

 [Topics](#) [My posts](#) [More](#) [CATEGORIES](#)[General](#)[Site Feedback](#) [All categories](#)[Skip to main content](#)

# 2025 USA-NA-AIO Round 2, Problem 2, Part 5

USAAIO 

May 2025

## Part 5 (10 points, coding task)

In this part, you are asked to build your own multi-head attention module that subclasses `nn.Module`.

- For simplicity, we ignore any masking. That is, each position in an attending sequence attends to all positions in a being attended sequence.
- In your code, you do not need to worry about whether your code is efficient in an autoprogressive token generation process when your module is used in inference in a GPT-like task.

That is, if we use your code in a GPT-like task to autoprogressively generate tokens, it is totally fine if you repeatedly generate the same key and value at a given position rather than more efficiently storing their values in cache.

- The class name is `MyMHA`.
- Attributes:
  - `D_1` : Dimension of a hidden state/token in an attending sequence.
  - `D_2` : Dimension of a hidden state/token in a being attended sequence.
  - `D_v` : Dimension of a value vector.
  - `D_qk` : Dimension of a query/key vector.
  - `H` : Number of heads.

- $W_Q$  : A linear module whose weights is a query-projection matrix. The shape should be consistent with your answer in Part 2. No bias.
- $W_K$  : A linear module whose weights is key-projection matrix. The shape should be consistent with your answer in Part 2. No bias.
- $W_V$  : A linear module whose weights is value-projection matrix. The shape should be consistent with your answer in Part 2. No bias.
- $W_O$  : A linear module whose weights is an out-projection matrix. The shape should be consistent with your answer in Part 4. No bias.
- Method `__init__` :
  - Inputs
    - $D_1$
    - $D_2$
    - $D_{qk}$
    - $D_v$
    - $H$
  - Outputs
    - None
  - What to do inside this method
    - Initialize attribute values
- Method `forward` :

[Skip to main content](#)

- Inputs:
  - An attending sequence (tensor) with shape  $(B, L_1, D_1)$
  - A being addended sequence (tensor) with shape  $(B, L_2, D_2)$
- Outputs
  - Post-out-projection outputs with shape  $(B, L_1, D_1)$
- What to do inside this method
  - Compute the outputs
  - After each operation, add a comment on the tensor shape
  - **Do not use any loop**

---

**USAAIO** **May 2025**

```
### WRITE YOUR SOLUTION HERE ###
```

```
class MyMHA(nn.Module):
    def __init__(self, D_1, D_2, D_qk, D_v, H):
        super().__init__()
        self.D_1 = D_1
        self.D_2 = D_2
        self.D_qk = D_qk
        self.D_v = D_v
        self.H = H

        self.W_Q = nn.Linear(in_features=D_1, out_features=H*D_qk, bias=False)
        self.W_K = nn.Linear(in_features=D_2, out_features=H*D_qk, bias=False)
        self.W_V = nn.Linear(in_features=D_2, out_features=H*D_v, bias=False)
        self.W_O = nn.Linear(in_features=H*D_v, out_features=D_1, bias=False)
```

[Skip to main content](#)

```

def forward(self, x, y):
    B = x.shape[0] # batch size
    L_1 = x.shape[1] # the length of sequence x
    L_2 = y.shape[1] # the length of sequence y

    Q = self.W_Q(x) # shape: (B,L_1,H*D_qk)
    K = self.W_K(y) # shape: (B,L_2,H*D_qk)
    V = self.W_V(y) # shape: (B,L_2,H*D_v)

    Q = Q.reshape(B,L_1,self.H,self.D_qk) # shape: (B,L_1,H,D_qk)
    K = K.reshape(B,L_2,self.H,self.D_qk) # shape: (B,L_2,H,D_qk)
    V = V.reshape(B,L_2,self.H,self.D_v) # shape: (B,L_2,H,D_v)

    Q = Q.permute(0,2,1,3) # shape: (B,H,L_1,D_qk)
    K = K.permute(0,2,1,3) # shape: (B,H,L_2,D_qk)
    V = V.permute(0,2,1,3) # shape: (B,H,L_2,D_v)

    logits = Q @ K.transpose(-2,-1) / (self.D_qk**0.5) # shape: (B,H,L_1,L_2)
    alpha = torch.softmax(logits, dim=-1) # shape: (B,H,L_1,L_2)

    O = alpha @ V # shape: (B,H,L_1,D_v)

    O = O.permute(0,2,1,3) # shape: (B,L_1,H,D_v)
    O = O.reshape(B,L_1,self.H*self.D_v) # shape: (B,L_1,H*D_v)
    return self.W_O(O) # shape: (B,L_1,D_1)

""" END OF THIS PART """

```

[Skip to main content](#)

## ✦ Related topics

Topic

Replies

Activity

Topic	Replies	Activity
2025 USA-NA-AIO Round 2, Problem 2, Part 1	2	Dec 2025
2025 USA-NA-AIO Round 2, Problem 2, Part 7	1	May 2025
2025 USA-NA-AIO Round 2, Problem 2, Part 2	2	Dec 2025
2025 USA-NA-AIO Round 2, Problem 2, Part 13	1	May 2025
2025 USA-NA-AIO Round 2, Problem 2, Part 14	1	May 2025

 Powered by Discourse