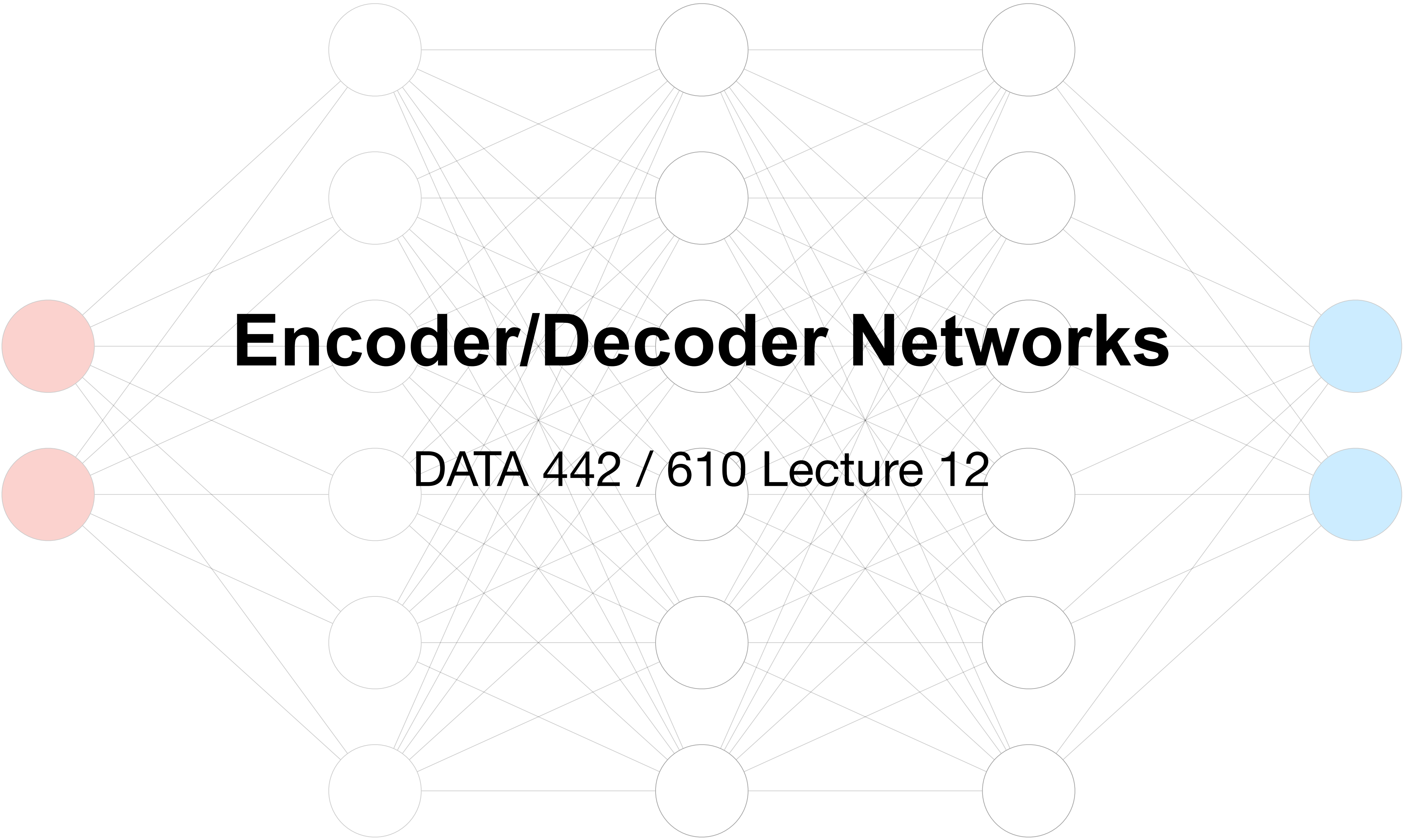




The diagram illustrates an Encoder/Decoder Network architecture. It features a central grid of 12 white circular nodes arranged in 4 rows and 3 columns. On the left side, there are 2 red circular nodes, and on the right side, there are 2 blue circular nodes. All nodes are interconnected by a dense web of thin gray lines, representing a fully connected network. The text 'Encoder/Decoder Networks' is centered over the network.

Encoder/Decoder Networks

DATA 442 / 610 Lecture 12

Outline

- Review CNN Topics
- Encoder/Decoder networks
- Introduction to RNNs

Reading: *Deep Learning*, Chapter 10

Training deep (convolutional) neural networks

ArXiv:1502.03167

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_1 \dots x_m\}$;

Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{ mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{ mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{ normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{ scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Deeper neural networks are more difficult to train

Batch normalization standardizes each layer's inputs to address *internal covariate shift*

Residual connections skip layers, improving training and making it easier to learn *identity* mappings

ArXiv:1512.03385

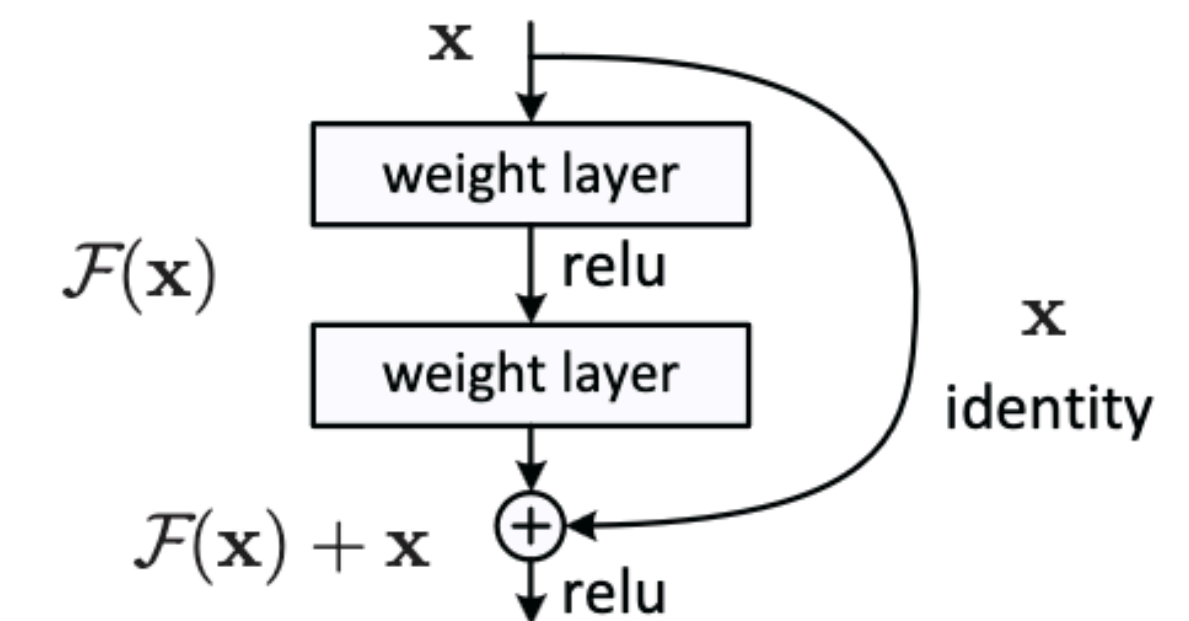


Figure 2. Residual learning: a building block.

A bit more on ResNet

Different residual block architectures have been proposed

A **bottleneck block** reduces the number of feature maps in each *spatial* (3x3) convolution

This allows deeper models with efficient memory footprints

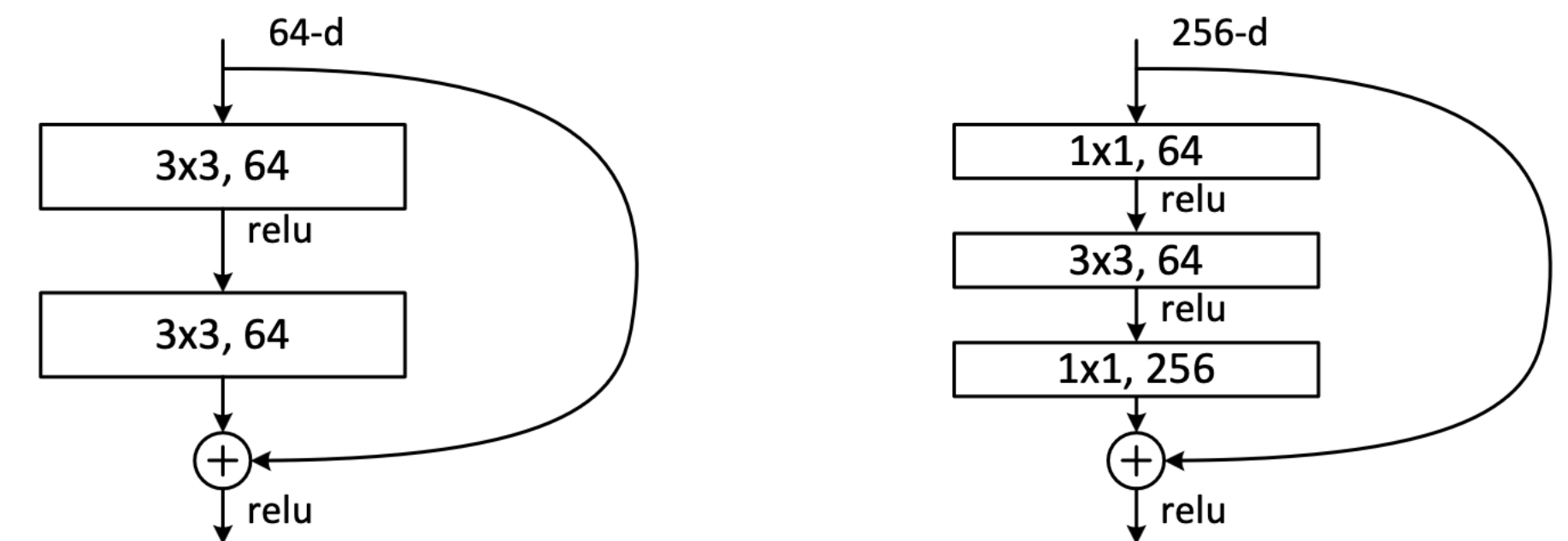


Figure 5. A deeper residual function $\mathcal{F}$ for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a “bottleneck” building block for ResNet-50/101/152.

A bit more on TorchVision

Pre-trained models

Training recipes

Code implementations

Table of all available classification weights

Accuracies are reported on ImageNet-1K using single crops:

Weight

AlexNet_Weights.IMAGENET1K_V1
ConvNeXt_Base_Weights.IMAGENET1K_V1
ConvNeXt_Large_Weights.IMAGENET1K_V1
ConvNeXt_Small_Weights.IMAGENET1K_V1
ConvNeXt_Tiny_Weights.IMAGENET1K_V1
DenseNet121_Weights.IMAGENET1K_V1
DenseNet161_Weights.IMAGENET1K_V1
DenseNet169_Weights.IMAGENET1K_V1
DenseNet201_Weights.IMAGENET1K_V1
EfficientNet_B0_Weights.IMAGENET1K_V1
EfficientNet_B1_Weights.IMAGENET1K_V1
EfficientNet_B1_Weights.IMAGENET1K_V2
EfficientNet_B2_Weights.IMAGENET1K_V1
EfficientNet_B3_Weights.IMAGENET1K_V1
EfficientNet_B4_Weights.IMAGENET1K_V1
EfficientNet_B5_Weights.IMAGENET1K_V1
EfficientNet_B6_Weights.IMAGENET1K_V1
EfficientNet_B7_Weights.IMAGENET1K_V1
EfficientNet_V2_L_Weights.IMAGENET1K_V1
EfficientNet_V2_M_Weights.IMAGENET1K_V1
EfficientNet_V2_S_Weights.IMAGENET1K_V1
GoogLeNet_Weights.IMAGENET1K_V1
Inception_V3_Weights.IMAGENET1K_V1
MNASNet0_5_Weights.IMAGENET1K_V1
MNASNet0_75_Weights.IMAGENET1K_V1
MNASNet1_0_Weights.IMAGENET1K_V1

<https://docs.pytorch.org/vis>

Image classification reference training scripts

This folder contains reference training scripts for image classification. They serve as a log of how to train specific models, as provide baseline training and evaluation scripts to quickly bootstrap research.

Except otherwise noted, all models have been trained on 8x V100 GPUs with the following parameters:

Parameter	value
--batch_size	32
--epochs	90
--lr	0.1
--momentum	0.9
--wd, --weight-decay	1e-4
--lr-step-size	30
--lr-gamma	0.1

AlexNet and VGG

Since AlexNet and the original VGG architectures do not include batch normalization, the default initial learning rate `--lr 0.1` is too high.

```
torchrun --nproc_per_node=8 train.py\
--model $MODEL --lr 1e-2
```

Here `$MODEL` is one of `alexnet`, `vgg11`, `vgg13`, `vgg16` or `vgg19`. Note that `vgg11_bn`, `vgg13_bn`, `vgg16_bn`, and `vgg19_bn` include batch normalization and thus are trained with the default parameters.

GoogLeNet

The weights of the GoogLeNet model are ported from the original paper rather than trained from scratch.

Inception V3

The weights of the Inception V3 model are ported from the original paper rather than trained from scratch.

Since it expects tensors with a size of $N \times 3 \times 299 \times 299$, to validate the model use the following command:

```
torchrun --nproc_per_node=8 train.py --model inception_v3\
```

<https://github.com/pytorch/vision/tree/main/references/classification>

```
class ResNet(nn.Module):
```

ock, Bottleneck]],

= False,

```
ion: Optional[list[bool]] = None,
able[..., nn.Module]] = None,
```

Norm2d ayer

```
ilation is None:
    tuple indicates if we should replace
    a dilated convolution instead
    ilation = [False, False, False]
    h_dilation) != 3:
```

```
ith_dilation should be None "  
tuple, got {replace_stride_with_dilation}"
```

```
er_group
    self.inplanes, kernel_size=7, stride=2, padding=3,
```

```
f.inplanes)
ce=True)
2d(kernel_size=3, stride=2, padding=1)
layer(block, 64, layers[0])
layer(block, 128, layers[1], stride=2,
on[0])
layer(block, 256, layers[2], stride=2,
on[1])
layer(block, 512, layers[3], stride=2,
on[2])
eAvgPool2d((1, 1))
    block.expansion, num_classes)
```

[torch/vision/blob/main/torchvision/models/
resnet.py](https://pytorch.org/docs/stable/torchvision/models/resnet.html)

Towards Transformers



Transformer components:

- Tokenization
- Embedding
- Encoder + decoder
- Attention mechanism

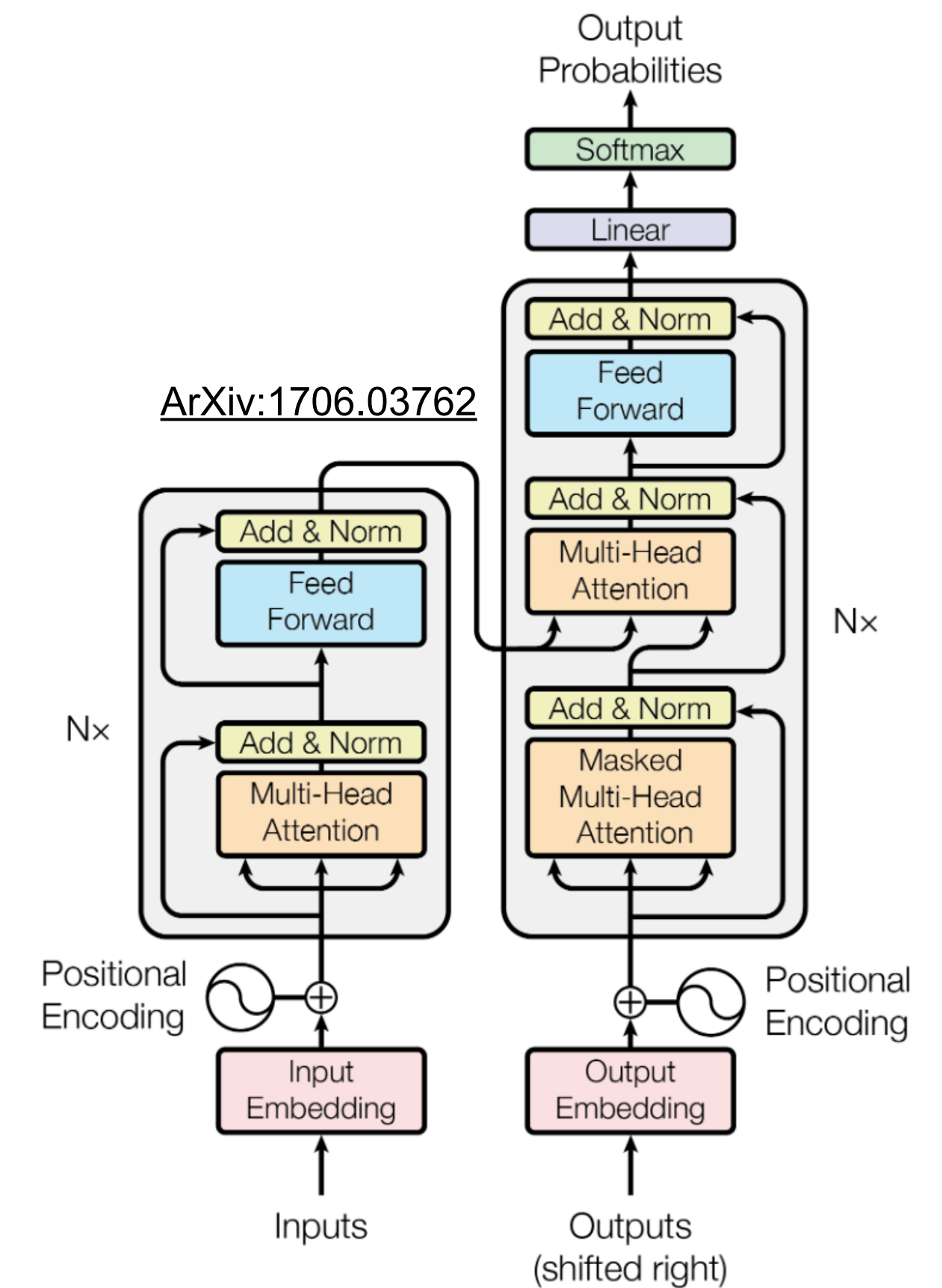
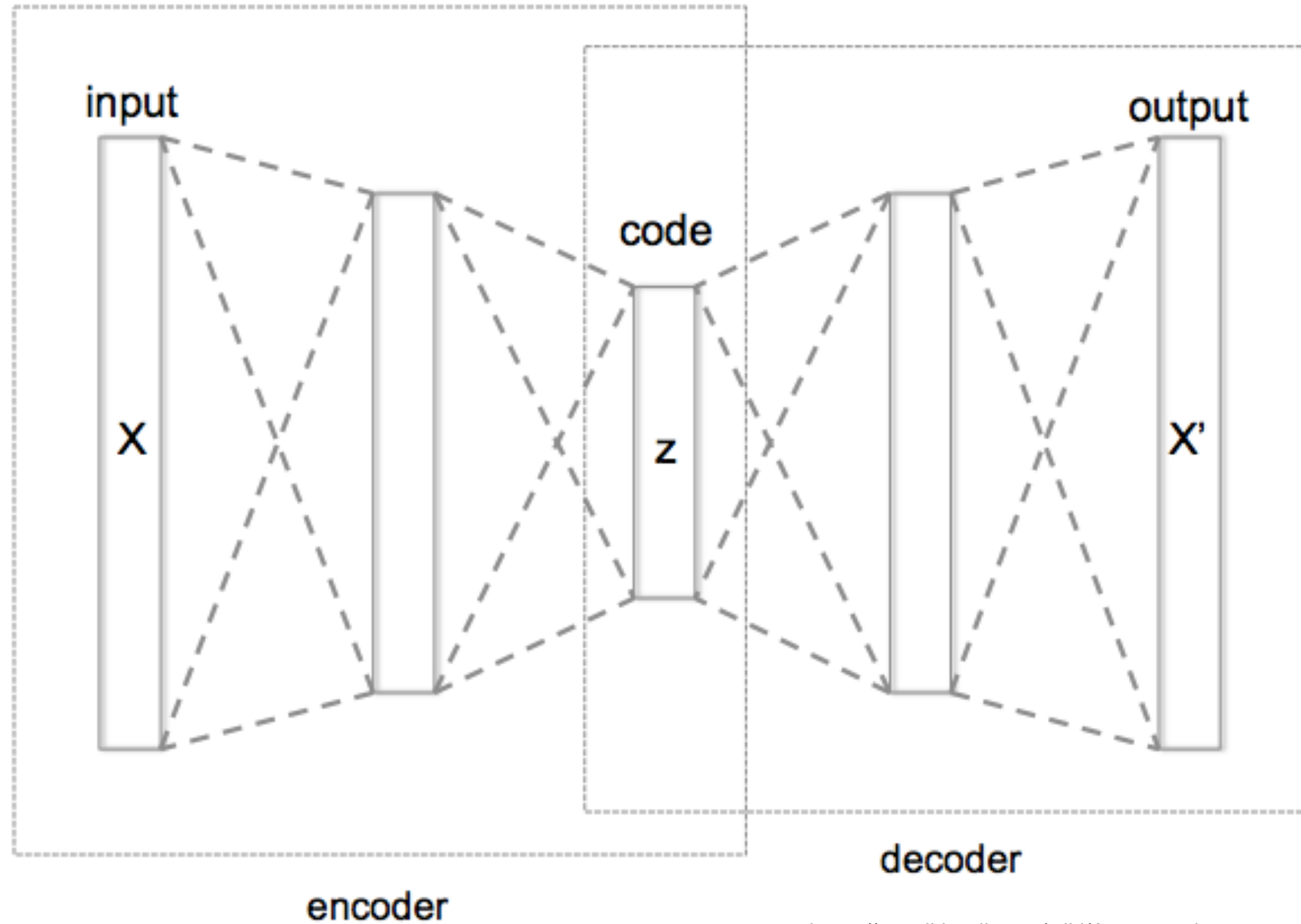


Figure 1: The Transformer - model architecture.

Encoder/Decoder networks



<https://en.wikipedia.org/wiki/Autoencoder>

Hands-on with Encoder/Decoder networks



That's all!

- Slides will be posted on the course website: <https://colen-teaching.github.io/DATA442-Fall-2026>
- Lab demos will also be posted
- **PS2** due October 23
- **REMEMBER**: TA Office Hours Tuesday 12-1pm, Student Success Center