# 02456 Deep Learning
# Mock exam

Technical University of Denmark

December 10, 2025

This document contains the exam questions. **All answers must be filled in on the *answer sheet*; do not hand in this document**. Please read the instructions on the *answer sheet* carefully.

**Question 1**

Consider two fully connected neural networks that map a scalar input to a scalar output:

- Network 1 has one hidden layer with $N$ units.
- Network 2 has $L$ hidden layers with $M$ units in each layer.

Assume that $N$, $L$ and $M$ are all positive integers and every neuron has a bias.

Recall from the book (Prince 2023, Problem 4.10) that a neural network with a single input, a single output, and $K$ hidden layers, each containing $D$ neurons has a total $3D + 1 + (K-1)D(D+1)$ parameters.

Which statement correctly describes when Network 1 has a larger number of parameters than Network 2?

A. When $N > \frac{(L-1)M^2+(L+1)M}{3}$.

B. When $N > \frac{(L-1)M^2+(L+2)M}{3}$.

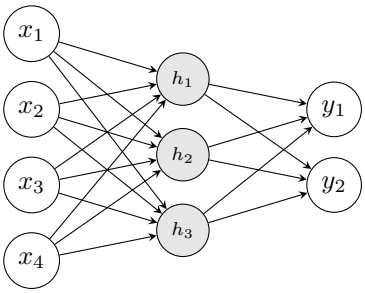
C. When $N = M$.

D. When $L = 1$.

E. Don't know.

**Question 2**

Consider the fully connected neural network shown below, where $\mathbf{x} \in \mathbb{R}^4$ is the input and $\mathbf{y} \in \mathbb{R}^2$ is the output.




All internal nodes use the sigmoid activation function, and there is no activation function on the output. Assuming that `nn` has been imported from `torch`, which one of the following PyTorch code snippets implements such a network correctly?

A.

```
nn.Sequential(
    nn.Linear(4, 3),
    nn.Sigmoid(),
    nn.Linear(3, 2),
)
```

B.

```
nn.Sequential(
    nn.Linear(2, 3),
    nn.Sigmoid(),
    nn.Linear(3, 4),
)
```

C.

```
nn.Sequential(
    nn.Linear(4, 3, 2),
    nn.Sigmoid(),
)
```

D.

```
nn.Sequential(
    nn.Linear(4, 3),
    nn.Sigmoid(),
    nn.Linear(3, 3),
    nn.Sigmoid(),
    nn.Linear(3, 2),
)
```

E. Don't know

**Question 3**

Consider a 2D max pooling layer with a kernel size of $(2, 2)$, no padding, and a stride of 2. The input to the layer is

$$X = \begin{bmatrix} 3 & 5 & 0 & 7 & 0 & 2 \\ 0 & 0 & 4 & 0 & 5 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 5 & 0 & 7 & 0 & 2 \\ 0 & 0 & 4 & 0 & 5 & 0 \end{bmatrix}. \tag{1}$$

What is the output of the layer?

A. $\begin{bmatrix} 5 & 7 \\ 5 & 7 \end{bmatrix}$

B. $\begin{bmatrix} 5 & 7 & 5 \\ 1 & 1 & 1 \\ 5 & 7 & 5 \end{bmatrix}$

C. $\begin{bmatrix} 5 & 5 & 7 & 7 & 5 \\ 1 & 1 & 4 & 5 & 5 \\ 1 & 1 & 1 & 1 & 1 \\ 5 & 5 & 7 & 7 & 2 \\ 5 & 5 & 7 & 7 & 5 \end{bmatrix}$

D. $\begin{bmatrix} 5 & 5 & 7 & 7 & 5 & 5 \\ 5 & 5 & 7 & 7 & 5 & 5 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 5 & 5 & 7 & 7 & 5 & 5 \\ 5 & 5 & 7 & 7 & 5 & 5 \end{bmatrix}$

E. Don't know

**Question 4**

Consider the following PyTorch model and loss function:

```
model = nn.Sequential(
    nn.Linear(4, 8),
    nn.ReLU(),
    nn.Linear(8, 2)
)
loss = nn.MSELoss()
```

Let `B` denote the batch size. For which of the following shapes of `input` and `target` will the expression `loss(model(input), target)` evaluate correctly?

A. `input : [B, 4], target : [B, 2]`

B. `input : [B, 4], target : [B]`

C. `input : [B, 8], target : [B, 2]`

D. `input : [B, 4], target : [2, B]`

E. Don't know

**Question 5**

Consider scaled dot-product attention

$$A = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_m}}\right)V.$$

The query $Q$, key $K$, and value $V$ tensors have shapes

$$Q \in \mathbb{R}^{64\times 256}, \quad K \in \mathbb{R}^{64\times 256}, \quad V \in \mathbb{R}^{64\times 128}.$$

What is the shape of the attention output $A$?

A. $\mathbb{R}^{256\times 128}$
B. $\mathbb{R}^{64\times 64}$
C. $\mathbb{R}^{256\times 64}$
D. $\mathbb{R}^{64\times 128}$
E. Don't know

**Question 6**

Recall that a layer has a *residual connection* if, for an input $\mathbf{h}_i$, the output of the layer is

$$\mathbf{h}_{i+1} = \mathbf{h}_i + f_i(\mathbf{h}_i), \tag{2}$$

where $f_i$ is a (learned) transformation such as a small neural network. Which one of the following PyTorch code snippets defines a layer with a residual connection?

A.

```
class Layer(nn.Module):
    def __init__(self, d_in, d_out):
        super().__init__()
        self.network = nn.Sequential(nn.Linear(d_in, d_out), nn.ReLU())
    def forward(self, x):
        return x + self.network(x)
```

B.

```
class Layer(nn.Module):
    def __init__(self, d_in, d_out):
        super().__init__()
        self.network = nn.Sequential(nn.Linear(d_in, d_out), nn.ReLU(), nn.
    Identity())
    def forward(self, x):
        return self.network(x)
```

C.

```
class Layer(nn.Module):
    def __init__(self, d_in, d_out):
        super().__init__()
        self.network = nn.Sequential(nn.Linear(d_in, d_out), nn.ReLU())
    def forward(self, x):
        return self.network(torch.cat([x, x], dim=-1))
```

D.

```
class Layer(nn.Module):
    def __init__(self, d_in, d_out):
        super().__init__()
        self.network = nn.Sequential(nn.Linear(d_in, d_out), nn.ReLU())
    def forward(self, x):
        return torch.cat([x, self.network(x)], dim=-1)
```

E. Don't know

**Question 7**

Recall that for a layer with a ReLU activation function, the variance of the weights under He and Glorot initialisation is given by

$$\sigma^2_{\text{He}} = \frac{2}{D_{\text{in}}}, \qquad \sigma^2_{\text{Glorot}} = \frac{4}{D_{\text{in}} + D_{\text{out}}}, \tag{3}$$

where $D_{\text{in}}$ and $D_{\text{out}}$ denote the input and output dimensionalities of the layer, respectively.

Consider a neural network with input dimension 28, a single hidden layer of size 128, and output dimension 10. If we initialise the weights of the *hidden layer* using Glorot initialisation, what variance should we use?

A. $\sigma^2 = \frac{4}{28+10}$

B. $\sigma^2 = \frac{4}{10+128}$

C. $\sigma^2 = \frac{4}{28+128}$

D. $\sigma^2 = \frac{4}{28+128+10}$

E. Don't know

**Question 8**

Consider the PyTorch LSTM layer

```
lstm = nn.LSTM(input_size=32, hidden_size=16)
x = torch.empty([64, 32])
z, (h_n, c_n) = lstm(x)
```

The input `x` has shape `[T, input_size]` with no batch dimension (i.e. an unbatched sequence of length `T = 64`). What is the shape of the LSTM output `z`?

A. `[64, 32]`

B. `[64, 16]`

C. `[1, 16]`

D. `[16, 64]`

E. Don't know

# References

Prince, SJ (2023). *Understanding Deep Learning*. MIT Press.