AML_LAB_ASSIGNMENT - 02-1

March 28, 2024

1 Transformer Architecture Lab Assignment

This lab assignment is designed to give you practical experience with building and understanding the Transformer architecture, as proposed by Vaswani et al. in "Attention is All You Need". The assignment is divided into several sections, each focusing on a key component of the Transformer model. You will implement these components from scratch, following the TODOs provided.

1.1 Learning Objectives

- Understand the internals of the Transformer architecture.
- Implement the Scaled Dot-Product Attention mechanism.
- Build the Multi-Head Attention mechanism.
- Assemble the Transformer Encoder Block.
- Integrate Positional Encoding.

1.2 Before You Start

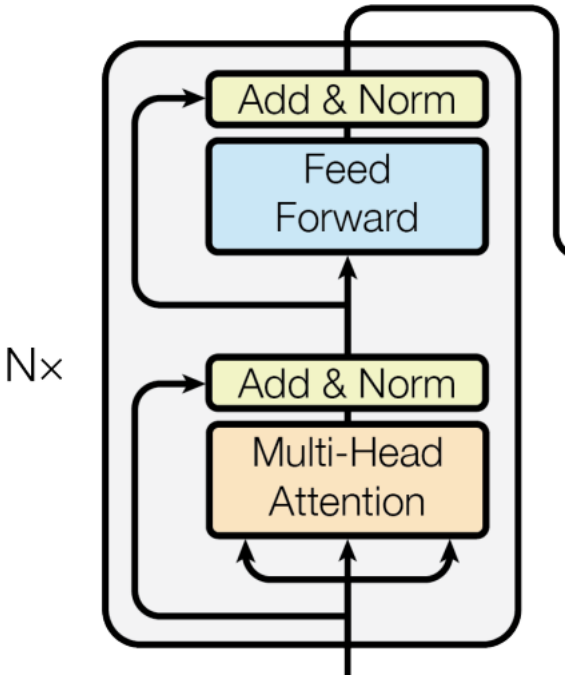
Make sure you have a basic understanding of the following concepts: - PyTorch or another deep learning framework. - The concept of neural networks, especially feed-forward and attention mechanisms. - How gradient backpropagation works.

1.3 References for Learning

- "Attention is All You Need", Vaswani et al., 2017 (Original Transformer paper).
- The Illustrated Transformer, Jay Alammar. Provides an excellent visual and intuitive understanding of the model.
- PyTorch Official Tutorials for a refresher on PyTorch.

1.4 Section 1: Scaled Dot-Product Attention

Scaled Dot-Product Attention is a fundamental mechanism at the heart of the Transformer architecture, introduced in the paper "Attention Is All You Need" by Vaswani et al. This mechanism allows the model to dynamically focus on different parts of the input sequence when performing a task, making it very effective for tasks that benefit from understanding the relationships and relative positioning of tokens in the sequence, such as language understanding and generation.



How It Works The mechanism computes attention scores based on queries (Q), keys (K), and values (V) derived from the input data, typically through learned linear transformations. The attention scores determine how much focus should be placed on each part of the input data when producing the output for a particular token.

The operation is defined as follows:

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

Q, K, V: These are matrices created from the input embeddings, where Q represents the queries, K the keys, and V the values. In the context of self-attention, these are derived from the same input sequence but transformed differently.

QKˆT: This is the dimensionality of the keys (and queries). The scaling factor $\sqrt{d_k}$ is used to avoid extremely large values of the dot product, especially in higher dimensions, which can lead to vanishing gradients during training. This scaling helps stabilize the gradient descent optimization process.

Softmax: The softmax function is applied to the rows of the dot product matrix, turning them into probabilities that sum to 1. This operation determines how much each value is expressed at a certain position in the output sequence, effectively allowing the model to focus on relevant parts of the input.

[1]:
```
import torch.nn.functional as F
import torch

def scaled_dot_product(q, k, v):
```

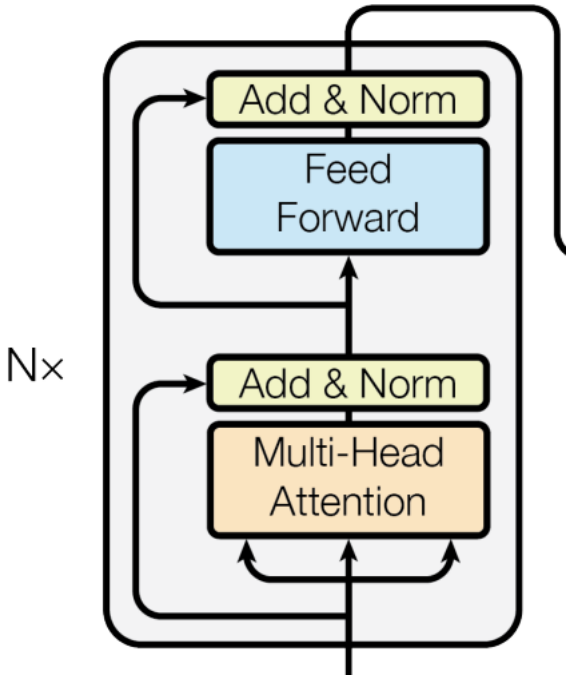
```
    d_k = q.size(-1)
    scores = torch.matmul(q, k.transpose(-2, -1)) / torch.sqrt(torch.tensor(d_k,␣
 ↪dtype=torch.float32))
    attention_weights = F.softmax(scores, dim=-1)
    output = torch.matmul(attention_weights, v)
    return output, attention_weights

# Initialize the seed and tensors as provided
torch.manual_seed(42)
seq_len, d_k = 3, 2
q = torch.randn(seq_len, d_k)
k = torch.randn(seq_len, d_k)
v = torch.randn(seq_len, d_k)

values, attention = scaled_dot_product(q, k, v)

print("Q\n", q)
print("K\n", k)
print("V\n", v)
print("Values\n", values)
print("Attention\n", attention)
```

```
Q
 tensor([[ 0.3367,  0.1288],
        [ 0.2345,  0.2303],
        [-1.1229, -0.1863]])
K
 tensor([[ 2.2082, -0.6380],
        [ 0.4617,  0.2674],
        [ 0.5349,  0.8094]])
V
 tensor([[ 1.1103, -1.6898],
        [-0.9890,  0.9580],
        [ 1.3221,  0.8172]])
Values
 tensor([[ 0.5698, -0.1520],
        [ 0.5379, -0.0265],
        [ 0.2246,  0.5556]])
Attention
 tensor([[0.4028, 0.2886, 0.3086],
        [0.3538, 0.3069, 0.3393],
        [0.1303, 0.4630, 0.4067]])
```

1.5 Section 2: Multi-Head Attention

Multi-Head Attention is a pivotal innovation introduced in the Transformer model, aimed at enhancing the model's ability to focus on different parts of the input sequence for a given task. It operates on the principle of dividing the attention mechanism into multiple "heads," enabling the

model to simultaneously attend to information from different representation subspaces at different positions. This parallel attention processing allows the model to capture a more complex interplay of features within the input.



Explanation: Initialization: This class initializes linear layers for projecting the queries, keys, values, and the final output. It also calculates d_k, the dimension of each head.

Splitting Heads: The split_heads method reshapes the input tensors to separate heads, allowing the model to process parts of the input in parallel across different representation subspaces.

Forward Pass: In the forward method, inputs are first projected, then split into multiple heads. The scaled dot-product attention is computed for each head. The outputs of all heads are then concatenated and passed through a final linear projection layer.

This code snippet provides a basic framework for implementing Multi-Head Attention. In practice, you might need to adjust it based on your specific requirements, including handling padding masks for variable-length sequences.

[2]:
```
import torch
import torch.nn as nn
import torch.nn.functional as F

class MultiHeadAttention(nn.Module):
    def __init__(self, d_model, num_heads):
        super(MultiHeadAttention, self).__init__()
        self.num_heads = num_heads
        self.d_model = d_model
        self.d_k = d_model // num_heads
```

```
        self.query_projection = nn.Linear(d_model, d_model)
        self.key_projection = nn.Linear(d_model, d_model)
        self.value_projection = nn.Linear(d_model, d_model)
        self.final_projection = nn.Linear(d_model, d_model)

        self.scale = torch.sqrt(torch.FloatTensor([self.d_k]))

    def split_heads(self, x, batch_size):

        #Todo 1.1
        # Split the last dimension into (num_heads, depth)
        x = x.view(batch_size,-1,self.num_heads,self.d_k)
        #Todo 1.2
        return x.permute(0,2,1,3)  # (batch_size, num_heads, seq_length, depth)

    def forward(self, query, key, value, mask=None):
        batch_size = query.size(0)

        # Get query, key and value from responding projection layer
        query = self.query_projection(query)
        key = self.key_projection(key)
        value = self.value_projection(value)



        # Split query, key and value to multiple heads
        #Todo 1.3
        query = self.split_heads(query,batch_size)
        key = self.split_heads(key,batch_size)
        value = self.split_heads(value,batch_size)

        # Self attention
        scores = torch.matmul(query, key.transpose(-2, -1)) / self.scale

        if mask is not None:
            # Todo 1.4
            #Sets attention scores to -inf where mask is 0. This prevents masked␣
↪elements from influencing the attention output by making them negligible after␣
↪softmax.
            scores = scores.masked_fill(mask == 0, float('-inf'))#code here

        #Applies softmax to the scores, converting them into a probability␣
↪distribution that sums to 1, representing the importance of each value.
        attention_weights = F.softmax(scores,dim= -1)
```

```
        #Multiplies attention weights by the value vectors to get the final␣
↪attention output, effectively selecting or highlighting information based on␣
↪computed importance.
        attention_output = torch.matmul(attention_weights, value)

        attention_output = attention_output.permute(0, 2, 1, 3).contiguous()

        # Concatenate heads
        #Todo 1.5
        # Reshape the attention output tensor to concatenate the heads

        attention_output = attention_output.view(batch_size, -1, self.d_model) ␣
↪# Concatenate heads



        # Final projection
        #Todo 1.6
        # Perform the final projection to transform the concatenated tensor
        output = self.final_projection(attention_output)

        return output, attention_weights
```

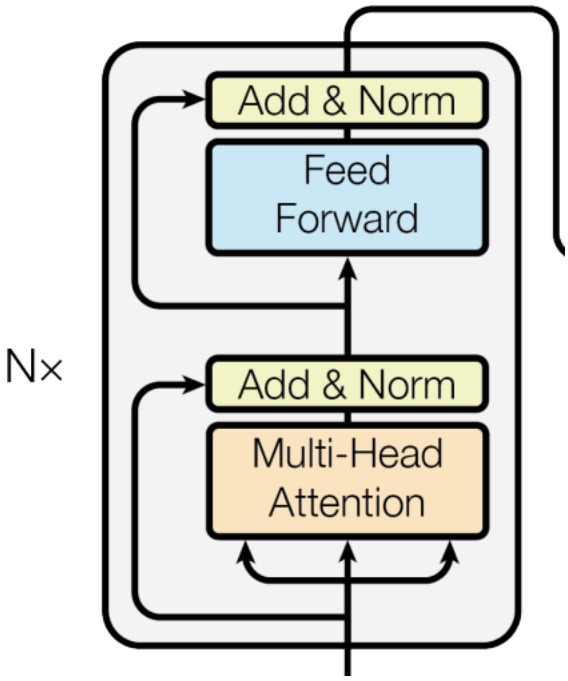
Test the Multi Head Attention module

[3]:
```
# Example instantiation
d_model = 512  # Embedding size
num_heads = 8
mha = MultiHeadAttention(d_model, num_heads)

# inputs
seq_length = 60
batch_size = 20

#Todo 1.7
#pass the values batch_size, seq_length, d_model
dummy_query = torch.rand(batch_size, seq_length, d_model)
dummy_key = torch.rand(batch_size, seq_length, d_model)
dummy_value = torch.rand(batch_size, seq_length, d_model)

# Forward pass
output, attention_weights = mha(dummy_query, dummy_key, dummy_value)
print(output.shape)  # Expected shape: (batch_size, seq_length, d_model)
print(attention_weights.shape)  # Shape: (batch_size, num_heads, seq_length,␣
↪seq_length)
```

```
torch.Size([20, 60, 512])
torch.Size([20, 8, 60, 60])
```

1.6 Section 3: Transformer Encoder Block

The Transformer Encoder Block is a core component of the Transformer architecture, specifically within the encoder part of the model. It is designed to process the input sequence in a way that captures both the individual meaning of each token and the context provided by the rest of the sequence. Each encoder block is composed of several key components that work together to achieve this goal.



1.7 Explanation:

Multi-Head Attention: The encoder block starts with a multi-head attention layer that allows the model to focus on different parts of the input sequence.

Feedforward Network: After attention, the block applies a position-wise feedforward network. It's a simple fully connected neural network applied to each position separately and identically.

Normalization and Dropout: Before and after each of these main layers, the block applies dropout followed by layer normalization as a form of regularization and to stabilize the learning process.

Residual Connections: There are residual connections around each of the main layers (the multi-head attention and the feedforward network). This helps in mitigating the vanishing gradient problem in deep networks.

This code snippet assumes that the MultiHeadAttention class has been defined as shown in the previous example. The Transformer Encoder Block is a fundamental building block of Transformer-based models, encapsulating the core ideas of multi-head self-attention and position-wise transformation within a layer normalization and residual connection framework.

[4]:
```
import torch
import torch.nn as nn
import torch.nn.functional as F

class TransformerEncoderBlock(nn.Module):
    def __init__(self, d_model, num_heads, d_ff, dropout_rate=0.1):
        super(TransformerEncoderBlock, self).__init__()
        self.multi_head_attention = MultiHeadAttention(d_model, num_heads)

        """To do 2.1, implement a feed_forward layer.
        The architecture is Linear+ReLU+Dropout+Linear
        """

        self.feed_forward = nn.Sequential(nn.Linear(d_model, d_ff),
            nn.ReLU(),
            nn.Dropout(dropout_rate),
            nn.Linear(d_ff, d_model)

        )

        self.norm1 = nn.LayerNorm(d_model)
        self.norm2 = nn.LayerNorm(d_model)
        self.dropout1 = nn.Dropout(dropout_rate)
        self.dropout2 = nn.Dropout(dropout_rate)

    def forward(self, x, mask=None):

        """To do 2.2: implement the whole forward
        """

        # Multi-Head Attention
        attn_output, _ = self.multi_head_attention(x, x, x, mask)
        x = x + self.dropout1(attn_output)  # Apply dropout to the output of the␣
↪attention layer
        x = self.norm1(x)  # Apply LayerNorm

        # Feedforward
        ff_output = self.feed_forward(x)
        x = x + self.dropout2(ff_output)  # Apply dropout to the output of the␣
↪feedforward layer
        x = self.norm2(x)  # Apply LayerNorm

        return x
```

Test the Encoderblock module

[5]:
```
# Example instantiation
d_model = 512  # Embedding size
num_heads = 8
d_ff = 2048  # Size of the FFN layer

#Todo 3
encoder_block = TransformerEncoderBlock(d_model, num_heads, d_ff)  # Instantiate␣
↪the encoder block


#code here

# input
seq_length = 60
batch_size = 20
dummy_input = torch.rand(batch_size, seq_length, d_model)

# Forward pass
output = encoder_block(dummy_input)
print(output.shape)  # Expected shape: (batch_size, seq_length, d_model)
```

```
torch.Size([20, 60, 512])
```

1.8 Section 4: Positional Encoding

Positional Encoding is a technique used in the Transformer model to inject information about the position of each token in the sequence. Since the Transformer architecture does not inherently process sequential data in order (unlike RNNs or LSTMs), it lacks a way to inherently distinguish the order of tokens in the input. Positional encodings address this limitation by providing a unique signature for each position that can be added to the token embeddings, thus allowing the model to take the order of tokens into account.

1.9 Explanation:

Initialization: The PositionalEncoding module calculates the positional encodings for positions 0 through max_len for all dimensions of the model. The encoding uses a combination of sine and cosine functions of different frequencies.

Sinusoidal Functions: Odd and even indices of the positional encoding vector are filled with sine and cosine values, respectively. This pattern helps the model to distinguish between different positions due to the unique waveform shapes and frequencies.

Adding to Input Embeddings: The positional encoding is added to the input embeddings. This ensures that the model can not only learn from the data but also how the data is positioned relative to each other within the sequence.

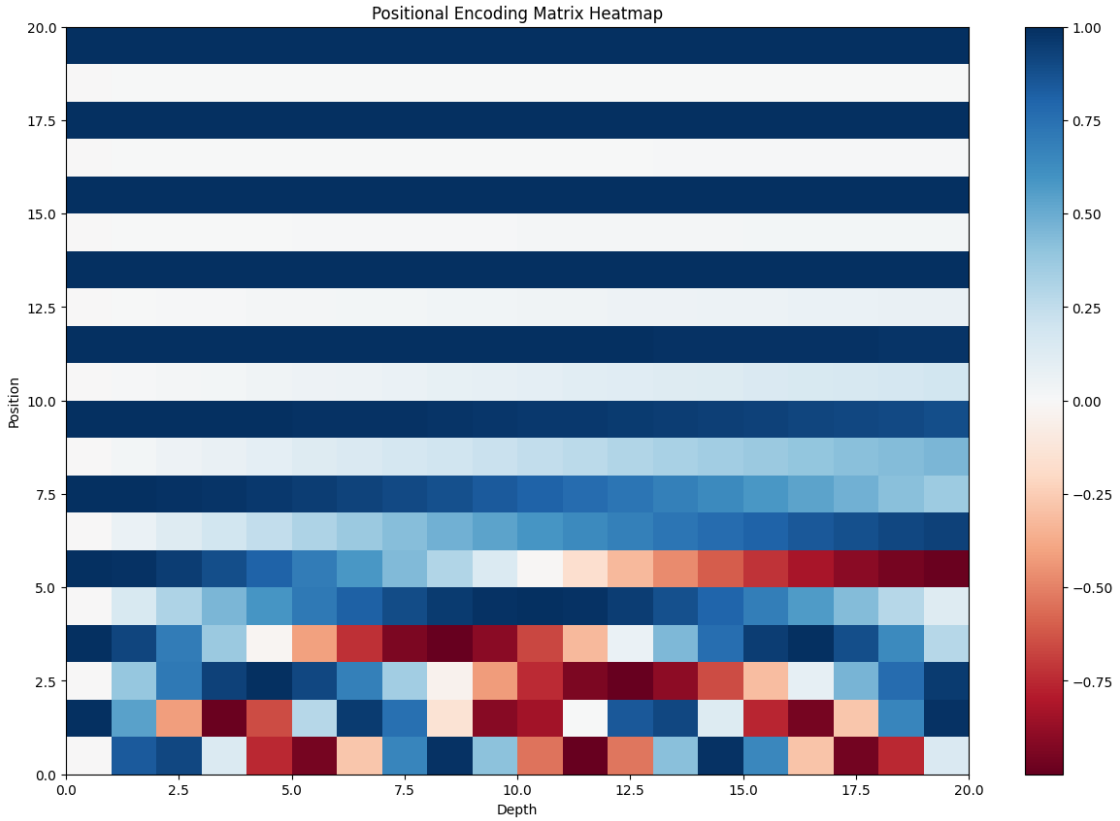
Positional Encoding is essential for the Transformer model to consider the order of words or elements in the sequence, allowing it to effectively process sequential data without relying on recurrent layers.

[6]:
```
import torch
import torch.nn as nn
import numpy as np
import matplotlib.pyplot as plt
import math

class PositionalEncoding(nn.Module):
    def __init__(self, d_model, max_len=5000):
        super(PositionalEncoding, self).__init__()
        """To do: 4.1
        """
        # Generate positions in a sequence up to a maximum length and Add a new␣
 ↪dimension to make it a column vector
        position = torch.arange(max_len).unsqueeze(1)
        #Calculate the div_term for positional encoding
        div_term = torch.exp(torch.arange(0, d_model, 2).float() * (-math.
 ↪log(10000.0) / d_model))



        pe = torch.zeros(max_len, d_model)

        #Todo 4.2
        pe[:, 0::2] = torch.sin(position * div_term)# sin function
        pe[:, 1::2] = torch.cos(position * div_term)# cos function

        pe = pe.unsqueeze(0).transpose(0, 1)  # Shape: [max_len, 1, d_model]
        self.register_buffer('pe', pe)

    def forward(self, x):
        x = x + self.pe[:x.size(0), :]
        return x
```

Test the Positional Encoding function

[7]:
```
# Generate the positional encoding for visualization with the definition above
d_model = 20  # Reduced size for visualization purposes
max_len = 100  # Sequence length for the visualization
pos_encoding = PositionalEncoding(d_model, max_len)
pos_encoding_matrix = pos_encoding.pe.squeeze().transpose(0, 1).numpy()

# Todo 4.3 Visualizing the positional encoding
import matplotlib.pyplot as plt  # Import matplotlib for visualization

# Visualize the Positional Encoding matrix after ensuring all necessary imports
plt.figure(figsize=(15, 10))
plt.pcolormesh(pos_encoding_matrix, cmap='RdBu')
```

```
plt.xlabel('Depth')
plt.xlim((0, d_model))
plt.ylabel('Position')
plt.colorbar()
plt.title('Positional Encoding Matrix Heatmap')
plt.show()
```



2 Marks:

Todo 1 : 30 Marks

Todo 2 : 35 Marks

Todo 3 : 10 Marks

Todo 4 : 25 Marks

[8]:
```
pip install --upgrade nbconvert jupyter
```

```
Requirement already satisfied: nbconvert in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (7.16.3)
Requirement already satisfied: jupyter in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (1.0.0)
```

```
Requirement already satisfied: beautifulsoup4 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
nbconvert) (4.12.3)
Requirement already satisfied: bleach!=5.0.0 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
nbconvert) (6.1.0)
Requirement already satisfied: defusedxml in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
nbconvert) (0.7.1)
Requirement already satisfied: jinja2>=3.0 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
nbconvert) (3.1.3)
Requirement already satisfied: jupyter-core>=4.7 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
nbconvert) (5.7.1)
Requirement already satisfied: jupyterlab-pygments in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
nbconvert) (0.3.0)
Requirement already satisfied: markupsafe>=2.0 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
nbconvert) (2.1.5)
Requirement already satisfied: mistune<4,>=2.0.3 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
nbconvert) (3.0.2)
Requirement already satisfied: nbclient>=0.5.0 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
nbconvert) (0.9.0)
Requirement already satisfied: nbformat>=5.7 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
nbconvert) (5.9.2)
Requirement already satisfied: packaging in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
nbconvert) (23.0)
Requirement already satisfied: pandocfilters>=1.4.1 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
nbconvert) (1.5.1)
Requirement already satisfied: pygments>=2.4.1 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
nbconvert) (2.17.2)
Requirement already satisfied: tinycss2 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
nbconvert) (1.2.1)
Requirement already satisfied: traitlets>=5.1 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
nbconvert) (5.14.1)
Requirement already satisfied: notebook in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jupyter) (7.1.2)
```

```
Requirement already satisfied: qtconsole in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jupyter) (5.5.1)
Requirement already satisfied: jupyter-console in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jupyter) (6.6.3)
Requirement already satisfied: ipykernel in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jupyter) (6.29.0)
Requirement already satisfied: ipywidgets in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jupyter) (8.1.2)
Requirement already satisfied: six>=1.9.0 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
bleach!=5.0.0->nbconvert) (1.16.0)
Requirement already satisfied: webencodings in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
bleach!=5.0.0->nbconvert) (0.5.1)
Requirement already satisfied: platformdirs>=2.5 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jupyter-core>=4.7->nbconvert) (4.2.0)
Requirement already satisfied: jupyter-client>=6.1.12 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
nbclient>=0.5.0->nbconvert) (8.6.0)
Requirement already satisfied: fastjsonschema in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
nbformat>=5.7->nbconvert) (2.19.1)
Requirement already satisfied: jsonschema>=2.6 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
nbformat>=5.7->nbconvert) (4.21.1)
Requirement already satisfied: soupsieve>1.2 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
beautifulsoup4->nbconvert) (2.5)
Requirement already satisfied: appnope in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
ipykernel->jupyter) (0.1.3)
Requirement already satisfied: comm>=0.1.1 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
ipykernel->jupyter) (0.2.1)
Requirement already satisfied: debugpy>=1.6.5 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
ipykernel->jupyter) (1.8.0)
Requirement already satisfied: ipython>=7.23.1 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
ipykernel->jupyter) (8.21.0)
Requirement already satisfied: matplotlib-inline>=0.1 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
ipykernel->jupyter) (0.1.6)
```

```
Requirement already satisfied: nest-asyncio in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
ipykernel->jupyter) (1.6.0)
Requirement already satisfied: psutil in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
ipykernel->jupyter) (5.9.8)
Requirement already satisfied: pyzmq>=24 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
ipykernel->jupyter) (25.1.2)
Requirement already satisfied: tornado>=6.1 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
ipykernel->jupyter) (6.4)
Requirement already satisfied: widgetsnbextension~=4.0.10 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
ipywidgets->jupyter) (4.0.10)
Requirement already satisfied: jupyterlab-widgets~=3.0.10 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
ipywidgets->jupyter) (3.0.10)
Requirement already satisfied: prompt-toolkit>=3.0.30 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jupyter-console->jupyter) (3.0.43)
Requirement already satisfied: jupyter-server<3,>=2.4.0 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
notebook->jupyter) (2.12.5)
Requirement already satisfied: jupyterlab-server<3,>=2.22.1 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
notebook->jupyter) (2.25.2)
Requirement already satisfied: jupyterlab<4.2,>=4.1.1 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
notebook->jupyter) (4.1.5)
Requirement already satisfied: notebook-shim<0.3,>=0.2 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
notebook->jupyter) (0.2.3)
Requirement already satisfied: qtpy>=2.4.0 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
qtconsole->jupyter) (2.4.1)
Requirement already satisfied: decorator in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
ipython>=7.23.1->ipykernel->jupyter) (5.1.1)
Requirement already satisfied: jedi>=0.16 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
ipython>=7.23.1->ipykernel->jupyter) (0.19.1)
Requirement already satisfied: stack-data in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
ipython>=7.23.1->ipykernel->jupyter) (0.6.3)
Requirement already satisfied: pexpect>4.3 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
ipython>=7.23.1->ipykernel->jupyter) (4.9.0)
```

```
Requirement already satisfied: attrs>=22.2.0 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jsonschema>=2.6->nbformat>=5.7->nbconvert) (23.2.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jsonschema>=2.6->nbformat>=5.7->nbconvert) (2023.12.1)
Requirement already satisfied: referencing>=0.28.4 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jsonschema>=2.6->nbformat>=5.7->nbconvert) (0.33.0)
Requirement already satisfied: rpds-py>=0.7.1 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jsonschema>=2.6->nbformat>=5.7->nbconvert) (0.17.1)
Requirement already satisfied: python-dateutil>=2.8.2 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jupyter-client>=6.1.12->nbclient>=0.5.0->nbconvert) (2.8.2)
Requirement already satisfied: anyio>=3.1.0 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jupyter-server<3,>=2.4.0->notebook->jupyter) (4.2.0)
Requirement already satisfied: argon2-cffi in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jupyter-server<3,>=2.4.0->notebook->jupyter) (23.1.0)
Requirement already satisfied: jupyter-events>=0.9.0 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jupyter-server<3,>=2.4.0->notebook->jupyter) (0.9.0)
Requirement already satisfied: jupyter-server-terminals in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jupyter-server<3,>=2.4.0->notebook->jupyter) (0.5.2)
Requirement already satisfied: overrides in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jupyter-server<3,>=2.4.0->notebook->jupyter) (7.7.0)
Requirement already satisfied: prometheus-client in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jupyter-server<3,>=2.4.0->notebook->jupyter) (0.19.0)
Requirement already satisfied: send2trash>=1.8.2 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jupyter-server<3,>=2.4.0->notebook->jupyter) (1.8.2)
Requirement already satisfied: terminado>=0.8.3 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jupyter-server<3,>=2.4.0->notebook->jupyter) (0.18.0)
Requirement already satisfied: websocket-client in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jupyter-server<3,>=2.4.0->notebook->jupyter) (1.7.0)
Requirement already satisfied: async-lru>=1.0.0 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jupyterlab<4.2,>=4.1.1->notebook->jupyter) (2.0.4)
Requirement already satisfied: httpx>=0.25.0 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jupyterlab<4.2,>=4.1.1->notebook->jupyter) (0.27.0)
```

```
Requirement already satisfied: jupyter-lsp>=2.0.0 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jupyterlab<4.2,>=4.1.1->notebook->jupyter) (2.2.2)
Requirement already satisfied: babel>=2.10 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jupyterlab-server<3,>=2.22.1->notebook->jupyter) (2.14.0)
Requirement already satisfied: json5>=0.9.0 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jupyterlab-server<3,>=2.22.1->notebook->jupyter) (0.9.14)
Requirement already satisfied: requests>=2.31 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jupyterlab-server<3,>=2.22.1->notebook->jupyter) (2.31.0)
Requirement already satisfied: wcwidth in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from prompt-
toolkit>=3.0.30->jupyter-console->jupyter) (0.2.13)
Requirement already satisfied: idna>=2.8 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
anyio>=3.1.0->jupyter-server<3,>=2.4.0->notebook->jupyter) (3.4)
Requirement already satisfied: sniffio>=1.1 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
anyio>=3.1.0->jupyter-server<3,>=2.4.0->notebook->jupyter) (1.3.0)
Requirement already satisfied: certifi in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
httpx>=0.25.0->jupyterlab<4.2,>=4.1.1->notebook->jupyter) (2023.5.7)
Requirement already satisfied: httpcore==1.* in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
httpx>=0.25.0->jupyterlab<4.2,>=4.1.1->notebook->jupyter) (1.0.5)
Requirement already satisfied: h11<0.15,>=0.13 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
httpcore==1.*->httpx>=0.25.0->jupyterlab<4.2,>=4.1.1->notebook->jupyter)
(0.14.0)
Requirement already satisfied: parso<0.9.0,>=0.8.3 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jedi>=0.16->ipython>=7.23.1->ipykernel->jupyter) (0.8.3)
Requirement already satisfied: python-json-logger>=2.0.4 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jupyter-events>=0.9.0->jupyter-server<3,>=2.4.0->notebook->jupyter) (2.0.7)
Requirement already satisfied: pyyaml>=5.3 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jupyter-events>=0.9.0->jupyter-server<3,>=2.4.0->notebook->jupyter) (6.0.1)
Requirement already satisfied: rfc3339-validator in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jupyter-events>=0.9.0->jupyter-server<3,>=2.4.0->notebook->jupyter) (0.1.4)
Requirement already satisfied: rfc3986-validator>=0.1.1 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jupyter-events>=0.9.0->jupyter-server<3,>=2.4.0->notebook->jupyter) (0.1.1)
Requirement already satisfied: ptyprocess>=0.5 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
```

```
pexpect>4.3->ipython>=7.23.1->ipykernel->jupyter) (0.7.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
requests>=2.31->jupyterlab-server<3,>=2.22.1->notebook->jupyter) (2.0.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
requests>=2.31->jupyterlab-server<3,>=2.22.1->notebook->jupyter) (1.26.16)
Requirement already satisfied: argon2-cffi-bindings in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
argon2-cffi->jupyter-server<3,>=2.4.0->notebook->jupyter) (21.2.0)
Requirement already satisfied: executing>=1.2.0 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from stack-
data->ipython>=7.23.1->ipykernel->jupyter) (2.0.1)
Requirement already satisfied: asttokens>=2.1.0 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from stack-
data->ipython>=7.23.1->ipykernel->jupyter) (2.4.1)
Requirement already satisfied: pure-eval in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from stack-
data->ipython>=7.23.1->ipykernel->jupyter) (0.2.2)
Requirement already satisfied: fqdn in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jsonschema>=2.6->nbformat>=5.7->nbconvert) (1.5.1)
Requirement already satisfied: isoduration in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jsonschema>=2.6->nbformat>=5.7->nbconvert) (20.11.0)
Requirement already satisfied: jsonpointer>1.13 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jsonschema>=2.6->nbformat>=5.7->nbconvert) (2.1)
Requirement already satisfied: uri-template in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jsonschema>=2.6->nbformat>=5.7->nbconvert) (1.3.0)
Requirement already satisfied: webcolors>=1.11 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
jsonschema>=2.6->nbformat>=5.7->nbconvert) (1.13)
Requirement already satisfied: cffi>=1.0.1 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
argon2-cffi-bindings->argon2-cffi->jupyter-server<3,>=2.4.0->notebook->jupyter)
(1.15.1)
Requirement already satisfied: pycparser in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
cffi>=1.0.1->argon2-cffi-bindings->argon2-cffi->jupyter-
server<3,>=2.4.0->notebook->jupyter) (2.21)
Requirement already satisfied: arrow>=0.15.0 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
isoduration->jsonschema>=2.6->nbformat>=5.7->nbconvert) (1.3.0)
Requirement already satisfied: types-python-dateutil>=2.8.10 in
/opt/homebrew/Caskroom/miniconda/base/lib/python3.11/site-packages (from
arrow>=0.15.0->isoduration->jsonschema>=2.6->nbformat>=5.7->nbconvert)
```

```
(2.8.19.20240106)
Note: you may need to restart the kernel to use updated packages.
```

[ ]: