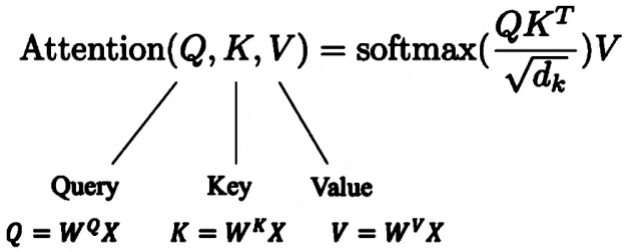
ECS795P Deep Learning and Computer Vision, 2024

Submitted by: Ruthwik Ganesh (230702930) – ec23759@qmul.ac.uk

Coursework 1: Building a transformer from Scratch.

1. The calculation formula for self-attention is as follows. Assuming the input X has dimensions [768, 6], and W^Q, W^K and W^V all have dimensions [768, 768], calculate the dimensions of K, Q, V, and the output of Attention (Q, K, V). (10% of CW1)



Q = $W^Q X$ = [768, 768] x [768, 6] => ***[768, 6]***
K =$W^K X$ = [768, 768] x [768, 6] => ***[768, 6]***
V = $W^V X$ = [768, 768] x [768, 6] => ***[768, 6]***

QK^T = [768, 6] x [6, 768] => [768, 768]
$(QK^T)V$ = [768, 768] x [768, 6] = [768, 6]
Thus, Attention (Q, K, V) = ***[768, 6]***

2. 1) How is the self-attention mechanism implemented?

   i. The initial step in the computation of self-attention involves the generation of three vectors for each input vector of the encoder, specifically the embedding of each word. For each word, three distinct vectors are formulated: a Query vector, a Key vector, and a Value vector. The creation of these vectors is accomplished through the matrix multiplication of the embedding with three matrices that were trained during the model's training process.

   ii. The subsequent stage in the computation of self-attention involves the determination of a score to each word within the input sentence concerning this specific word. This score serves as a metric to gauge the level of attention to allocate to various segments of the input sentence during the encoding process of a word at a particular position. It is derived by executing the dot product between the query vector and the key vector associated with the word being scored.

   iii. Following the previous step involves the division of the computed scores by the square root of the dimension of the key vectors. This choice aims to enhance the stability of gradients. Subsequently, the outcome undergoes a softmax operation. The softmax operation serves the purpose of normalizing the scores, ensuring they all assume positive values and collectively sum to 1. The softmax values essentially dictate the extent to which each word will

be emphasized at the given position. While the word in the current position invariably attains the highest softmax value, it is noteworthy that there are instances where it is beneficial to attend to another word that bears relevance to the word under consideration.

iv. In the next step, each value vector is multiplied with the softmax value. The underlying rationale is to preserve the values associated with the targeted word(s) of interest, while mitigating the impact of irrelevant words. This mitigation is achieved by multiplying irrelevant values with diminutive coefficients, such as 0.001.

v. Final step involves the summation of the weighted value vectors. This summation culminates in the generation of the output for the self-attention layer at the current position, specifically in the context of the first word being considered.

2) What are the primary applications of self-attention? (mention 3 points)

Self-attention, particularly in the context of deep learning and natural language processing, has various applications. Here are three primary applications:

1. Natural Language Processing (NLP): Self-attention mechanisms are widely used in NLP tasks, such as machine translation, sentiment analysis, and text summarization. Models like Transformer, which introduced the self-attention mechanism, have shown remarkable performance in capturing long-range dependencies in sequential data like sentences. The ability to weigh different parts of the input sequence differently allows the model to focus on relevant information, improving the overall understanding and performance of NLP tasks.
2. Image Processing and Computer Vision: Self-attention has been adapted to computer vision tasks to improve the performance of image recognition and object detection models. By incorporating self-attention mechanisms, models can capture spatial dependencies between different regions of an image effectively. This helps in identifying and focusing on relevant parts of the input image, allowing the model to make more accurate predictions in tasks such as image classification, segmentation, and object detection.
3. Generative Models: Self-attention has found applications in generative models, such as Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs). In these models, self-attention mechanisms help capture dependencies between different elements of the input data, enabling the generation of more coherent and realistic samples. This is particularly beneficial in tasks like image synthesis, where the model needs to understand global and local structures within the data for accurate generation.

3) What advantages does transformer have compared to CNN? (10% of CW1)

In the context of computer vision tasks, Convolutional Neural Networks (CNNs) have been the traditional and highly successful choice. However, transformers have started to show promise in computer vision as well. Here are some advantages of transformers compared to CNNs in the context of computer vision tasks:

1. Global Context Understanding: Transformers are designed to capture global context information effectively. In computer vision tasks, understanding the relationships and dependencies between distant pixels or regions in an image can be crucial for making accurate predictions. The self-attention mechanism in transformers allows them to consider information from the entire image, enabling better modeling of long-range dependencies.
2. Adaptability to Various Input Sizes: CNNs often require fixed input sizes, and handling variable-sized inputs can be challenging. Transformers, however, can handle variable-sized inputs more naturally due to their attention mechanism. This adaptability is beneficial in scenarios where the size of the input images may vary, such as in object detection or image segmentation tasks.
3. Attention to Informative Regions: Transformers can dynamically attend to informative regions in the input, ignoring irrelevant or noisy details. This attention mechanism allows the model to focus on the most relevant parts of the image, potentially improving the robustness of the model to variations and distractions in the input.
4. Parallelization: While CNNs rely on sequential convolutions, transformers can parallelize computations across different positions in the image. This parallelization can lead to more efficient training on hardware accelerators like GPUs, potentially reducing training times for large-scale computer vision models.

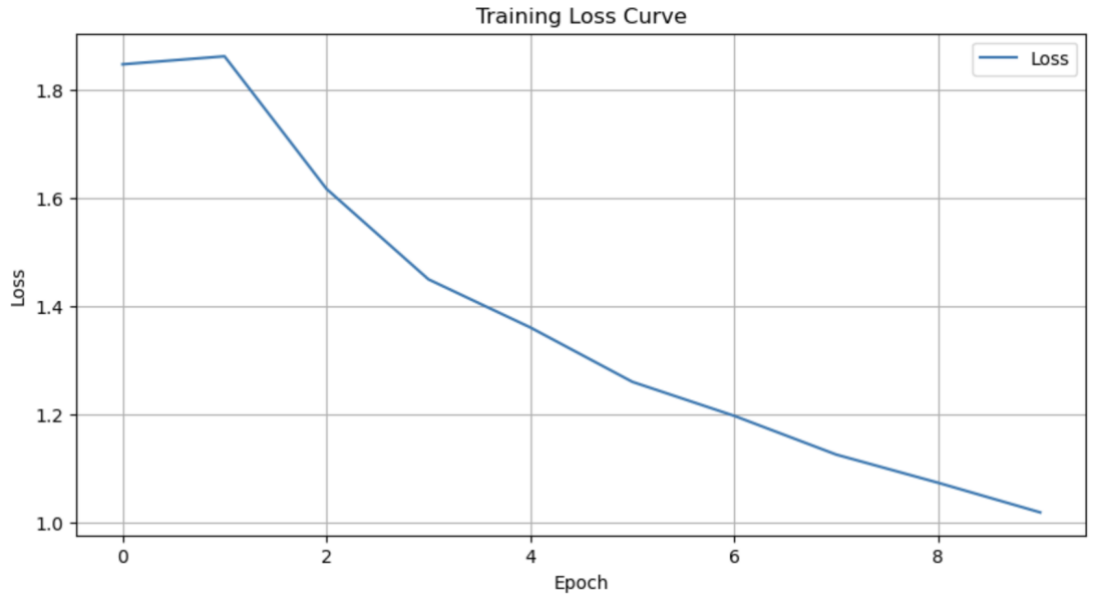
3. Answer the following questions based on the code:

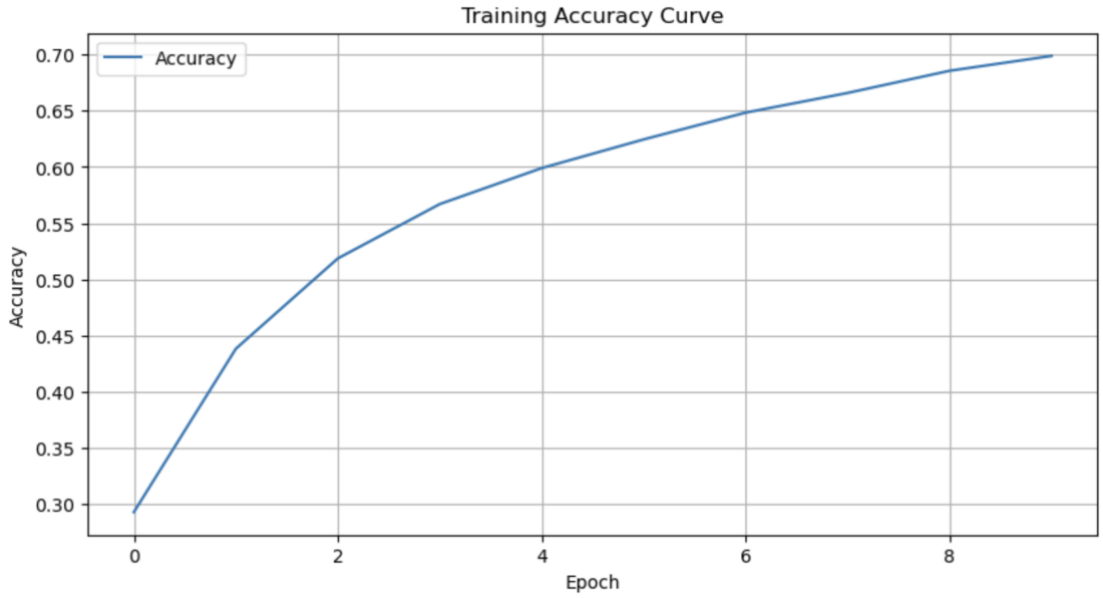
1) In the section “From attention to transformers - multi-headattention”, what are the shapes of the following variables: *tokens, qis_mh, kis_mh, vis_mh, attmat_mh, zis_mh, zis*?

1. **token**: batch size x token count x embedding length ***[1, 5, 768]***.
2. **Qis_mh, Kis_mh, Vis_mh**: batch size x token count x head count x head dimension ***[1, 5, 12, 64]***.
3. **attmat_mh**: batch size x head count x token count x token count ***[1, 12, 5, 5]***.
4. **zis_mh**: batch size x token count x head count x head dimension ***[1, 5, 12, 64]***.
5. **zis**: batch size x token count x embedding length ***[1, 5, 768]***.

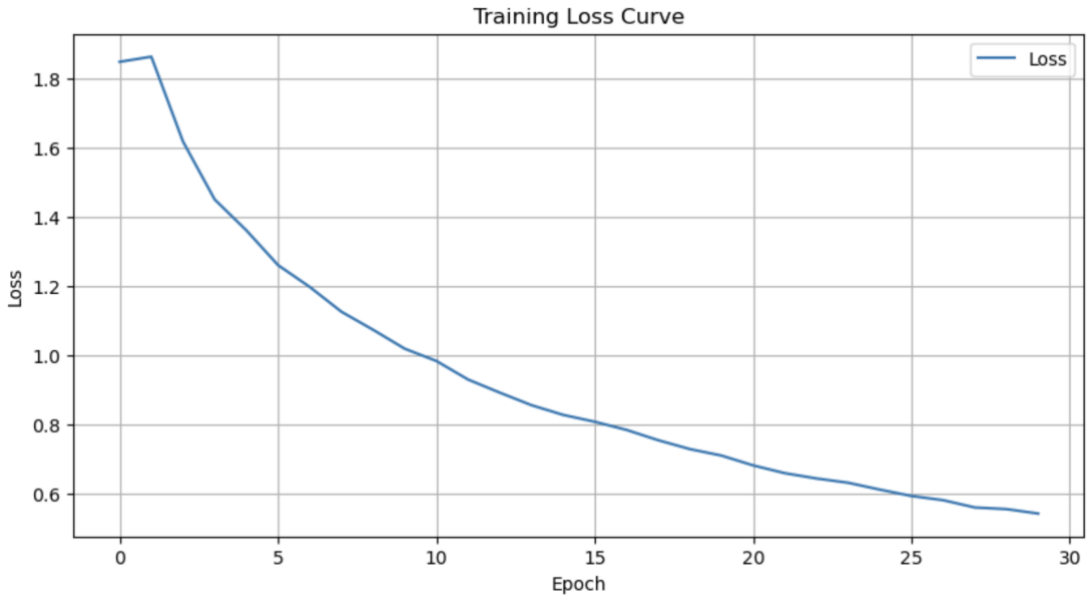
2) In the 'Classification' section, plot the curves of classification accuracy and loss as they vary with epochs for three different settings: 10 epochs, 30 epochs, and 50 epochs. (10% of CW1)

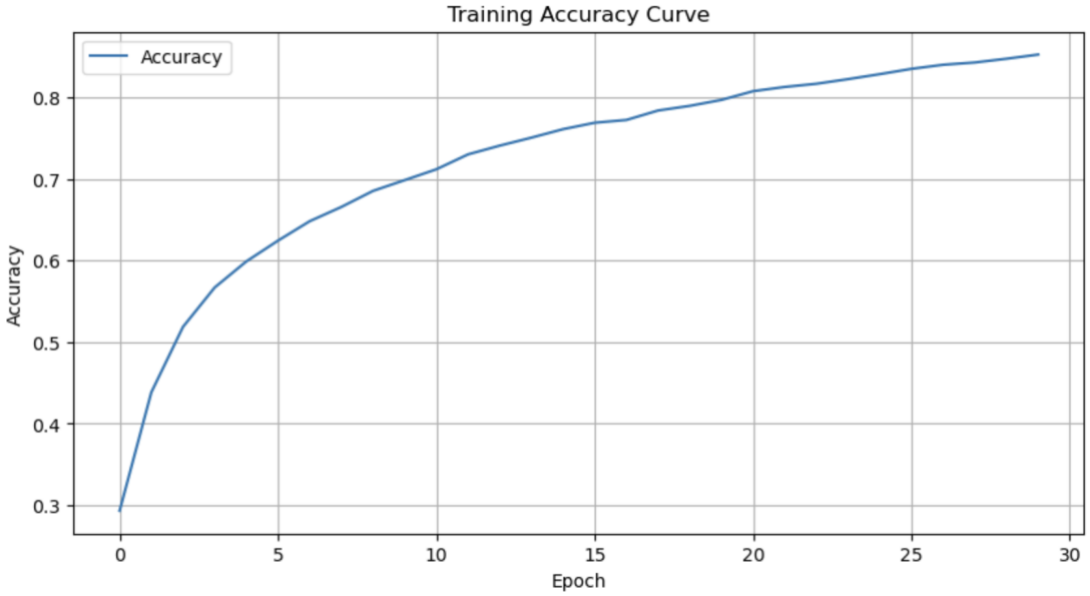
10 Epochs



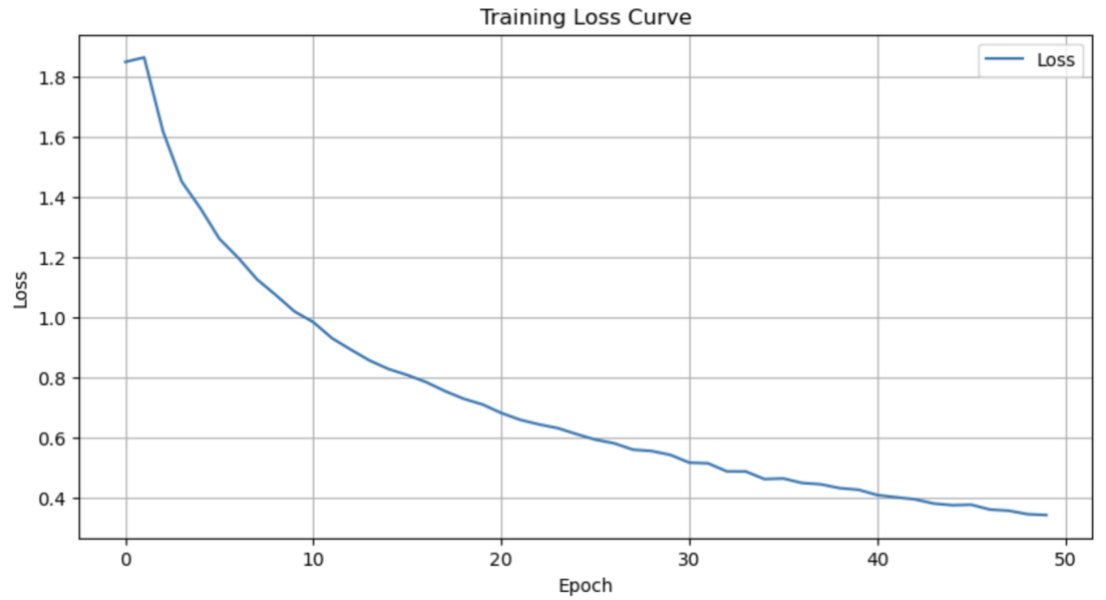


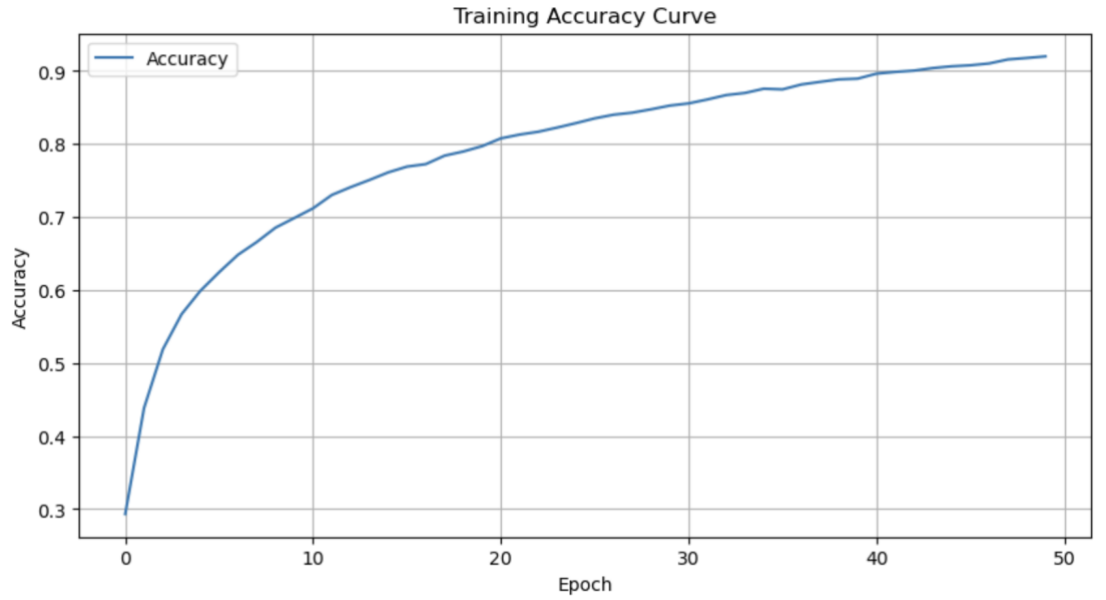
30 Epochs


Training Loss Curve
Loss
1.8
1.6
1.4
1.2
1.0
0.8
0.6
Loss
0
5
10
15
20
25
30
Epoch


Training Accuracy Curve
Accuracy
0.8
0.7
0.6
0.5
0.4
0.3
Accuracy
0
5
10
15
20
25
30
Epoch

50 Epochs


Training Loss Curve
Loss
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
Loss
0
10
20
30
40
50
Epoch


Training Accuracy Curve
Accuracy
0.9
0.8
0.7
0.6
0.5
0.4
0.3
Accuracy
0
10
20
30
40
50
Epoch