University of Central Missouri

Department of Computer Science & Cybersecurity

CS5710 Machine Learning

Fall 2025

Home Assignment 5.

Student name: Nikhilesh Katakam
700772365

Submission Requirements:

- Once finished your assignment push your source code to your repo (GitHub) and explain the work through the ReadMe file properly. Make sure you add your student info in the ReadMe file.
- Comment your code appropriately ***IMPORTANT.***
- Any submission after provided deadline is considered as a late submission.

Part A — Short-Answer

1. **Positional Encoding Concepts**

a) Why do we need positional encodings in transformer models?

Transformers rely on self-attention, which views tokens as an unordered set — meaning attention alone can't tell the difference between different token orders. Positional encodings add signals about each token's location or relative distance, allowing the model to recognize sequence structure (like distinguishing "dog bites man" from "man bites dog"). Without these positional cues, the model would miss essential ordering information needed for language, time-series data, and other sequential tasks.

b) Describe two key requirements for a good positional encoding scheme.

- **Positions must be uniquely identifiable:** Each position needs a distinct encoding so the model can tell them apart and learn behaviours tied to specific locations in the sequence.
- **Support for new or longer sequences / useful relative cues:** Positional schemes should either extend smoothly to lengths not seen during training (as sinusoidal encodings do) or capture relative distances between tokens so the model can understand positional relationships even when absolute positions change.

c) What does it mean for the positional encoding matrix $M$to be *unitary* and *norm-preserving*?

- **Unitary property:** A matrix (P) is unitary (or orthogonal if it's real-valued) when (PTP = I.). This ensures the transformation keeps inner products and angles intact.
- **Norm preservation:** Using such an encoding leaves the vector lengths unchanged, preventing unintended scaling of activations that could disrupt training or alter attention scores. A unitary positional mapping therefore maintains both vector norms and the overall geometric relationships (angles and dot products) between embeddings.

2. **Attention Mechanism**

a) Define "attention score" and explain how it determines the weight of each token.

An *attention score* is an alignment scalar that measures compatibility between a query (usually derived from the current token or decoder state) and a key (representing another token). Commonly computed as dot(Q, K) (sometimes scaled by $\sqrt{d_k}$). These scores are converted into normalized weights (via softmax) so that tokens with higher compatibility receive higher attention weight (i.e., influence the output more).

b) What mathematical operation is applied to convert alignment scores into attention weights?

Use the SoftMax over the scores associated with each query to turn the raw alignment values into a proper probability distribution—one where all weights are non-negative and add up to 1.

c) How is the *context vector* computed from these weights and values?

The **context vector** (or attention output) is the weighted sum of the value vectors $V$where weights are the softmax-normalized attention weights. Mathematically: Context = softmax$(QK^{\mathsf{T}}/\sqrt{d_k})V$.

3. **Multi-Head Attention**

a) What is the main advantage of using multiple attention heads?

Using multiple heads allows the model to focus on different aspects of the representation space and different locations in the sequence at the same time. For example, one head might capture syntax while another picks up long-distance semantic connections. This gives the model richer and more diverse attention patterns than relying on a single attention mechanism.

b) How does splitting Q, K, and V across different subspaces improve model representation?

By projecting Q/K/V into smaller subspaces (via separate learned linear maps) each head focuses on distinct features and patterns. This enables specialized attention: different heads can focus on different distance ranges, token types, or syntactic functions, then their outputs are combined to form a richer representation.

c) After multi-head attention, why is concatenation followed by another linear projection necessary?

Concatenating all head outputs gathers their diverse information into a single vector. A subsequent learned linear layer then blends these features, allowing the model to integrate what each head captured and produce a unified, enriched final representation.

4. Ethical Foundations

a) Explain why *ethics* is not the same as *laws* or *feelings*.

- **Law vs. ethics:** Laws are formal rules backed by institutions, while ethics is a wider, philosophical inquiry into what is morally right or wrong. Ethical judgment can extend beyond what the law permits—an action may be legal yet morally questionable, or illegal but viewed by some as ethically defensible.
- **Feelings vs. ethics:** Feelings are individual and subjective, whereas ethics relies on structured reasoning—principles, consistency, and consideration of consequences. Ethical evaluation requires rational justification, not just emotional reaction.

b) Briefly describe two classical ethical theories (e.g., utilitarianism and deontology) and how they would handle an AI decision scenario.

Utilitarianism (consequentialism): Focuses on actions that produce the greatest overall benefit. In this hiring scenario, it would favor whatever policy leads to the highest combined fairness and productivity. This might support a demographic-neutral scoring system if it yields the best overall outcomes, though broader social impacts must still be considered.

Deontology (duty-based ethics): Evaluates actions by whether they follow moral duties or rules—such as a strict obligation not to discriminate—regardless of the consequences. Under this view, the system should not use protected attributes or their proxies, even if doing so could improve predictive accuracy.

C) Why do philosophers argue that no single ethical theory clearly "wins" in all contexts?

Different ethical frameworks emphasize different moral values—such as consequences, rights, duties, or character. In practice, real decisions require balancing these competing priorities (like efficiency, fairness, and individual rights), so no single theory can fully resolve every situation on its own.

5. Types of AI Harms

a) Define allocational harm and representational harm in AI systems.

. **Allocational harm:** When automated systems unevenly allocate opportunities or resources—such as a hiring algorithm that disproportionately screens out qualified candidates from marginalized groups.

. **Representational harm:** When a system reinforces stereotypes, erases identities, or produces biased portrayals—for example, gender-skewed translations or facial recognition that performs poorly on certain skin tones, affecting dignity and cultural visibility.

b) Provide an example of each from real-world applications (e.g., translation, hiring, or facial recognition).

. **Allocational:** An automated hiring tool that consistently scores applicants from an underrepresented group lower, resulting in reduced chances of being invited to interviews.

. **Representational:** A translation system that turns gender-neutral phrases into gendered ones based on biased assumptions, reinforcing inaccurate stereotypes.

c) Why is representational harm often harder to measure than allocational harm?

. Allocational harms show up in concrete, measurable results—like differences in who gets hired or approved for loans. Representational harms, on the other hand, relate to issues of identity, dignity, and cultural meaning; they're more context-sensitive, harder to quantify, and usually evaluated through qualitative judgment rather than straightforward metrics.

6. Sources of Dataset Bias

a) List three reasons why bias arises during data collection or annotation in AI datasets.

. **Sampling bias:** The data collected doesn't reflect the full population you want to model—for example, web corpora dominated by U.S. and English content.

. **Labeler bias:** Human annotators inject their own cultural assumptions or subjective judgments into the labels.

. **Measurement/collection bias:** The tools or processes used to gather data—such as sensors, interfaces, or scraping methods—systematically overlook or distort certain groups.

b) What kinds of data or groups tend to be under-represented in large language datasets?

. Languages with few resources, marginalized communities, and non-Western cultural material tend to appear far less often in datasets, leaving them underrepresented.

c) How can bias amplification occur even after initial data preprocessing?

. When training data are imbalanced or the loss function pushes the model to prioritize frequent patterns (as cross-entropy often does), existing biases can get amplified. The model may end up leaning even more heavily toward majority patterns than the data alone would imply, strengthening those biases during learning.

7. Safety, Security, and Privacy

a) Define data poisoning and describe how it can manipulate a model's predictions.

Data poisoning involves adding carefully crafted malicious examples to the training set so the final model behaves incorrectly on targeted inputs. This shifts the model's decision boundaries in ways that cause misclassification or other harmful behaviours—such as tricking it into labelling a specific sample incorrectly or revealing sensitive information.

b) What are the ethical implications of model memorization (e.g., GPT-2 reproducing private or copyrighted text)?

. If a model memorizes and outputs private or copyrighted material (such as verbatim passages resembling GPT-2-style leakage), it can breach both privacy and intellectual property protections. This raises ethical concerns like revealing personal data, exposing proprietary content, and creating potential legal risks for whoever deploys the model.

C) How does model stealing threaten privacy and intellectual property in AI research?

Model stealing uses query access to mimic a model's behaviour, allowing an attacker to recreate it without paying the training costs. This compromises the creator's intellectual property, diminishes the value of their work, and can also open the door to misuse of the duplicated system.

Part B — Coding

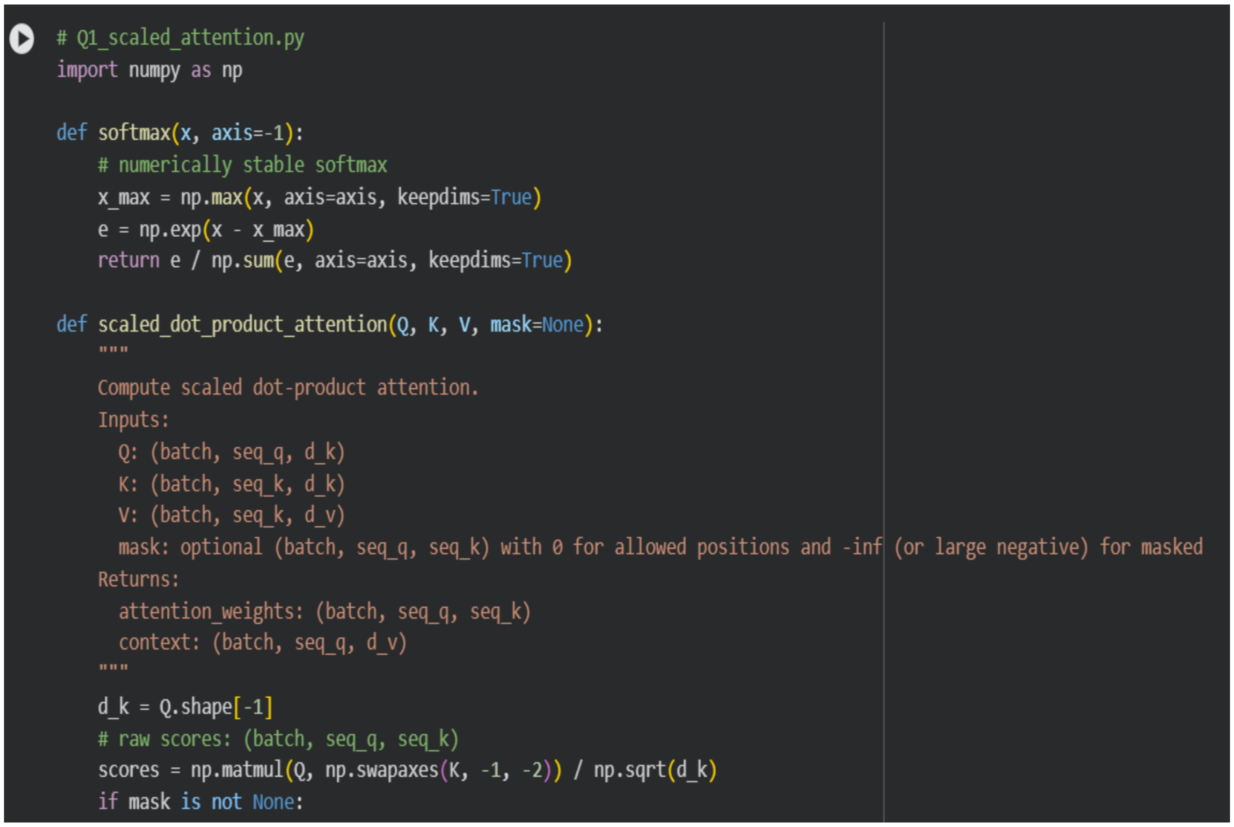
Q1. Compute Scaled Dot-Product Attention (Python)
Write a Python function to compute the scaled dot-product attention given query Q, key K, and value Vmatrices.

- Use NumPy for matrix operations.
- Normalize scores using softmax.
- Return both attention weights and the resulting context vector.

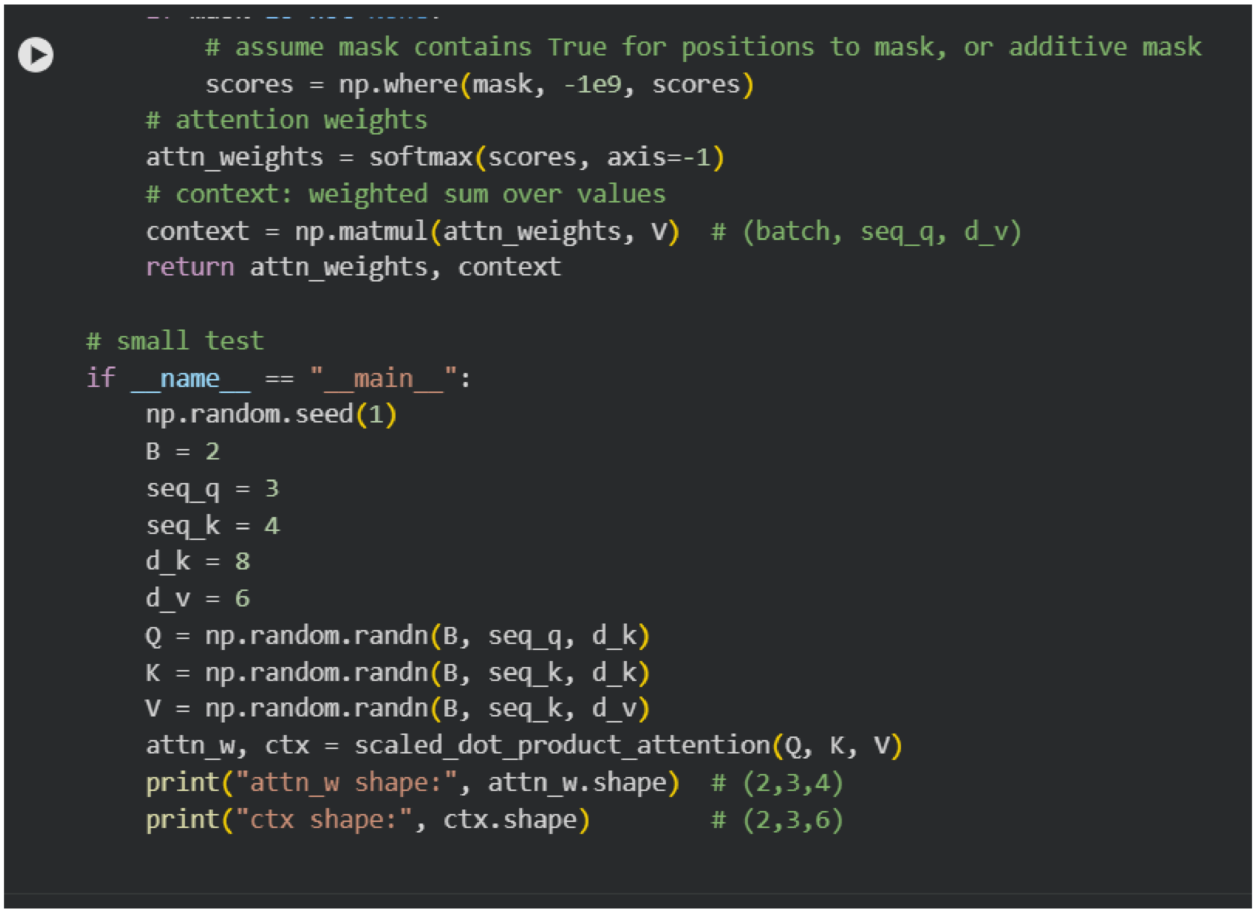
Hint:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Code:



```
# Q1_scaled_attention.py
import numpy as np

def softmax(x, axis=-1):
    # numerically stable softmax
    x_max = np.max(x, axis=axis, keepdims=True)
    e = np.exp(x - x_max)
    return e / np.sum(e, axis=axis, keepdims=True)

def scaled_dot_product_attention(Q, K, V, mask=None):
    """
    Compute scaled dot-product attention.
    Inputs:
      Q: (batch, seq_q, d_k)
      K: (batch, seq_k, d_k)
      V: (batch, seq_k, d_v)
      mask: optional (batch, seq_q, seq_k) with 0 for allowed positions and -inf (or large negative) for masked
    Returns:
      attention_weights: (batch, seq_q, seq_k)
      context: (batch, seq_q, d_v)
    """
    d_k = Q.shape[-1]
    # raw scores: (batch, seq_q, seq_k)
    scores = np.matmul(Q, np.swapaxes(K, -1, -2)) / np.sqrt(d_k)
    if mask is not None:
```



```
        # assume mask contains True for positions to mask, or additive mask
        scores = np.where(mask, -1e9, scores)
    # attention weights
    attn_weights = softmax(scores, axis=-1)
    # context: weighted sum over values
    context = np.matmul(attn_weights, V)  # (batch, seq_q, d_v)
    return attn_weights, context

# small test
if __name__ == "__main__":
    np.random.seed(1)
    B = 2
    seq_q = 3
    seq_k = 4
    d_k = 8
    d_v = 6
    Q = np.random.randn(B, seq_q, d_k)
    K = np.random.randn(B, seq_k, d_k)
    V = np.random.randn(B, seq_k, d_v)
    attn_w, ctx = scaled_dot_product_attention(Q, K, V)
    print("attn_w shape:", attn_w.shape)  # (2,3,4)
    print("ctx shape:", ctx.shape)        # (2,3,6)
```

OUTPUT

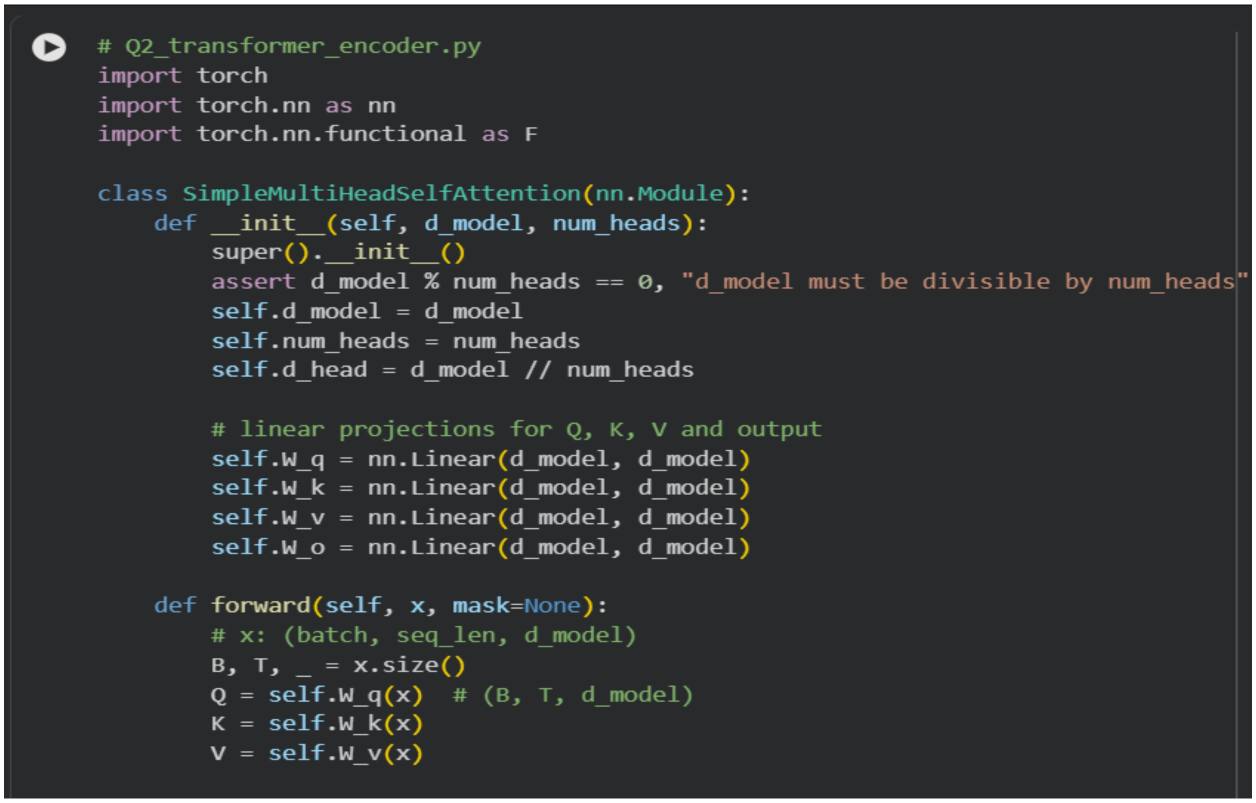
```
attn_w shape: (2, 3, 4)

ctx shape: (2, 3, 6)
```

Q2. Implement Simple Transformer Encoder Block (PyTorch)
Implement a simplified transformer encoder block in PyTorch with the following components:

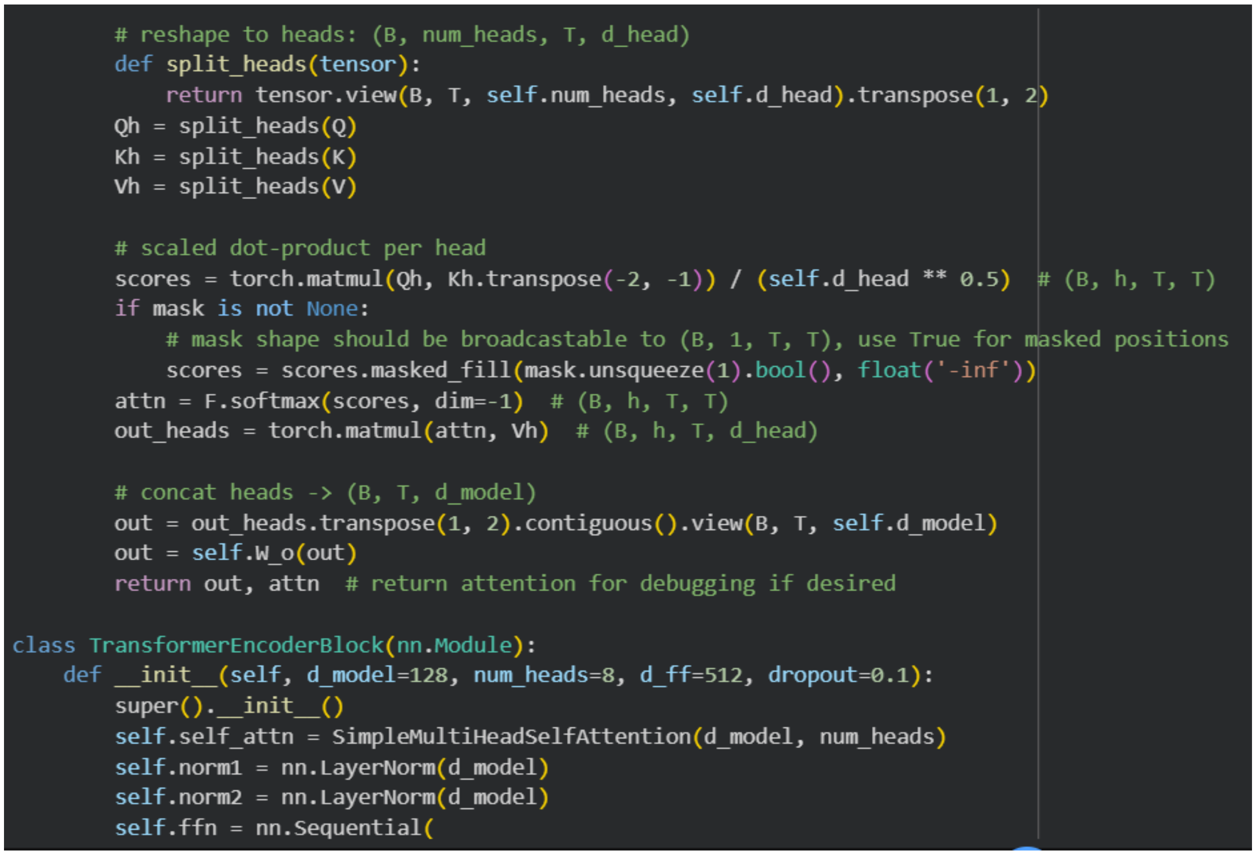
- Multi-head self-attention layer
- Feed-forward network (2 linear layers with ReLU)
- Add & Norm layers

Sub-tasks:
a) Initialize dimensions $d_{model} = 128, h = 8$.
b) Add residual connections and layer normalization.
c) Verify the output shape for a batch of 32 sentences, each with 10 tokens.

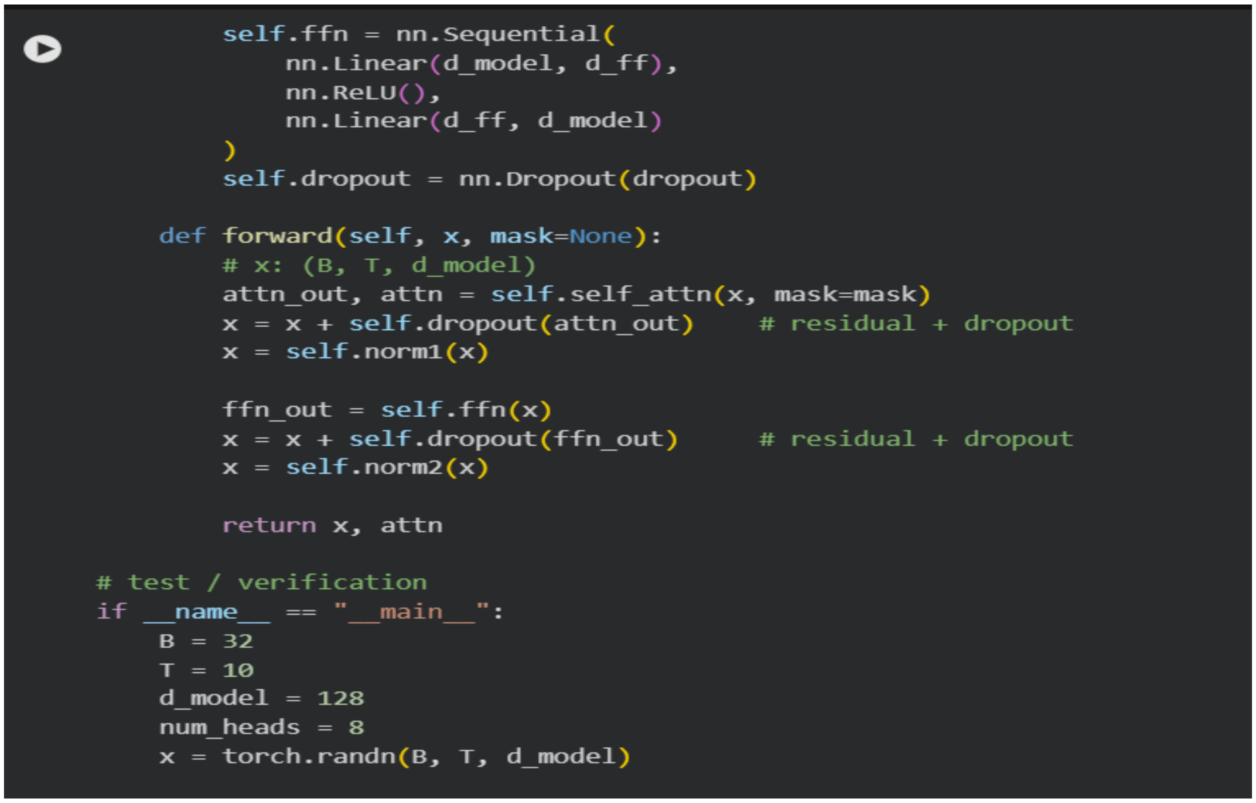
CODE



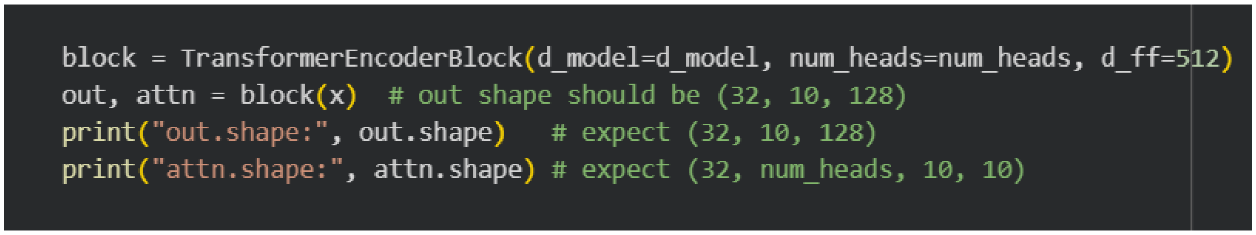
```python
# Q2_transformer_encoder.py
import torch
import torch.nn as nn
import torch.nn.functional as F

class SimpleMultiHeadSelfAttention(nn.Module):
    def __init__(self, d_model, num_heads):
        super().__init__()
        assert d_model % num_heads == 0, "d_model must be divisible by num_heads"
        self.d_model = d_model
        self.num_heads = num_heads
        self.d_head = d_model // num_heads

        # linear projections for Q, K, V and output
        self.W_q = nn.Linear(d_model, d_model)
        self.W_k = nn.Linear(d_model, d_model)
        self.W_v = nn.Linear(d_model, d_model)
        self.W_o = nn.Linear(d_model, d_model)

    def forward(self, x, mask=None):
        # x: (batch, seq_len, d_model)
        B, T, _ = x.size()
        Q = self.W_q(x)  # (B, T, d_model)
        K = self.W_k(x)
        V = self.W_v(x)
```



```python
        # reshape to heads: (B, num_heads, T, d_head)
        def split_heads(tensor):
            return tensor.view(B, T, self.num_heads, self.d_head).transpose(1, 2)
        Qh = split_heads(Q)
        Kh = split_heads(K)
        Vh = split_heads(V)

        # scaled dot-product per head
        scores = torch.matmul(Qh, Kh.transpose(-2, -1)) / (self.d_head ** 0.5)  # (B, h, T, T)
        if mask is not None:
            # mask shape should be broadcastable to (B, 1, T, T), use True for masked positions
            scores = scores.masked_fill(mask.unsqueeze(1).bool(), float('-inf'))
        attn = F.softmax(scores, dim=-1)  # (B, h, T, T)
        out_heads = torch.matmul(attn, Vh)  # (B, h, T, d_head)

        # concat heads -> (B, T, d_model)
        out = out_heads.transpose(1, 2).contiguous().view(B, T, self.d_model)
        out = self.W_o(out)
        return out, attn  # return attention for debugging if desired

class TransformerEncoderBlock(nn.Module):
    def __init__(self, d_model=128, num_heads=8, d_ff=512, dropout=0.1):
        super().__init__()
        self.self_attn = SimpleMultiHeadSelfAttention(d_model, num_heads)
        self.norm1 = nn.LayerNorm(d_model)
        self.norm2 = nn.LayerNorm(d_model)
        self.ffn = nn.Sequential(
```



```python
        self.ffn = nn.Sequential(
            nn.Linear(d_model, d_ff),
            nn.ReLU(),
            nn.Linear(d_ff, d_model)
        )
        self.dropout = nn.Dropout(dropout)

    def forward(self, x, mask=None):
        # x: (B, T, d_model)
        attn_out, attn = self.self_attn(x, mask=mask)
        x = x + self.dropout(attn_out)    # residual + dropout
        x = self.norm1(x)

        ffn_out = self.ffn(x)
        x = x + self.dropout(ffn_out)     # residual + dropout
        x = self.norm2(x)

        return x, attn

# test / verification
if __name__ == "__main__":
    B = 32
    T = 10
    d_model = 128
    num_heads = 8
    x = torch.randn(B, T, d_model)
```



```python
    block = TransformerEncoderBlock(d_model=d_model, num_heads=num_heads, d_ff=512)
    out, attn = block(x)  # out shape should be (32, 10, 128)
    print("out.shape:", out.shape)   # expect (32, 10, 128)
    print("attn.shape:", attn.shape) # expect (32, num_heads, 10, 10)
```

OUTPUT

```
out.shape: torch.Size([32, 10, 128])

attn.shape: torch.Size([32, 8, 10, 10])
```

CODE LINK: https://colab.research.google.com/drive/1y79sSStnhVUcV0vi_m-vWxqQozIkk35L?usp=drive_link

GITHUB LINK: https://github.com/nikhlilesh193/Machine-Learning--HomeAssignment-5.git