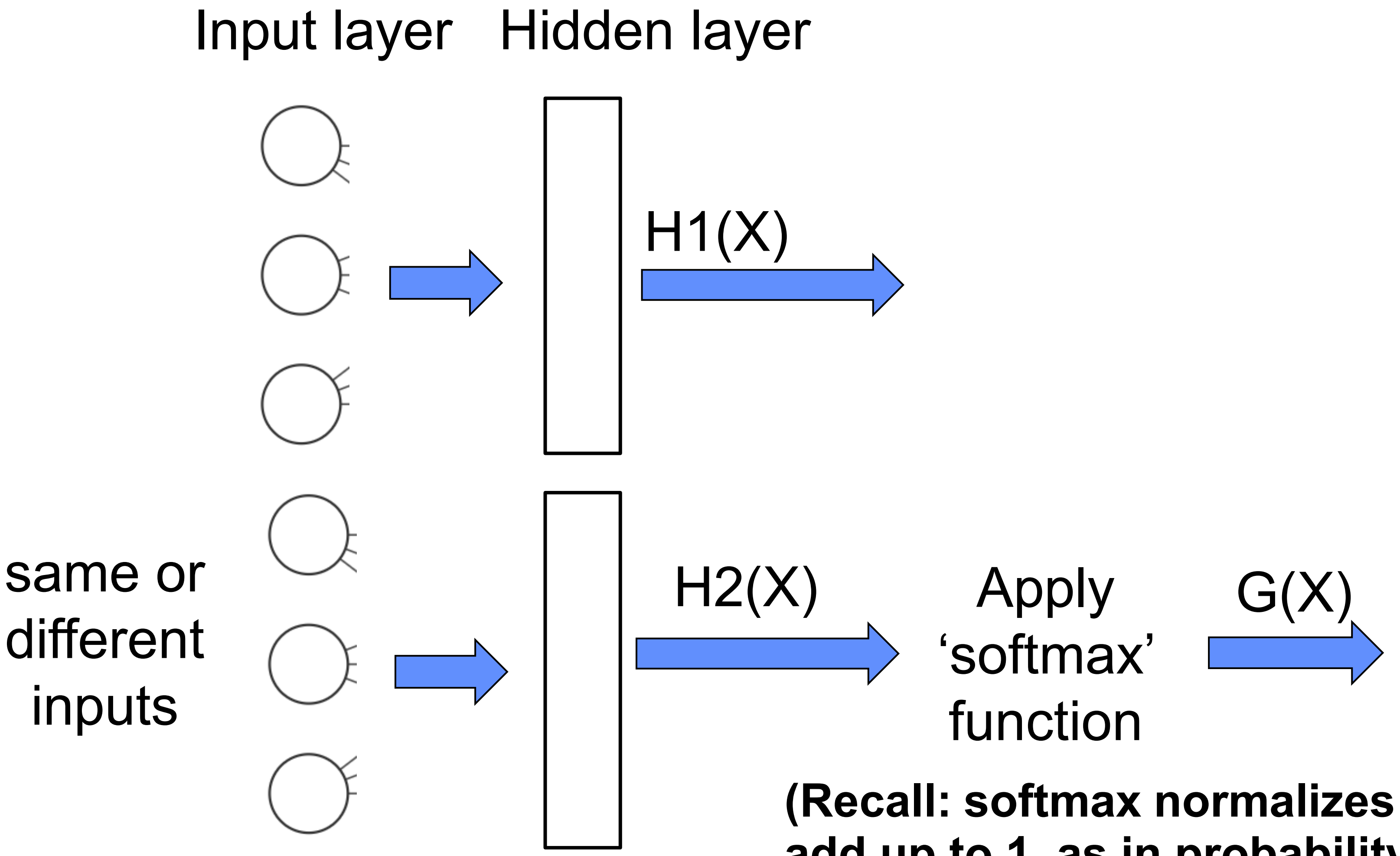
$$= \begin{cases} 0 & \text{if } X_1 < 0 \\ h(X_2) & \text{if } X_1 > 0 \end{cases}$$

X_1"gates" X_2 activation

Drawing the gate for two sets of hidden units

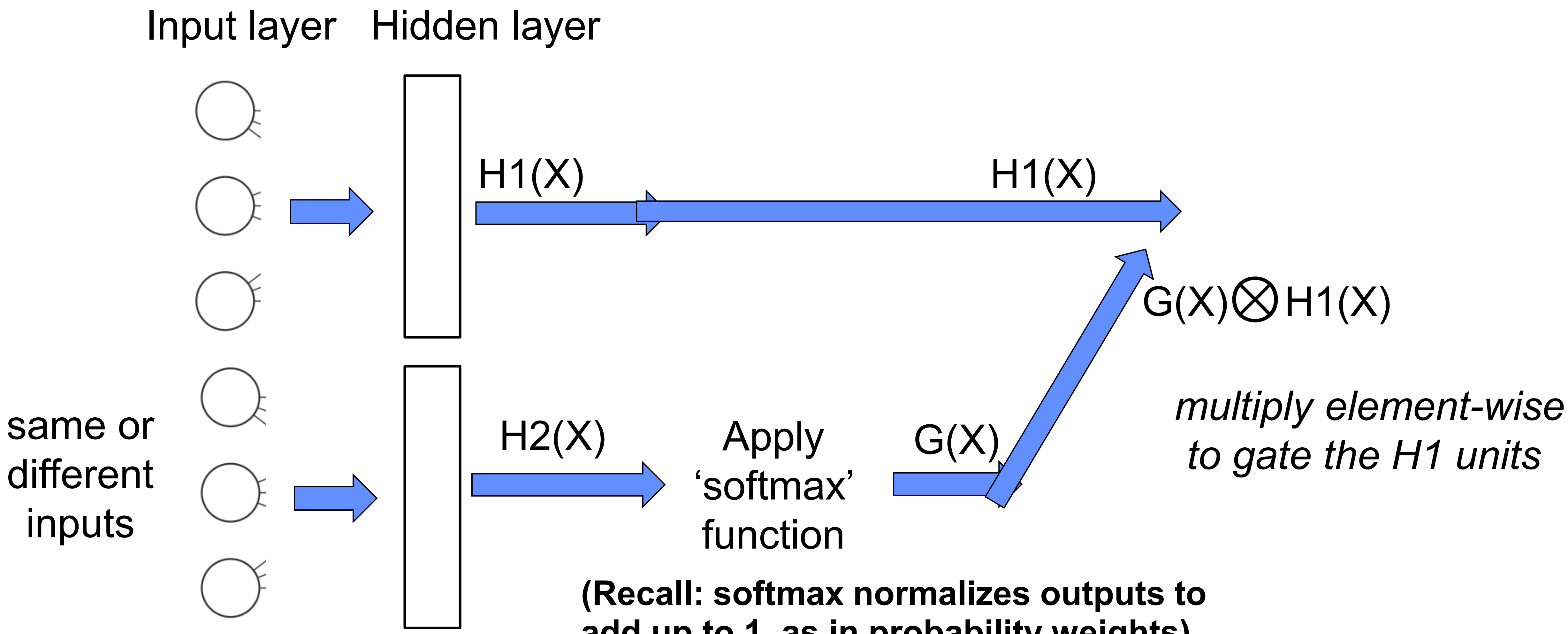


Drawing the gate for two sets of hidden units



(Recall: softmax normalizes outputs to add up to 1, as in probability weights)

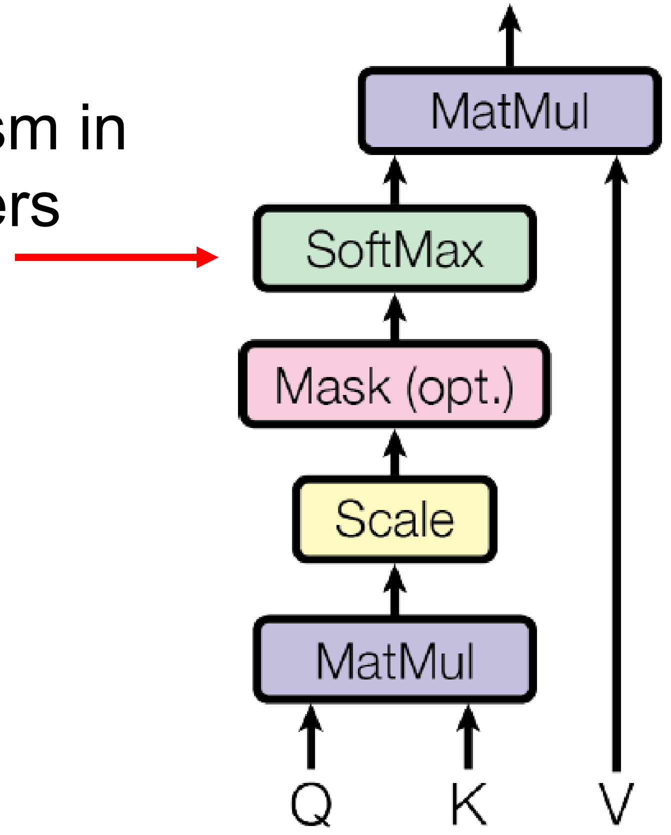
Drawing the gate for two sets of hidden units



(Recall: softmax normalizes outputs to add up to 1, as in probability weights)

Scaled Dot-Product Attention
(very rough summary)

"Attention" mechanism in language transformers use a softmax gate

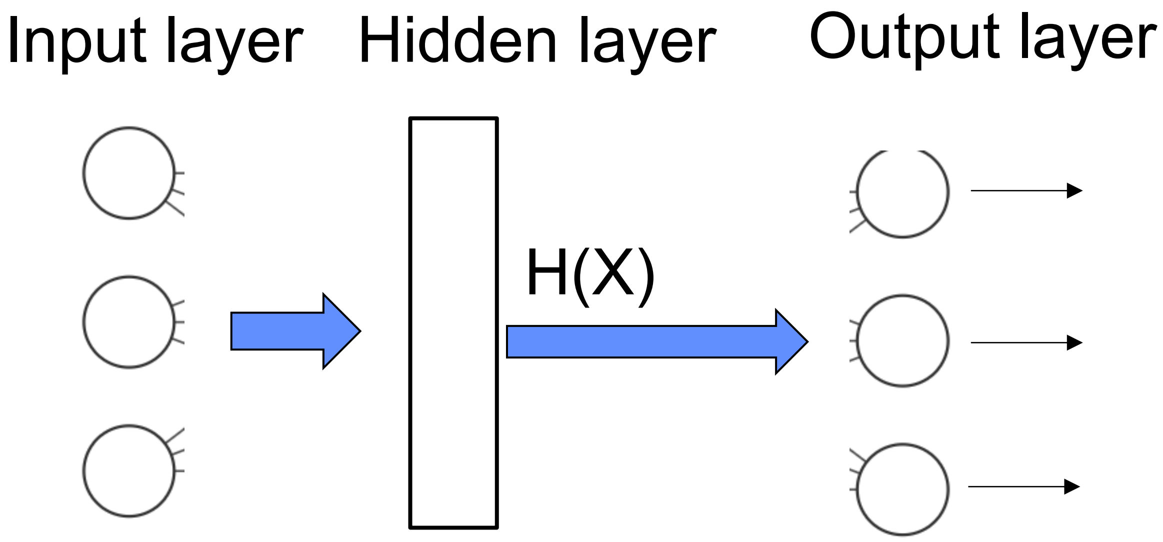


The gate is applied to possible prediction Values for decoding

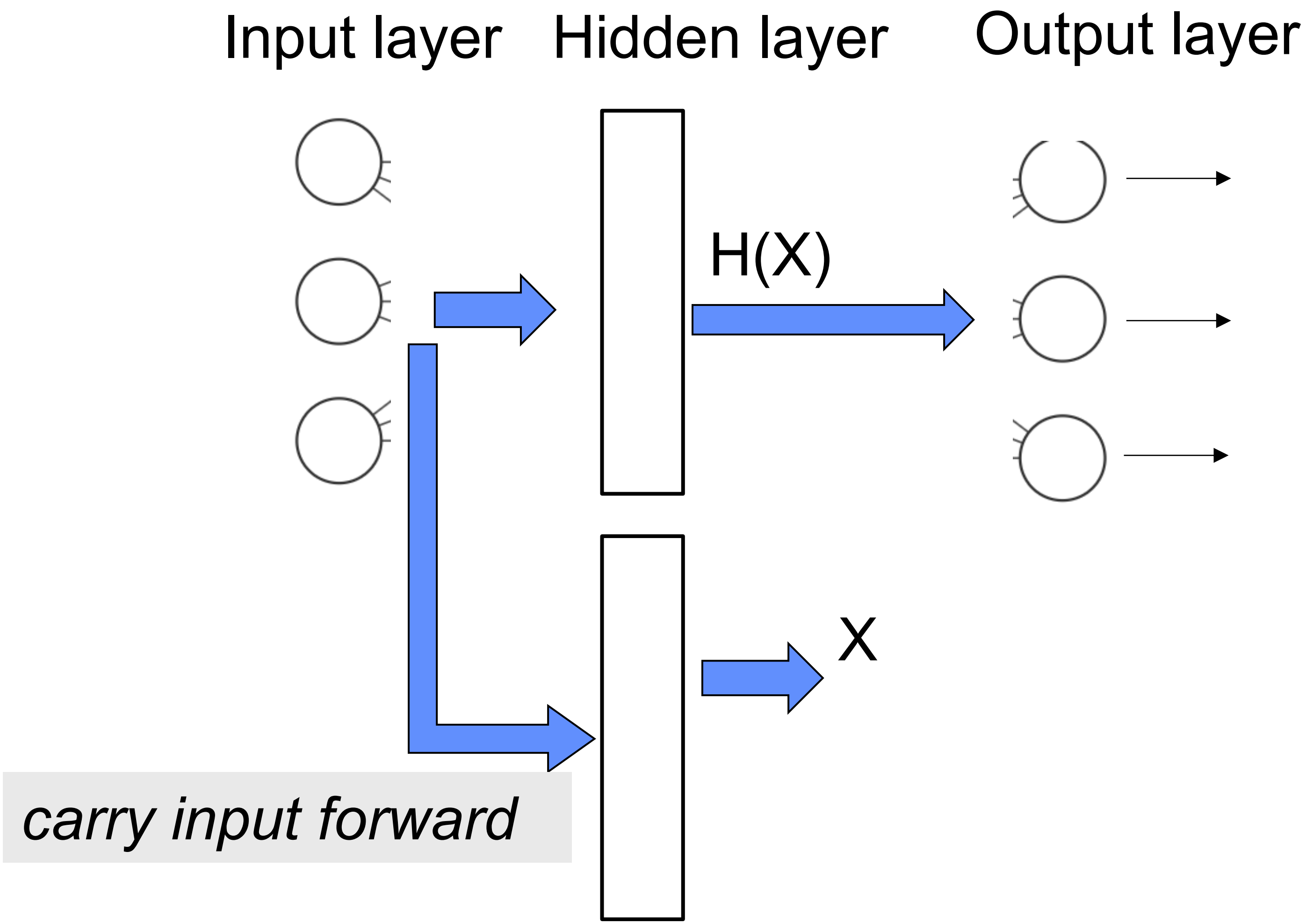
Q,K,V depend on input

Vaswani, et al. 2017
Attention Is All You Need (for Transformers)

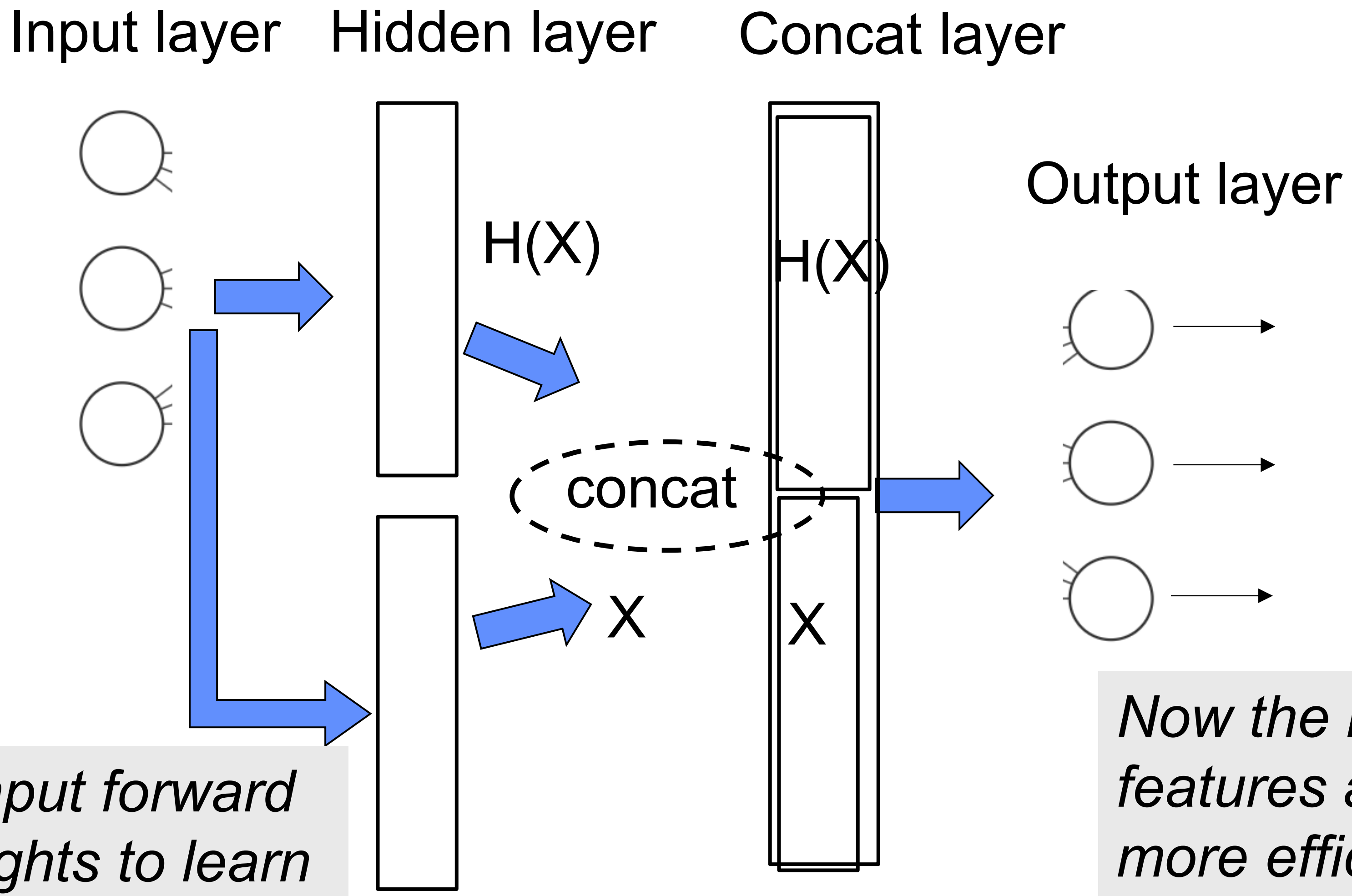
2. Skip connection: Recall the Multilayer Perceptron



To help the MLP learn directly from input carry input forward



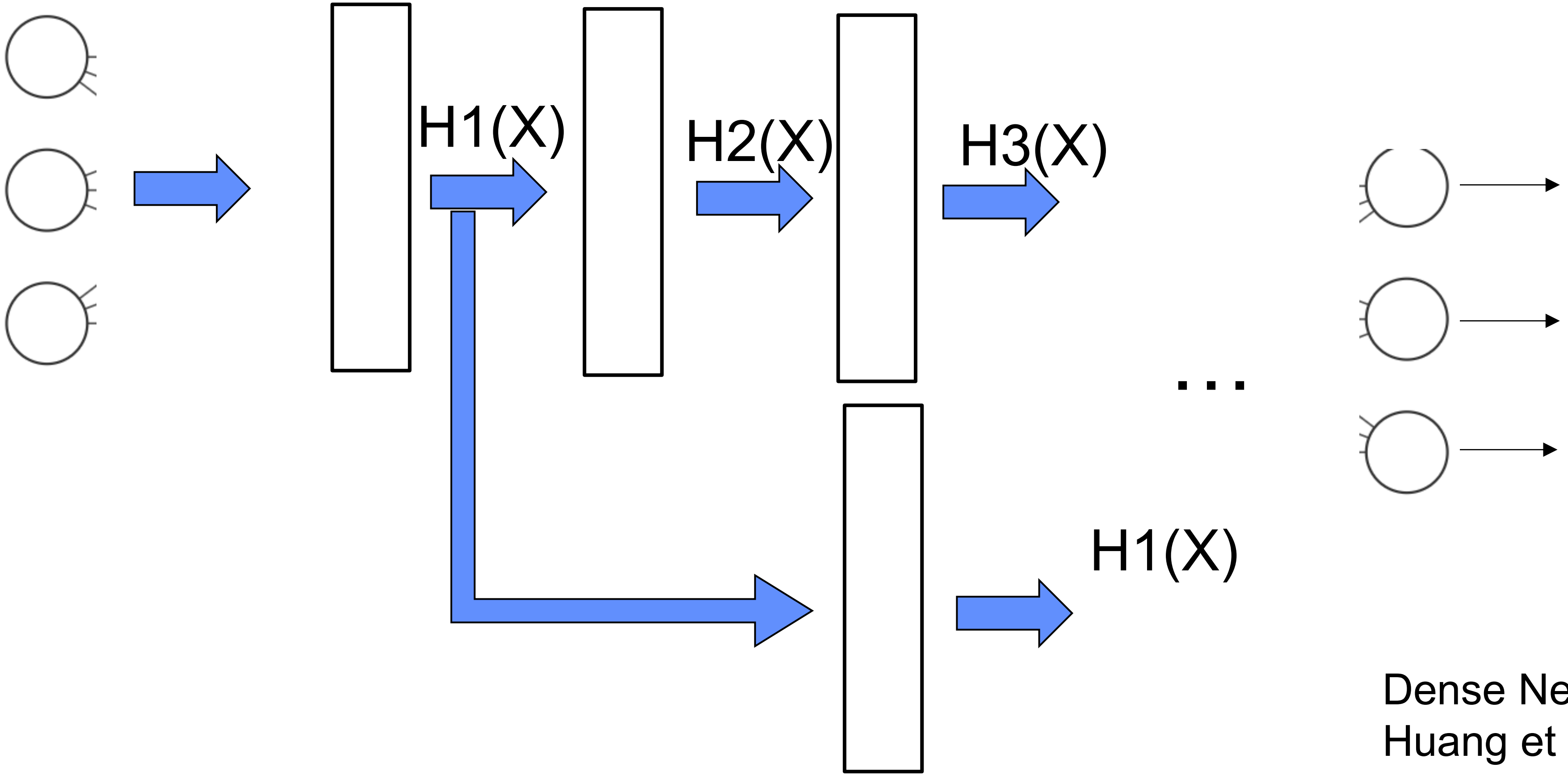
Concatenate input with hidden units into new layer



carry input forward
No weights to learn

Now the input features are used more efficiently

Can be done for any (or all) previous layer and *skip* any number of layers

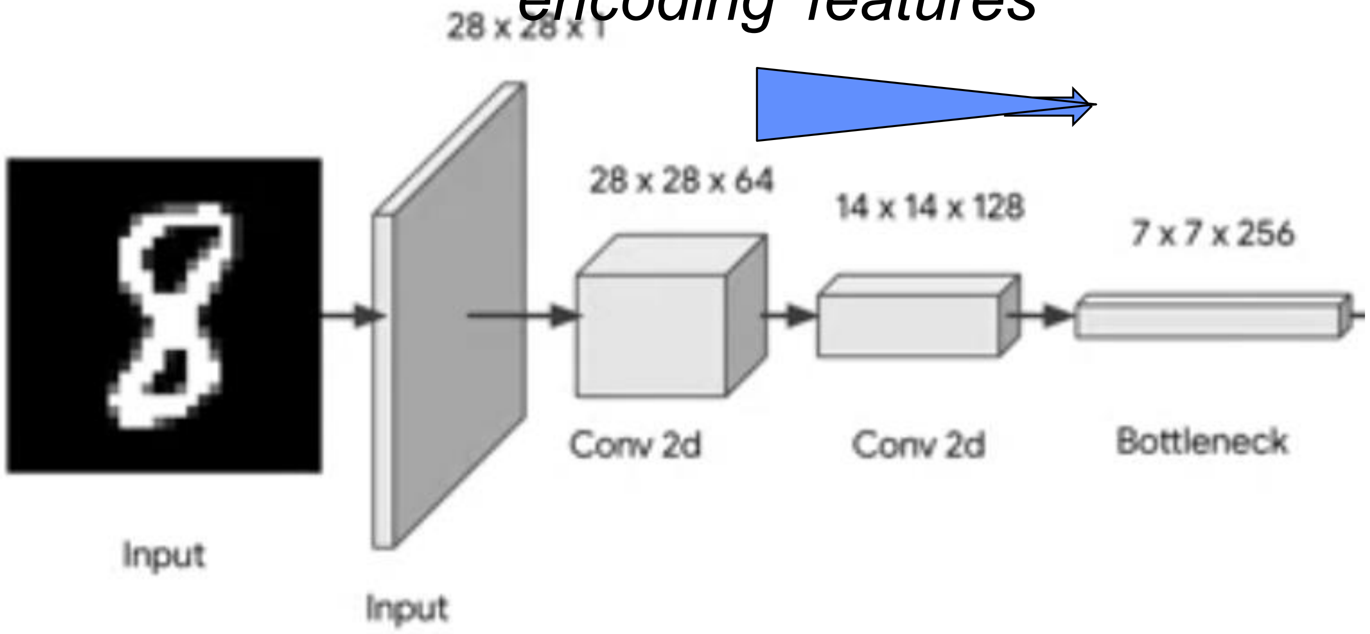


Dense Nets
Huang et al

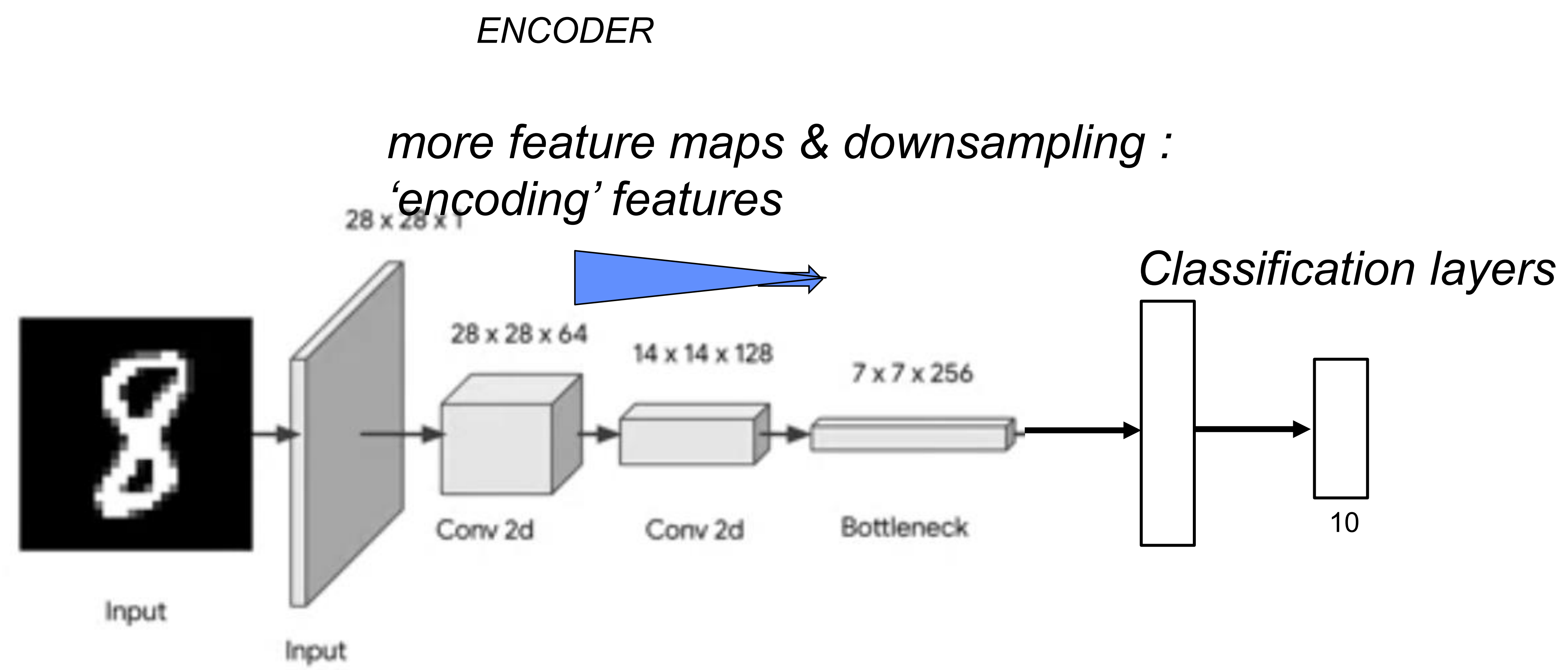
Recall: CNN architecture for MNIST classification

ENCODER

more feature maps & downsampling :
'encoding' features



Consider: CNN architecture for MNIST classification

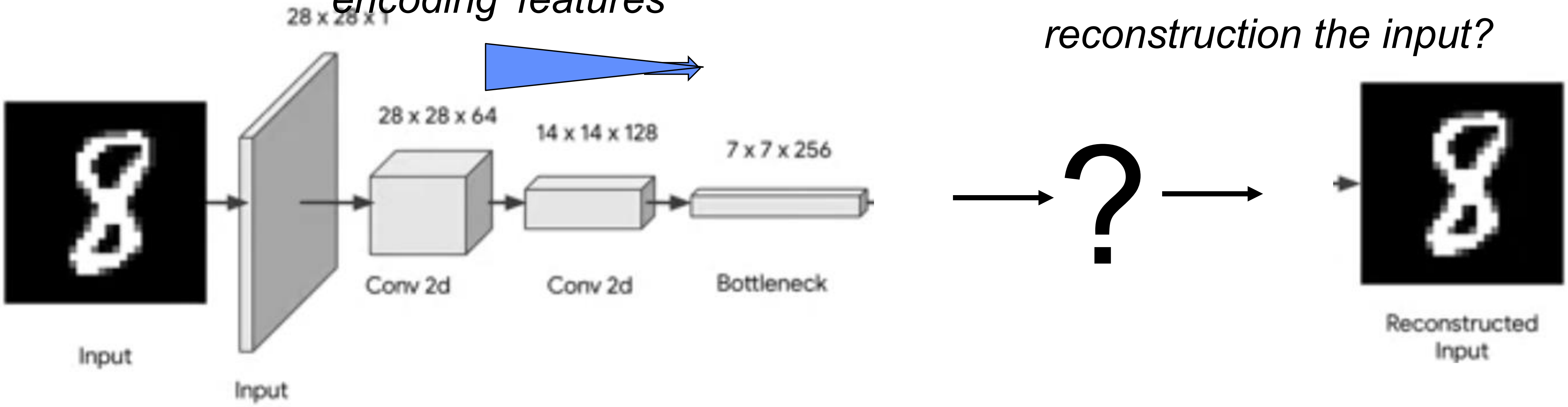


A CNN architecture for MNIST autoencoding

ENCODER

more feature maps & downsampling : 'encoding' features

What if we want to reconstruction the input?



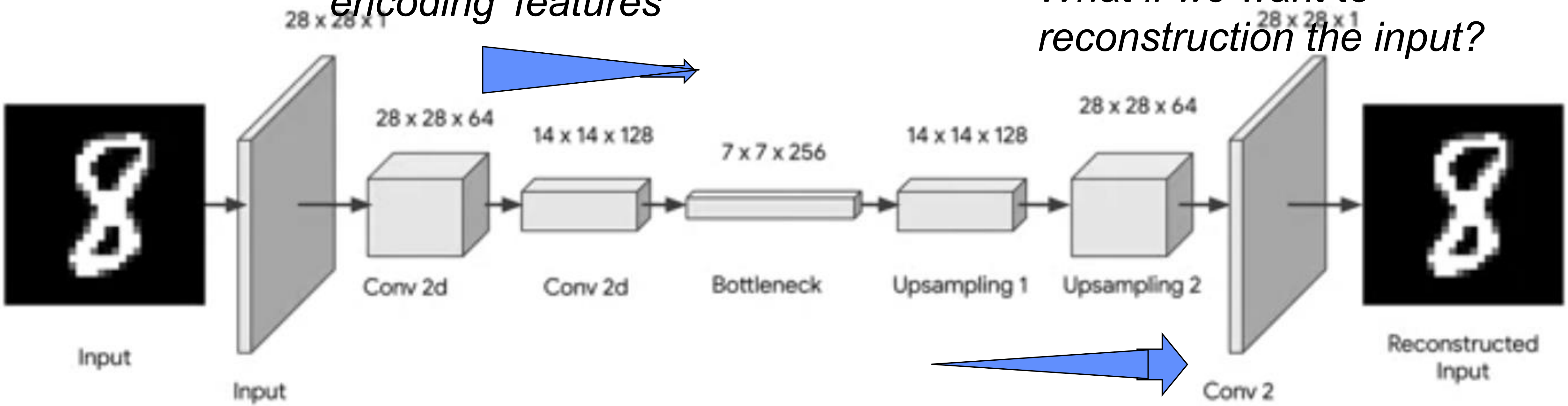
A CNN architecture for MNIST autoencoding

ENCODER

DECODER

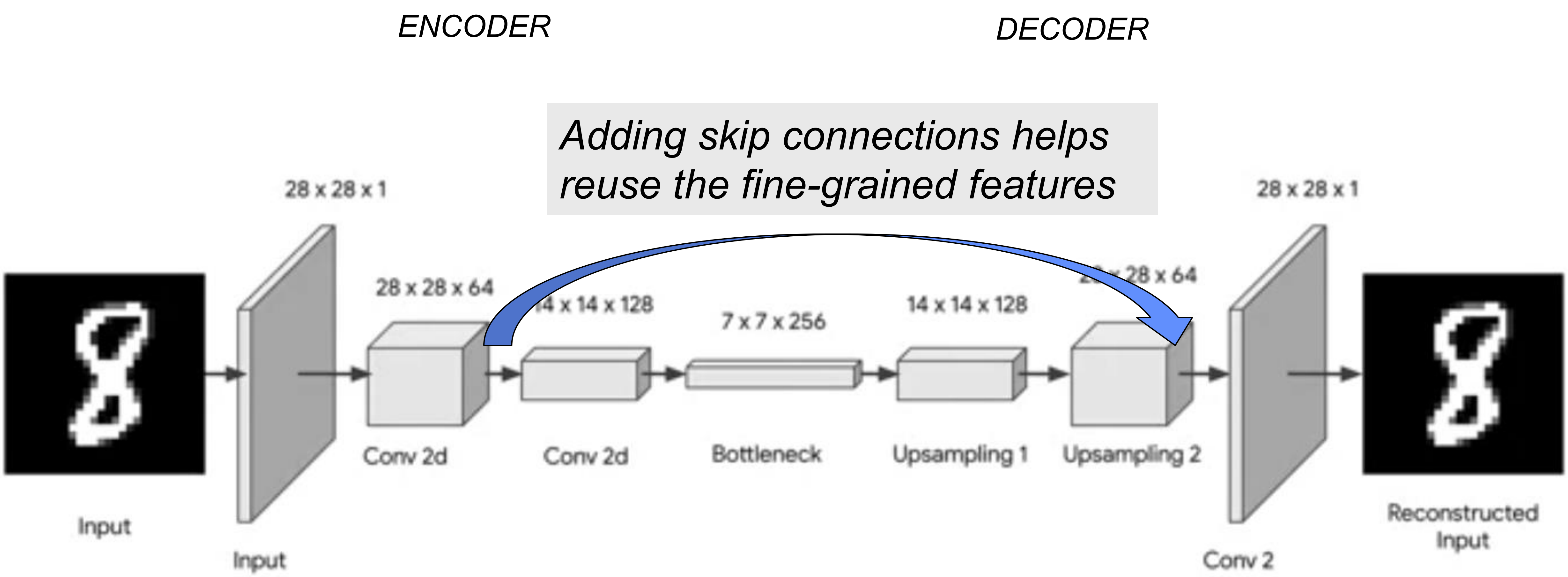
more feature maps & downsampling : 'encoding' features

What if we want to reconstruction the input?

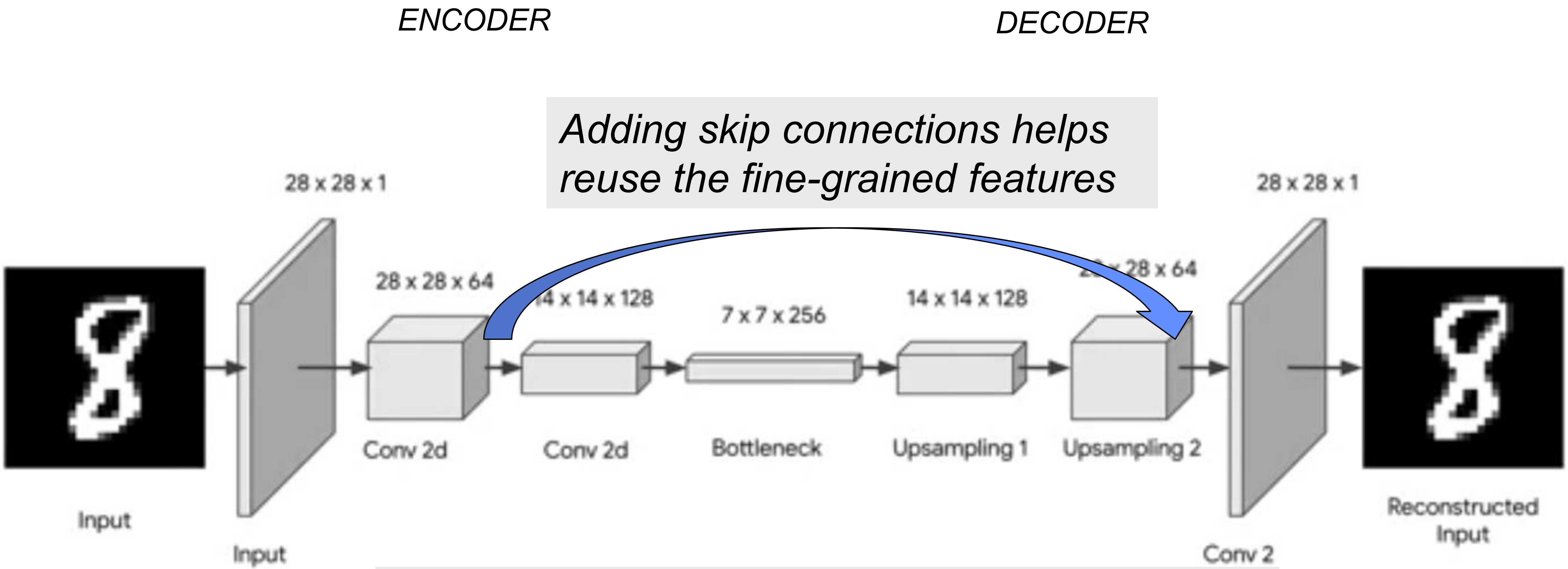


Use 'upsampling' to reverse the encoding – ie. decoding

A CNN architecture for MNIST autoencoding



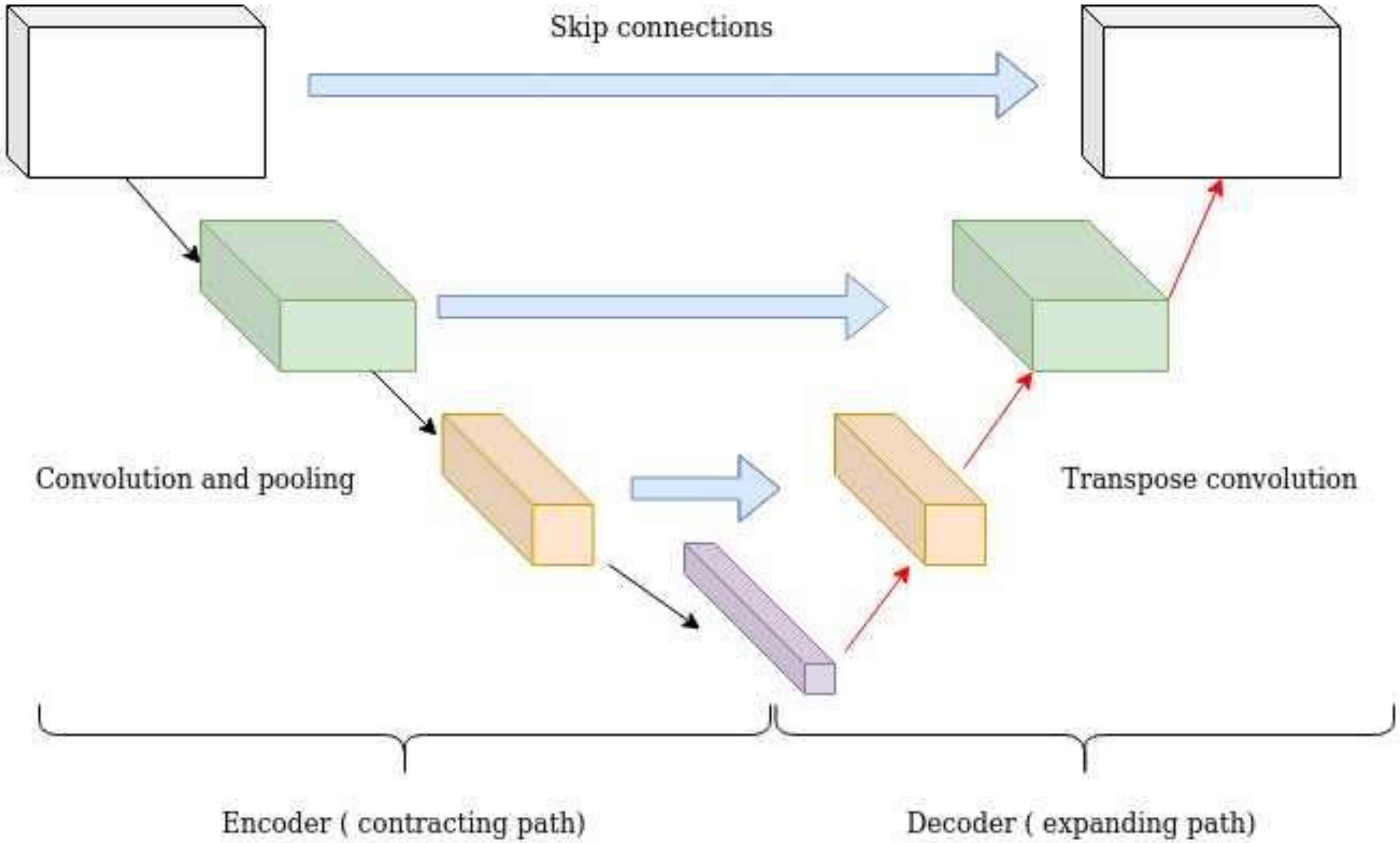
A CNN architecture for MNIST autoencoding



Adding skip connections helps reuse the fine-grained features

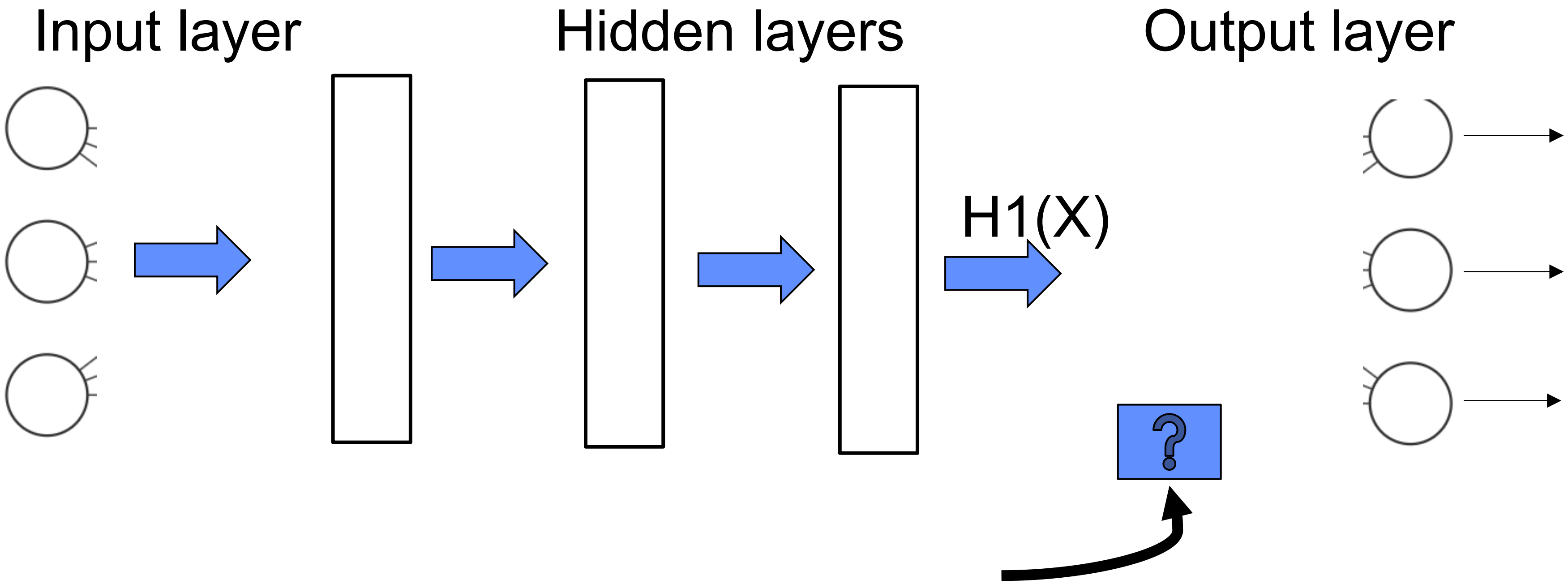
NOTE the 28x28x64 encoded maps have to be skipped ahead to where the 28x28x64 decoding maps are – which axis is concatenated?

Image Encoder-Decoder is a “UNET” architecture



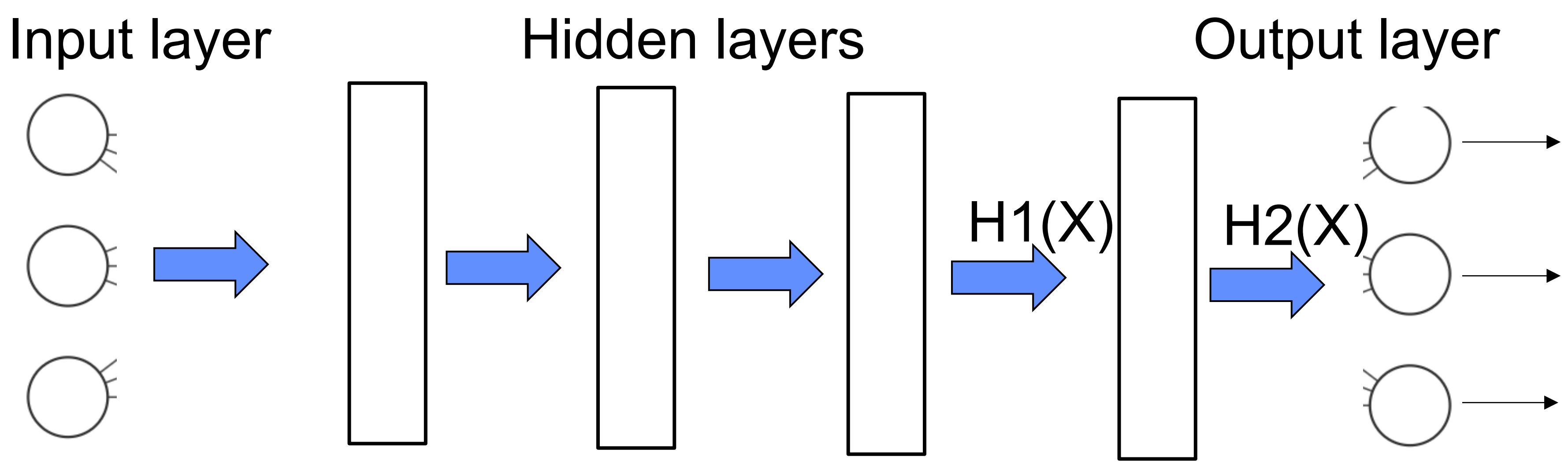
- **pause**

3. Residual connection: Can we keep adding layers?



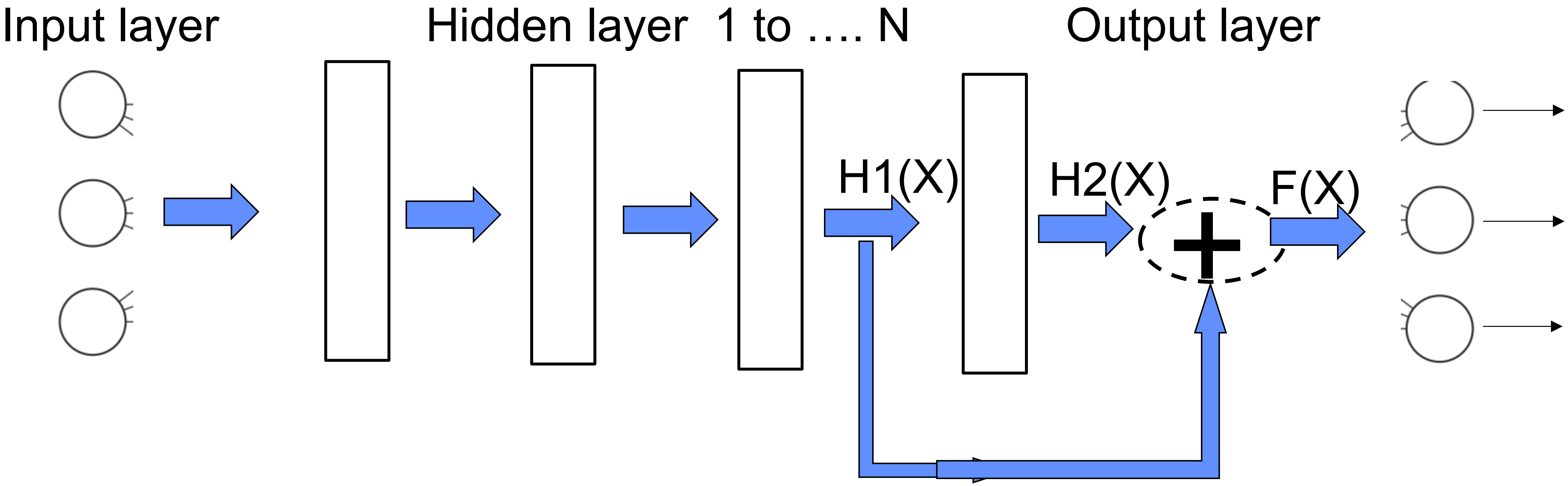
Given some deep network,
should I add another layer?
What should a new layer learn?

Consider: Can we keep adding layers?



If H1(X) is good then this new layer could be unnecessary or only for learning small differences - eg H2(X) should be almost same as H1(X)

Skip with addition makes a ‘residual’ connection



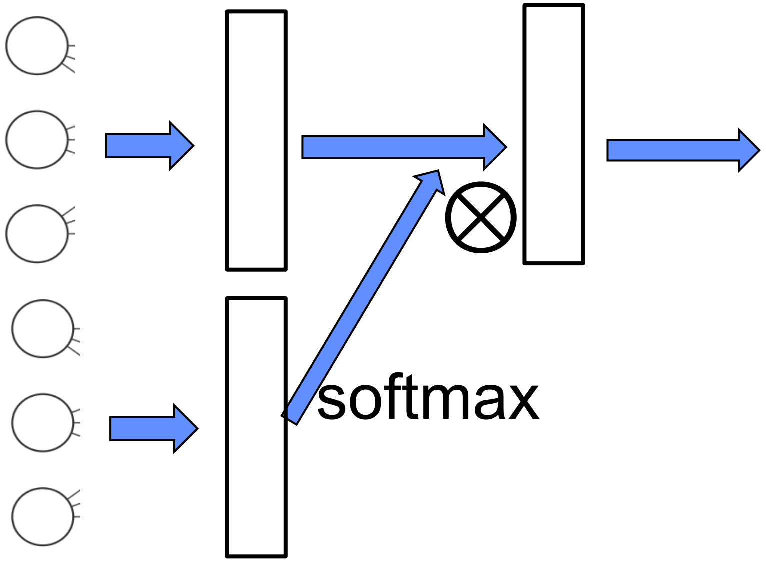
Make it easy for next layer to learn only incremental changes –
e.g. use F(X)=H2(X)+H1(X) so that H2(X)=F(X)-H1(X) .
The H2() function learned is a residual function

“Resnet” residual connections help deeper learning

Deep Residual Learning,
He et.al, 2015

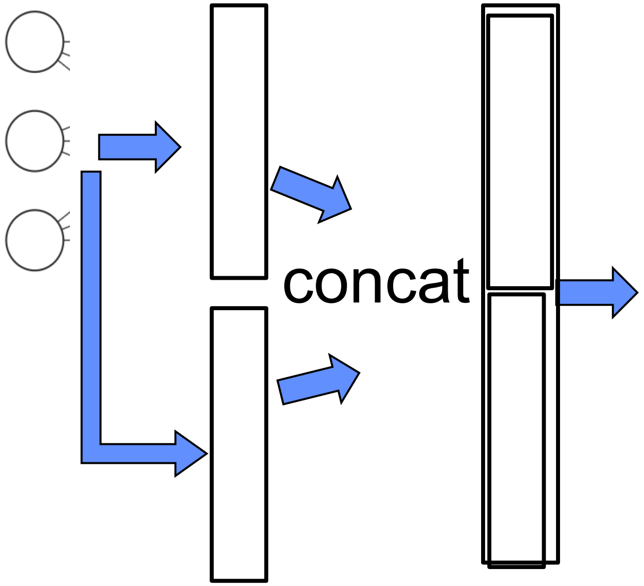
Summary: useful connections for architectures, and the intuitions

Softmax for gating



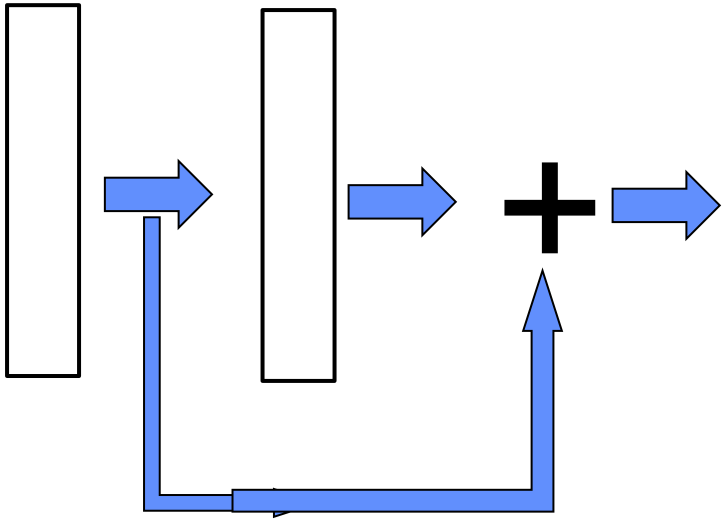
Recurrent nets,
language transformer nets

Skip connections
for feature reuse



UNET, also
feedforward nets..

Residual connections
help deeper learning



Resnet, large image
classification

Exercise

- MNIST autoencoder, reconstruct digits from noisy inputs

- Add skip connections with concatenation

Note: make sure you see how the outputs from encoding layers are matched up to inputs for decoding layers!

e.g. 14x14 encoding feature maps should be concatenated with 14x14 decoding maps

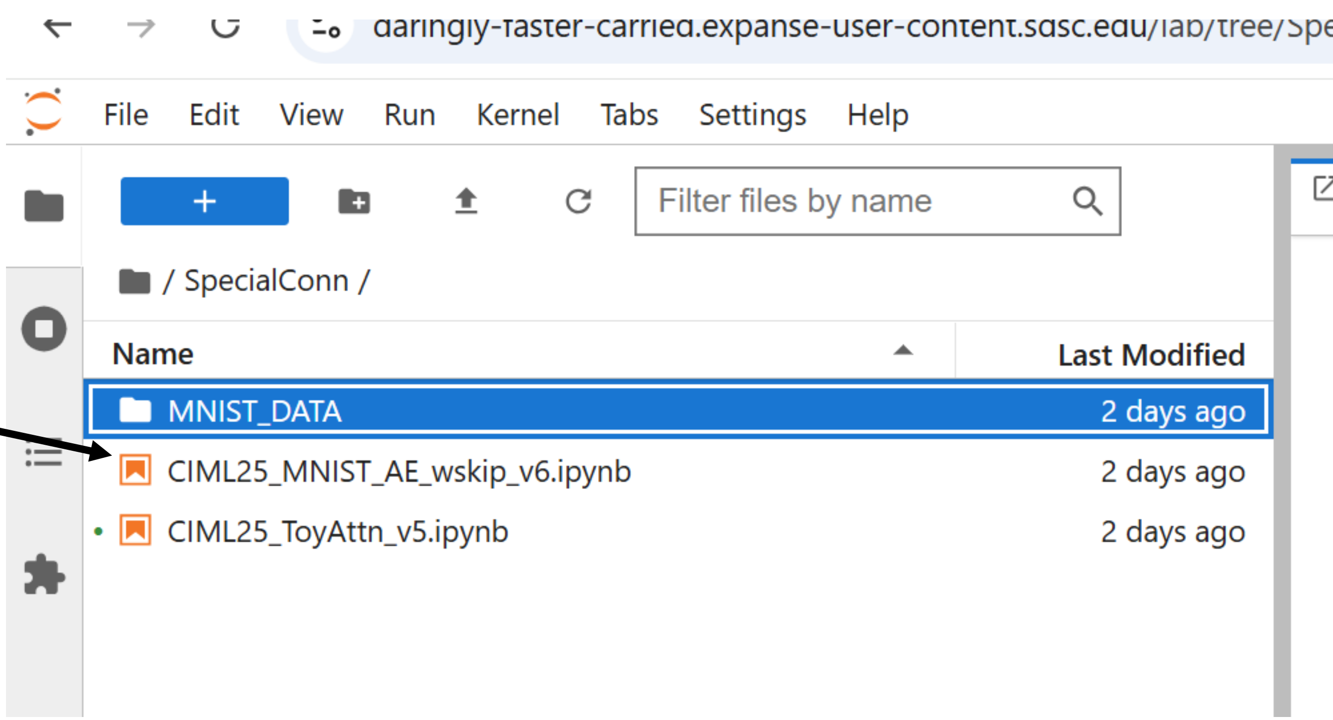
- Review outputs to see improvements

Login to expanse and start a notebook on gpu-shared queue

$ jupyter-gpu-shared-pylight

In jupyter notebook session open the (SI25) MNIST_Autoencoder_wskip notebook

Follow instructions in the notebook



Quick overview of code

```
# -------------------------------------------------------------
class MyEncoder(torch.nn.Module):
    def __init__(self):
        super(MyEncoder, self).__init__()
        #convolution layer then max pool to downsize
```

Encoder object

```
        x2 = F.relu(x2)
        x2 = self.max_pool_2(x2)
        #print('MYINFO enc fwd, after max2, x shape:',x2.shape)
        #return x2         #or x1,x2 to use skip connections
        return x1,x2    #or x1,x2 to use skip connections
```

Enocde forward function returns intermediate and last layer

```
# ---------------------------------------------------------------------
class MyDecoder(torch.nn.Module):
    def __init__(self):
        super(MyDecoder, self).__init__()
        #convolution layer then max pool to downsize,
        self.conv1      = torch.nn.Conv2d(numfilt*2, numfilt, 3, 1,paddin

        #if no skip connection use in channels = numfilt for Conv2
        self.conv2      = torch.nn.Conv2d(numfilt, numfilt, 3, 1,padding=

        # <<<<<<<<<-------------- uncomment this, comment out the above
        #for skip connection going into conv2 use mumfilt*2
        #self.conv2       = torch.nn.Conv2d(numfilt*2, numfilt, 3, 1,paddi
```

Decoder object

(for adding skip connections use the other self.conv2 statement by adding/deleting comments)

Quick overview of code

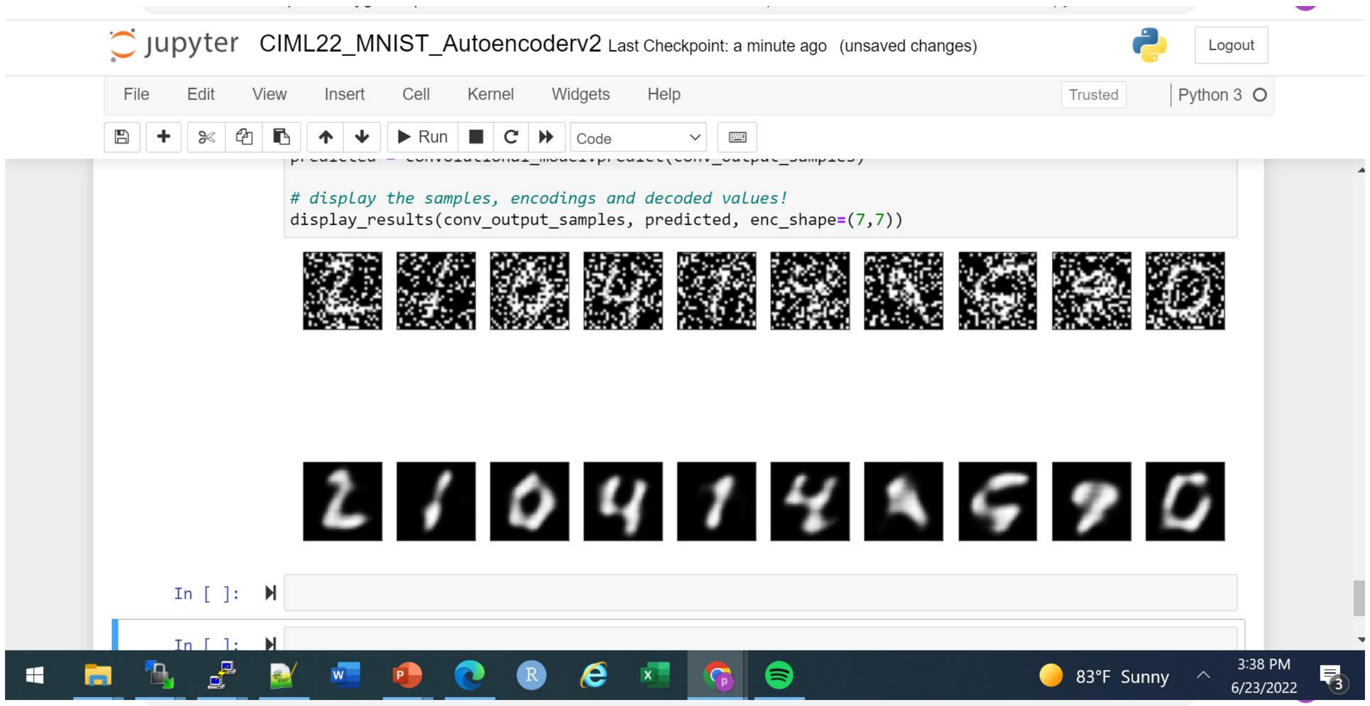
```
def forward(self, encx1,x):  #or use x1,x2 inputs
    x1 = self.conv1(x)
    x1 = F.relu(x1)
    x1  =torch.nn.functional.interpolate(x1,size=(14,14),mode='neares
    #print('MYINFO dec fwd, after inter1, x shape:',x1.shape, 'encx1s
    skip_concat_1 = torch.cat((x1,encx1), dim=1)
    #print('MYINFO, dec fwd, after concat1',skip_concat_1.shape)

    #<<<<<<---------- choose if x2 should use x1 alone, or x1 concat
    x2 = self.conv2(x1)
    #x2 = self.conv2(skip_concat_1)

    x2 = F.relu(x2)
    x2  =torch.nn.functional.interpolate(x2,size=(28,28),mode='neares
    #print('MYINFO dec fwd, after inter2, x shape:',x2.shape)
    x3 = self.conv3(x2)
    x3 = F.sigmoid(x3)
    return x3
```

Decoder forward function has an argument for the first encoding layer and last encoding output

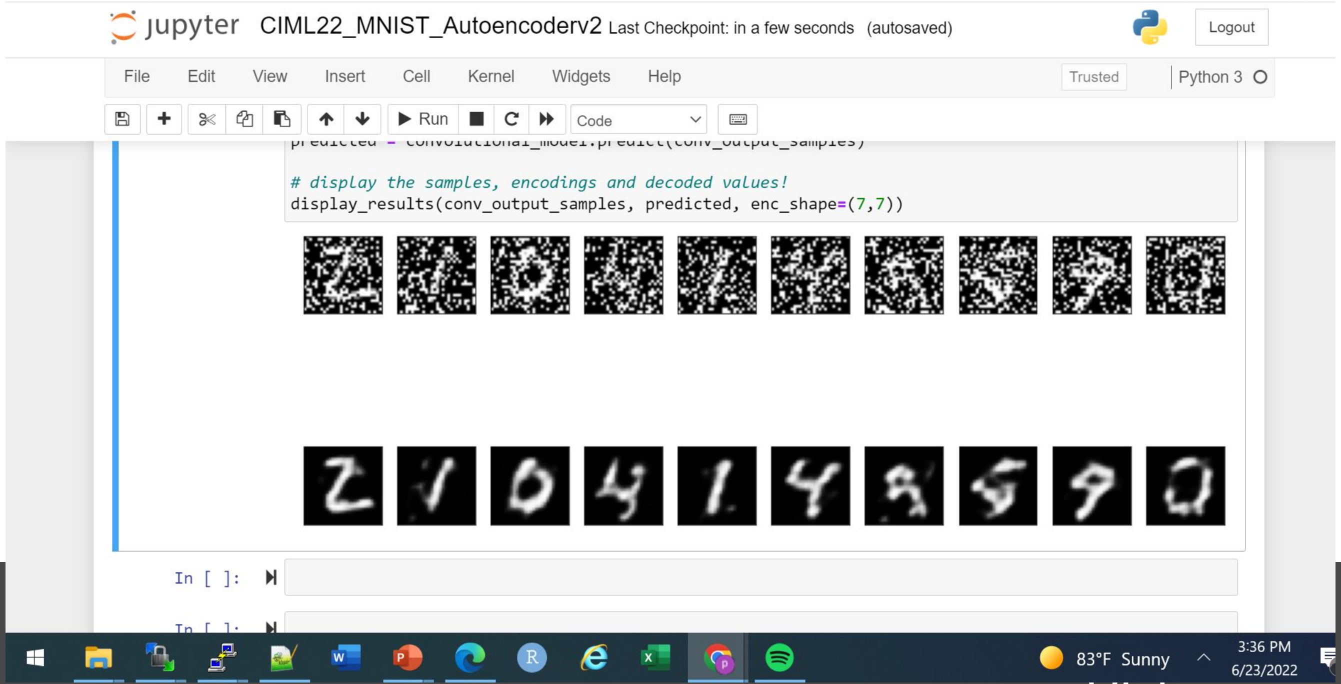
(for adding skip connections use self.conv2 with the concatenated input)



With out skip

20 epochs

Loss 0.1664



With skip,

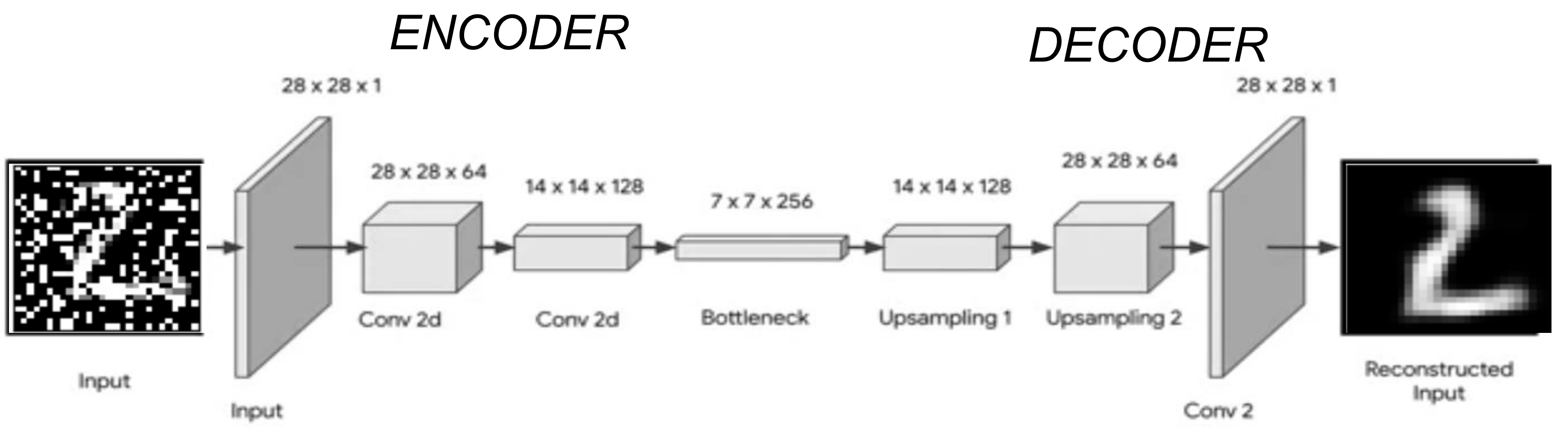
20 epochs loss 0.14

Are the numbers a little bit more reconstructed?

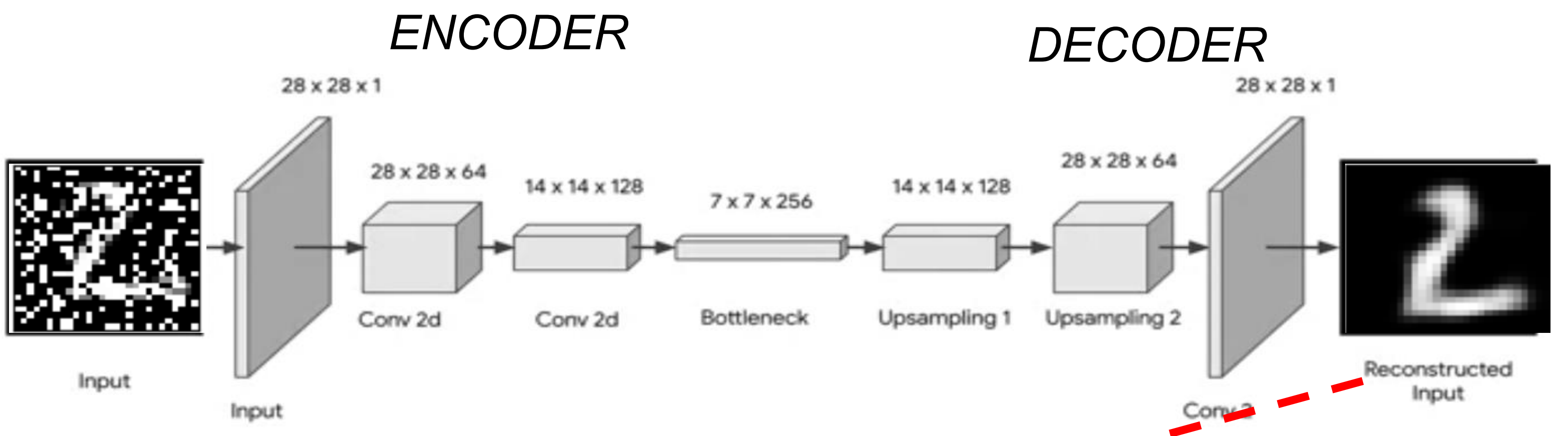
Autoencoding with Stable Diffusion

- Let's introduce the concepts and intuition behind stable diffusion

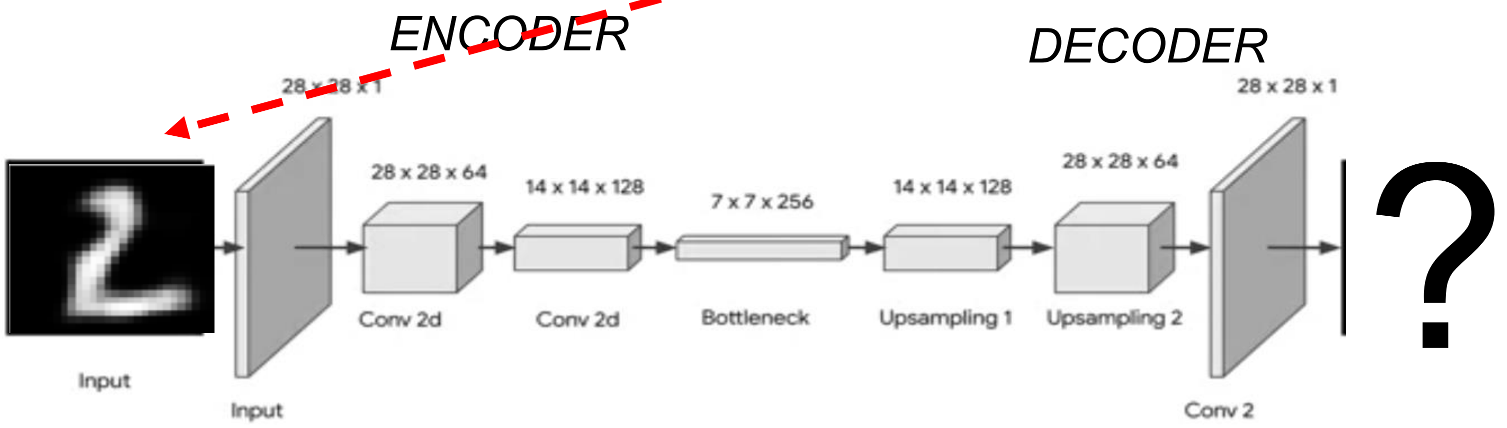
In principle, our denoising autoencoder removed noise pixels and/or filled in number pixels



In principle, our denoising autoencoder removed noise pixels and/or filled in number pixels

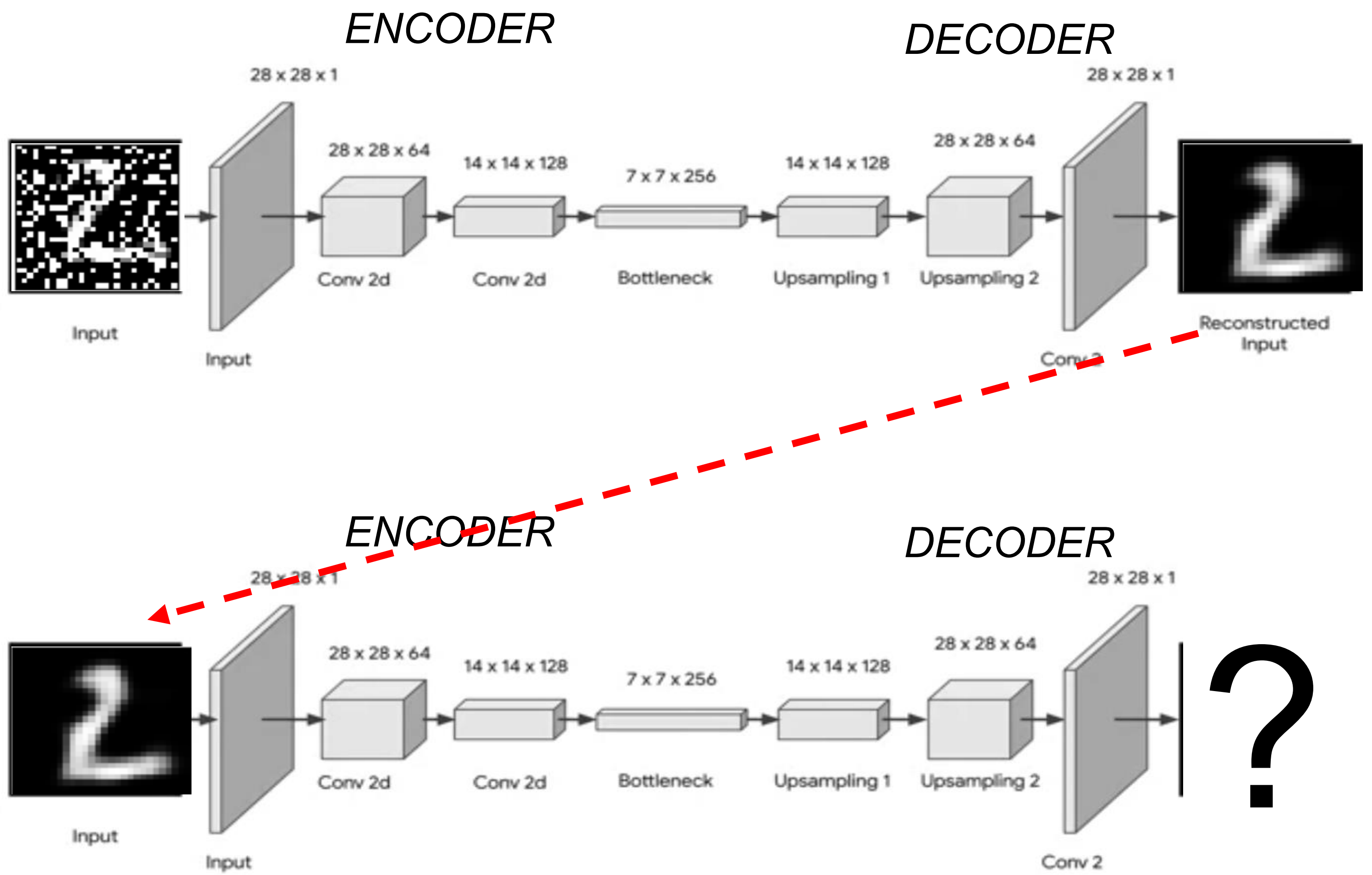


What would would happen if we fed the denoised output back into the autoencoder?



In principle, our denoising autoencoder removed noise pixels and/or filled in number pixels

What would would happen if we fed the denoised output back into the autoencoder?
A: better reconstruction
B: all pixels would be removed
C: all pixels would be filled in



First step of denoising

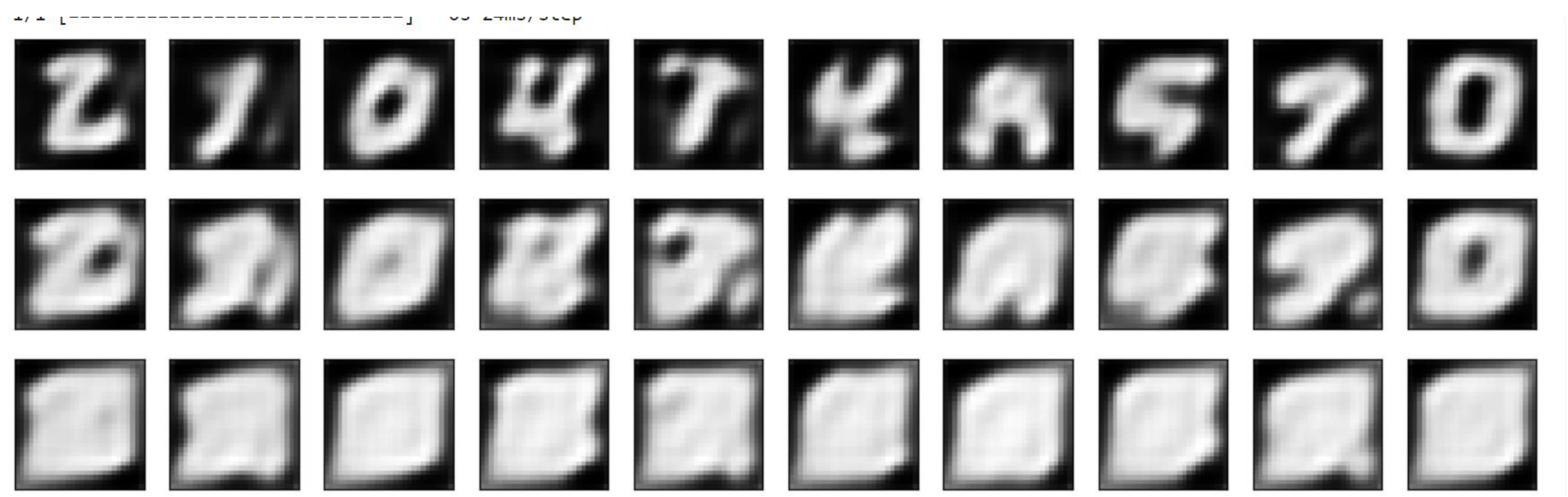
first step of denoising

1 more step of denoising

Is it better?

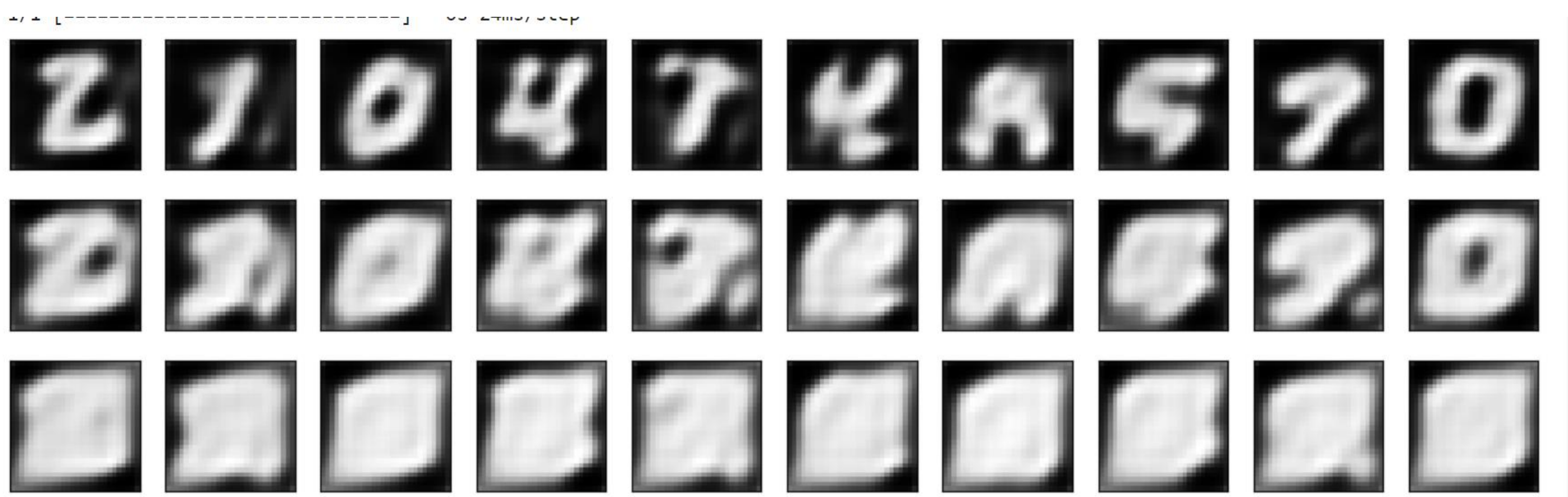
First step of denoising

3 more steps of denoising



Frist step of denoising

3 more steps of denoising



Let’s make this more stable, by training a network to just remove a little noise. It is like training to predict noise diffusion.

Stable Diffusion for Image Reconstruction

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. 2020.

- **Concept:**

create a sequence of images with noise, t=1…T

t=1 → t=T

Stable Diffusion for Image Reconstruction

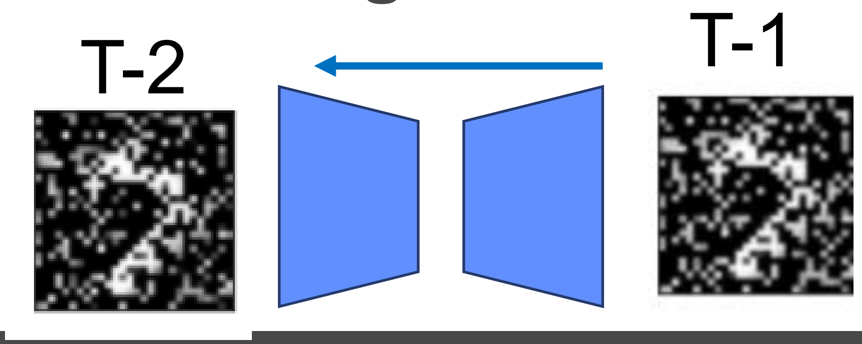
Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. 2020.

- **Concept:**

create a sequence of images with noise, t=1…T

train the network to reconstruct image t-1 from image t

etc…



Note: this example is in pixel space, but it is often applied in embedding space

- **From Ho et al. 2020**

Early denoising steps add overall structure
Later denoising steps add more detail

- end