EECS 182 Deep Neural Networks
Fall 2026 Gireeja Ranade

Homework 11

This homework is due on Wednesday, November 18, 2026, at 10:59PM.

1. Kernelized Linear Attention (Part II)

This is the second part of the "Kernelized Linear Attention" problem from HW10 (Problem 5). Please refer to Part I for notation and context.

In Part I of this problem, we considered ways to efficiently express the attention operation when sequences are long (e.g. a long document). Attention uses the following equation:

$$V'_{i\cdot} = \frac{\sum_{j=1}^{N} \text{sim}\left(Q_{i\cdot}, K_{j\cdot}\right) V_{j\cdot}}{\sum_{j=1}^{N} \text{sim}\left(Q_{i\cdot}, K_{j\cdot}\right)}. \tag{1}$$

We saw that when the similarity function is a kernel function (i.e. if we can write $\text{sim}\left(Q_{i\cdot}, K_{j\cdot}\right) = \Phi(Q_{i\cdot})\Phi(K_{j\cdot})^T$ for some function Φ), then we can use the associative property of matrix multiplication to simplify the formula to

$$V'_{i\cdot} = \frac{\phi\left(Q_{i\cdot}\right) \sum_{j=1}^{N} \phi\left(K_{j\cdot}\right)^T V_{j\cdot}}{\phi\left(Q_{i\cdot}\right) \sum_{j=1}^{N} \phi\left(K_{j\cdot}\right)^T}. \tag{2}$$

If we use a polynomial kernel with degree 2, this gives a computational cost of $\mathcal{O}\left(ND^2M\right)$, which for very large N is favorable to the softmax attention computational cost of $\mathcal{O}\left(N^2 \max\left(D, M\right)\right)$. ($N$ is the sequence length, D is the feature dimension of the queries and keys, and M is the feature dimension of the values.) Now, we will see whether we can use kernel attention to directly approximate the softmax attention:

$$V'_{i\cdot} = \frac{\sum_{j=1}^{N} \exp(\frac{Q_{i\cdot}K_{j\cdot}^T}{\sqrt{D}}) V_{j\cdot}}{\sum_{j=1}^{N} \exp(\frac{Q_{i\cdot}K_{j\cdot}^T}{\sqrt{D}})}. \tag{3}$$

(a) Approximating softmax attention with linearized kernel attention

i. As a first step, we can use Gaussian Kernel $\mathcal{K}_{\text{Gauss}}(q,k) = \exp(\frac{-||q-k||_2^2}{2\sigma^2})$ to rewrite the softmax similarity function, where $\text{sim}_{\text{softmax}}(q,k) = \exp(\frac{q^Tk}{\sqrt{D}})$. Let $\sigma^2 = \sqrt{D}$. **Rewrite the softmax similarity function using the Gaussian kernel.** (*Hint: You can write* $\exp(\frac{q^Tk}{\sqrt{D}})$ *as the product of the Gaussian kernel and two other terms.*)

ii. However, writing softmax attention using a Gaussian kernel does not directly enjoy the benefits of the reduced complexity using the feature map. This is because the feature map of the Gaussian kernel usually comes from the Taylor expansion of $\exp(\cdot)$, which has infinitely many terms, so

 All Rights Reserved. This may not be publicly shared without explicit permission.

computing it is still expensive [1]. However, we can approximate the Gaussian kernel using a random feature map and then reduce the computation cost, using a trick from Random feature attention (2021)[2]. (Rahimi and Recht, 2007)[3] proposed random Fourier features to approximate a desired shift-invariant kernel. The method nonlinearly transforms a pair of vectors q and k using a **random feature map** $\phi_{\text{random}}()$; the inner product between $\phi(q)$ and $\phi(k)$ approximates the kernel evaluation on q and k. More precisely:

$$\phi_{\text{random}}(q) = \sqrt{\frac{1}{D_{\text{random}}}}\left[\sin\left(\mathbf{w}_1^T q\right), \ldots, \sin\left(\mathbf{w}_{D_{\text{random}}}^T q\right), \cos\left(\mathbf{w}_1^T q\right), \ldots, \cos\left(\mathbf{w}_{D_{\text{random}}}^T q\right)\right]^\top. \tag{4}$$

Here $\mathbf{w}_1, \ldots, \mathbf{w}_{D_{\text{random}}} \in \mathbb{R}^D$ are random vectors, sampled independently from $\mathcal{N}(\mathbf{0}, \sigma^{-2}\mathbf{I}_D)$. Then

$$\mathbb{E}_{\mathbf{w}_1,\ldots,\mathbf{w}_{D_{\text{random}}}}\left[\phi_{\text{random}}(q)^T \phi_{\text{random}}(k)\right] = \exp\left(-\|q-k\|^2/2\sigma^2\right). \tag{5}$$

Use ϕ_{random} to approximate the above softmax similarity function with Gaussian Kernel and derive the computation cost for computing all the V' here.

(b) (Optional) In autoregressive models such as GPT, attention is masked so that the i-th position can only be influenced by a position j if $j \leq i$. A position cannot be influenced by later positions. This type of attention masking is called *causal masking*.

Derive how this causal masking changes equation 1 and 2. **Write equation 1 and 2 in terms of $S_{i\cdot}$ and $Z_{i\cdot}$**, which are defined as:

$$S_{i\cdot} = \sum_{j=1}^{i} \phi\left(K_{j\cdot}\right)^T V_{j\cdot}, \quad Z_{i\cdot} = \sum_{j=1}^{i} \phi\left(K_{j\cdot}\right)^T, \tag{6}$$

to simplify the causal masking kernel attention and derive the computational complexity of this new causal masking formulation scheme.

2. Coding Question: Hand-Design Transformers

Please follow the instructions in this notebook. You will implement a simple transformer model (with a single attention head) from scratch and then create hand-designed attention heads of the transformer model capable of solving a basic problem.

Once you have finished the notebook, answer the following questions in your submission of the written assignment:

(a) Design a transformer that selects by contents. Compare the variables of your hand-designed Transformer with those of the learned Transformer. **Identify the similarities and differences between the two sets of variables and provide a brief explanation for each difference.**

[1] https://www.csie.ntu.edu.tw/~cjlin/talks/kuleuven_svm.pdf

[2] Peng, Hao, et al. "Random feature attention." arXiv preprint arXiv:2103.02143 (2021).

[3] Rahimi, Ali, and Benjamin Recht. "Random features for large-scale kernel machines." Advances in neural information processing systems 20 (2007).

 All Rights Reserved. This may not be publicly shared without explicit permission.

(b) Design a transformer that selects by positions. Compare the variables of your hand-designed Transformer with those of the learned Transformer. **Identify the similarities and differences between the two sets of variables and provide a brief explanation for each difference.**

(c) (Optional) Design a transformer that selects by both content and position (the optional last section of the notebook, which outputs a positive number for each unique token and a negative number for each repeated token). **Comment on the similarities and differences between the weights and intermediate outputs of the learned and hand-coded models.**

3. Scaling Laws of Batch Size

A common question in neural network training is, how should I select my hyperparameters? While a proper hyperparameter sweep will always provide the best answer, sweeps can become impractical especially at larger network sizes.

In this homework question, we will derive a simple scaling law for the optimal learning rate under varying batch sizes. Complete this notebook, and answer the following questions in your submission:

(a) **For linear regression using SGD, attach curves of learning rate vs MSE loss for all the considered batch sizes, and curve showing relationship between batch size and optimal learning rate. What function does this resemble?**

(b) **Attach the same curves but for an MLP instead of linear regression. How do the scaling laws differ?**

(c) **Finally, show the same curves when using the Adam optimizer instead of SGD. Does the scaling for learning rate change when using Adam?**

4. Read a Research Paper: FaceNet

In class, you have learnt how self-supervised learning can be used to learn useful representations from large datasets without labels (e.g., learning features from ImageNet). While these features may inherently pick out some notion of "similarity" between different images in the dataset, they are not incentivized to *cluster* different data points based on any interesting similarity measure.

The paper "FaceNet: A Unified Embedding for Face Recognition and Clustering" explores how we can view the task of face recognition through the lens of self-supervised (or to be more accurate, *slightly* supervised) learning.

You may find this blog post easier to follow than the paper. Its two figures, shown below, are particularly helpful.

Read the paper and answer the questions below.

(a) **What are the two neural network architectures considered by the authors?**

(b) **Briefly describe the *triplet loss* and how it differs from a typical supervised learning objective.**

(c) **What is the challenge with generating all possible triplets? Briefly describe how the authors address this challenge.**

(d) **How many parameters and floating point operations (FLOPs) do the authors use for their neural network? How does this compare to a ResNet-50?** Read Table 1 of the original paper[4] on ResNet to find out the FLOPs of ResNet-50. Its parameter count can be found by searching online.

[4] He, Kaiming, et al. "Deep residual learning for image recognition." (2016).

 All Rights Reserved. This may not be publicly shared without explicit permission.

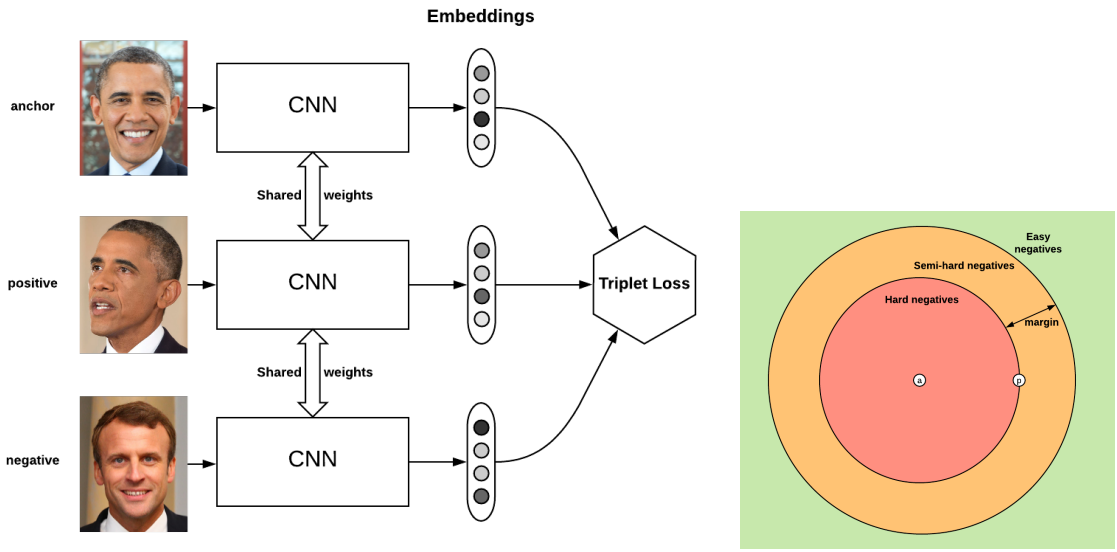


Figure 1: Figures from blog post.

(e) **What do the authors mean by *semi-hard* negatives?**

(f) **What are harmonic embeddings?**

(g) **How does the performance vary with embedding dimensionality?**

(h) **How does the performance vary with increasing amounts of training data?**

(i) Briefly share your favorite *emergent* property/result of the learned behavior with a triplet loss from the paper.

(j) **Which approach taken by the authors interested you the most? Why?** ($\approx$ 100 words)

5. A Brief Introduction to Transformer Interpretability

Modern neural networks, particularly large language models like Transformers, achieve remarkable performance but are often treated as "black boxes". Mechanistic interpretability is a research field dedicated to reverse-engineering the specific circuits these models learn. Instead of just observing their input-output behavior, we aim to understand the internal mechanisms and computations that lead to their capabilities. This problem explores a mathematical framework proposed by Elhage et al. that treats Transformers as a collection of interacting "circuits" built from attention and MLP layers.

The Residual Stream: A Central Communication Channel

A core concept in this framework is the **residual stream**. At any given layer l, the residual stream, denoted $\mathbf{X}^l$, is a sequence of vectors representing the state of the computation for each token in the input. Crucially, the Transformer, like all modern post-ResNet architectures, is based on residual connections, meaning the output of each layer is *added* to the stream from the previous layer. For a layer l containing a component (e.g., an attention head) with function $f^{(l)}$, the update rule is:

$$\mathbf{X}^{l+1} = \mathbf{X}^l + f^{(l)}(\mathbf{X}^l)$$

 All Rights Reserved. This may not be publicly shared without explicit permission.

The final output of an L-layer model is therefore the sum of the initial embedding and the outputs of all layers:

$$\mathbf{X}^{\text{final}} = \mathbf{X}^0 + \sum_{l=0}^{L-1} f^{(l)}(\mathbf{X}^l)$$

This additive structure is fundamental. It allows us to view the residual stream as a central communication bus or channel where different components read information from and additively write their results back to. This reframing is key to analyzing the function of individual components in relative isolation.

Throughout this problem, we refer to the following variables and their shapes.

Variable	Shape	Description
$\mathbf{T}$	$\mathbb{R}^{n_{\text{vocab}} \times n_{\text{context}}}$	Matrix of one-hot encoded input tokens. Each column is a token.
$\mathbf{X}^l$	$\mathbb{R}^{d_{\text{model}} \times n_{\text{context}}}$	"Residual stream" or "embedding" vectors of model at layer l. Each column is a vector for a token.
$\mathbf{W}_E$	$\mathbb{R}^{d_{\text{model}} \times n_{\text{vocab}}}$	Token embedding matrix. Maps one-hot vectors to the residual stream.
$\mathbf{W}_U$	$\mathbb{R}^{n_{\text{vocab}} \times d_{\text{model}}}$	Token unembedding matrix. Maps final residual stream vectors to logits.
$\mathbf{W}_Q^h, \mathbf{W}_K^h, \mathbf{W}_V^h$	$\mathbb{R}^{d_{\text{head}} \times d_{\text{model}}}$	Query, Key, and Value weight matrices for attention head h.
$\mathbf{W}_O^h$	$\mathbb{R}^{d_{\text{model}} \times d_{\text{head}}}$	Output weight matrix for attention head h. This weight matrix left multiplies the output from attention head h and projects it into the dimension of the residual stream.
$\mathbf{A}^h$	$\mathbb{R}^{n_{\text{context}} \times n_{\text{context}}}$	The attention pattern matrix (post-softmax) for attention head h.
$\mathbf{W}_{OV}^h$	$\mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$	The Output-Value matrix for attention head h, defined as $\mathbf{W}_{OV}^h = \mathbf{W}_O^h \mathbf{W}_V^h$.
$\mathbf{W}_{QK}^h$	$\mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$	The Query-Key matrix for attention head h, defined as $\mathbf{W}_{QK}^h = \mathbf{W}_Q^{hT} \mathbf{W}_K^h$.

Note that the superscript h is omitted when context is clear about which attention head we are referring to.

(a) **The Simplest Transformer (A Zero-Layer Model)**

i. Given a matrix of one-hot input tokens $\mathbf{T} \in \mathbb{R}^{n_{\text{vocab}} \times n_{\text{context}}}$, write the mathematical expression for the final logits, $\mathbf{L} \in \mathbb{R}^{n_{\text{vocab}} \times n_{\text{context}}}$. Your expression should be in terms of $\mathbf{T}$, the token embedding matrix $\mathbf{W}_E$, and the unembedding matrix $\mathbf{W}_U$.

ii. In simple terms, what algorithm does this zero-layer model implement? What information does the prediction for the token at position t depend on? This reveals the baseline functionality of the Transformer architecture before any contextual processing is introduced.

 All Rights Reserved. This may not be publicly shared without explicit permission.

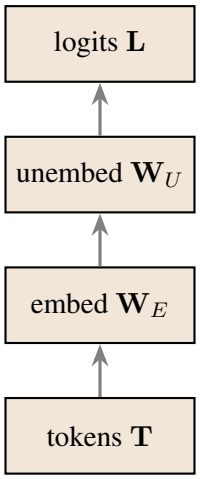


Figure 2: The Simplest Transformer: A Zero-Layer Model

(b) **Multi-Head Attention: Concatenation vs. Addition**

In the original Vaswani et al. paper on transformers, a multi-head attention layer is described differently than our "independent circuits" view. There, the outputs of all heads are concatenated and then multiplied by a single large output matrix. This question asks you to prove this is equivalent to an "additive and independent" view.

Consider an attention layer with H heads. Let the value computation output for head h be $\mathbf{r}^h \in \mathbb{R}^{d_{\text{head}} \times n_{\text{context}}}$. In the "concatenation" view, these are stacked vertically to form a matrix $\mathbf{R}^H \in \mathbb{R}^{d_{\text{head}} \cdot H \times n_{\text{context}}}$ and multiplied by a single large output matrix $\mathbf{W}_O^H \in \mathbb{R}^{d_{\text{model}} \times d_{\text{head}} \cdot H}$, and the final output is $\mathbf{H} = \mathbf{W}_O^H \mathbf{R}^H$. Typically, the dimension $d_{\text{head}} \cdot H$ is equal to d_{model} where $H = d_{\text{model}}/d_{\text{head}}$. In the "additive and independent" view, each head h has its own output matrix, $\mathbf{W}_O^h \in \mathbb{R}^{d_{\text{model}} \times d_{\text{head}}}$, and the total output is $\mathbf{H} = \sum_{h=1}^{H} \mathbf{W}_O^h \mathbf{r}^h$.

(i) Show that these formulations are equivalent. Specifically, demonstrate how $\mathbf{W}_O^H$ can be constructed from the individual $\mathbf{W}_O^h$ matrices to make the two expressions for $\mathbf{H}$ identical.

(ii) What is an advantage and disadvantage of each view?

For the following parts, we consider a Transformer with a single attention layer with H heads and no normalizations as shown in Figure 3.

(c) **The QK Circuit: Determining Attention Patterns**

An attention head can conceptually be split into two independent operations: one that decides *where to look* (the QK circuit) and one that decides *what information to move* (the OV circuit). Let's first analyze the "where to look" mechanism.

The pre-softmax attention score from a "query" token at position i to a "key" token at position j is computed as:

$$\mathbf{S}_{ij} = \mathbf{Q}_i^T \mathbf{K}_j$$

where $\mathbf{Q}_i = \mathbf{W}_Q \mathbf{X}_i$ and $\mathbf{K}_j = \mathbf{W}_K \mathbf{X}_j$ are the query and key vectors. (We drop the $1/\sqrt{d_{\text{head}}}$ scaling, which does not change the analysis.)

(i) Starting from the definitions above, derive the expression for $\mathbf{S}_{ij}$ as a bilinear form involving the residual stream vectors $\mathbf{X}_i$ and $\mathbf{X}_j$, and a single "virtual" weight matrix $\mathbf{W}_{QK}$. Explicitly define $\mathbf{W}_{QK}$ in terms of $\mathbf{W}_Q$ and $\mathbf{W}_K$.

(ii) The matrix $\mathbf{W}_{QK}$ can be interpreted as defining the "question" the attention head asks to determine which tokens to attend to. Consider a toy scenario where $d_{\text{model}} = 3$.

 All Rights Reserved. This may not be publicly shared without explicit permission.

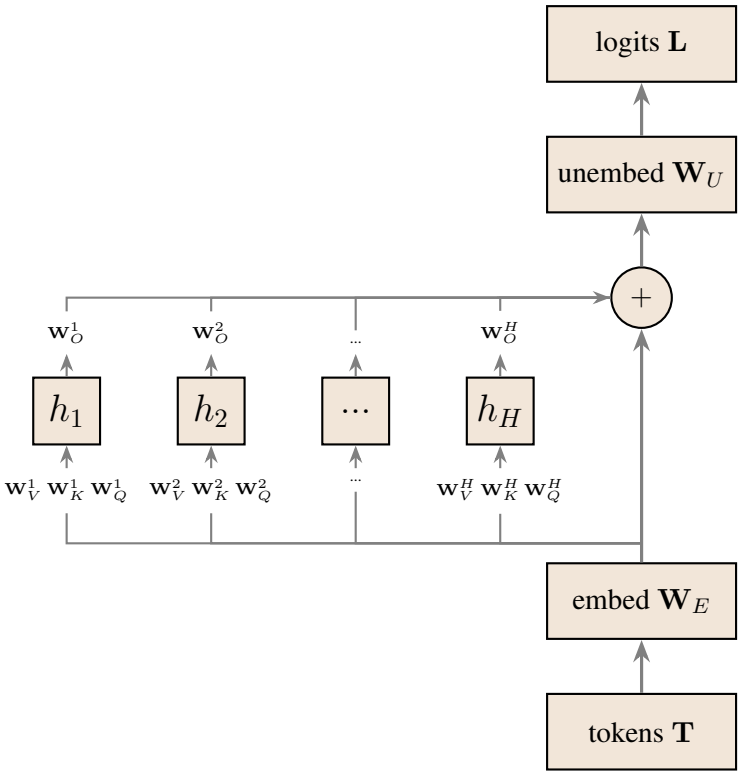


Figure 3: A Transformer With A Single Attention Layer with No Normalizations

(1) If $\mathbf{W}_{QK} = \mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, what kind of relationship between $\mathbf{X}_i$ and $\mathbf{X}_j$ would lead to a high attention score $\mathbf{S}_{ij} = \mathbf{X}_i^T \mathbf{W}_{QK} \mathbf{X}_j$? Describe in words what this head "looks for."

(2) If $\mathbf{W}_{QK} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$, what kind of relationship would lead to a high attention score? What happens to information in the second and third dimensions?

(3) If $\mathbf{W}_{QK} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$, what pattern does this head look for? How does this differ from the identity case?

(4) If $\mathbf{W}_{QK} = -\mathbf{I} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$, what pattern does this head look for? What is the effect of this on the attention pattern after softmax?

(d) **The OV Circuit: Reading and Writing Information**

Now let's analyze the information-moving part of the attention head. Once the attention pattern $\mathbf{A}^h$ is determined (by the QK circuit), the head reads information from attended tokens and writes it to the destination. This is governed by the OV circuit. Note that the output of head h is:

$$h(\mathbf{X}^0) = \mathbf{W}_O^h \mathbf{W}_V^h \mathbf{X}^0 \mathbf{A}^{hT} = \mathbf{W}_{OV}^h \mathbf{X}^0 \mathbf{A}^{hT}$$

 All Rights Reserved. This may not be publicly shared without explicit permission.

where $\mathbf{A}^h$ is the attention pattern (post-softmax) determined by the QK circuit.

(i) **[Understanding the Attention-Weighted Average]**
Before proceeding, let's clarify what $\mathbf{X}^0\mathbf{A}^T$ computes. Recall that:

- $\mathbf{X}^0 \in \mathbb{R}^{d_{\text{model}} \times n_{\text{context}}}$ where the j-th column $\mathbf{X}^0_j$ is the residual stream vector for the token at position j.
- $\mathbf{A} \in \mathbb{R}^{n_{\text{context}} \times n_{\text{context}}}$ is the attention pattern where $\mathbf{A}_{ij}$ is the attention weight from destination position i to source position j.
- Therefore, $\mathbf{A}^T \in \mathbb{R}^{n_{\text{context}} \times n_{\text{context}}}$ where $(\mathbf{A}^T)_{ji} = \mathbf{A}_{ij}$.

Show that the i-th column of $\mathbf{X}^0\mathbf{A}^T$ is an attention-weighted average of the source token vectors. Specifically, show:

$$(\mathbf{X}^0\mathbf{A}^T)_i = \sum_{j=1}^{n_{\text{context}}} \mathbf{A}_{ij}\mathbf{X}^0_j$$

(ii) Write the full expression for the final residual stream, $\mathbf{X}^{\text{final}}$, in terms of the initial stream $\mathbf{X}^0$ and the heads' weight matrices.

(iii) The update vector added to the residual stream at destination position t by head i is the t-th column of the head's output, denoted $h_i(\mathbf{X}^0)_t$. Prove that this update vector must lie within the column space of $\mathbf{W}^i_{OV}$. That is, show that $h_i(\mathbf{X}^0)_t \in \text{Col}(\mathbf{W}^i_{OV})$. This demonstrates that the head can only write to a low-dimensional subspace of the d_{model}-dimensional residual stream.

(e) **Formalizing Read and Write Subspaces via SVD**

Important: Separating the QK and OV Circuits
In this analysis, we focus exclusively on the **OV circuit** $\mathbf{W}_{OV}$, which characterizes *what* information is moved and *how* it is transformed.
Recall from part (d) that the full output of an attention head is:

$$h(\mathbf{X}^0) = \mathbf{W}_{OV}\mathbf{X}^0\mathbf{A}^T$$

This can be decomposed into two independent operations:

- **QK Circuit** (analyzed in part c): Computes the attention pattern $\mathbf{A}$, which determines *which* source tokens to attend to at each destination position.
- **OV Circuit** (analyzed here): The linear transformation $\mathbf{W}_{OV}$ applied to source tokens, determining *what* information is extracted and *where* it is written.

In our SVD analysis below, when we write $\mathbf{W}_{OV}\mathbf{X}_{\text{src}}$, we are analyzing what happens to information from a *single* source token. The attention pattern then determines which source tokens this transformation is applied to and with what weights. These two circuits operate *independently*: you can understand what the OV circuit does to each source token without knowing which tokens will be attended to.

In part (d), we showed that each attention head can only write to the column space of $\mathbf{W}_{OV}$, a low-dimensional subspace. But this raises deeper questions: *which* directions in the source token does the head read from? And *which* directions in the destination does it write to? The Singular Value Decomposition (SVD) provides a complete answer, revealing that attention heads are specialized communication channels between specific subspaces.

(i) Prove that $\text{rank}(\mathbf{W}_{OV}) \leq d_{\text{head}}$. Why does this imply that $\mathbf{W}_{OV}$ is a low-rank matrix (in typical transformer architectures)?

 All Rights Reserved. This may not be publicly shared without explicit permission.

(ii) Because $\mathbf{W}_{OV}$ is low-rank, its SVD has a special structure. Let $\mathbf{W}_{OV} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ be the (compact) SVD, where $\mathbf{U} \in \mathbb{R}^{d_{\text{model}} \times r}$, $\mathbf{\Sigma} \in \mathbb{R}^{r \times r}$, and $\mathbf{V} \in \mathbb{R}^{d_{\text{model}} \times r}$, with $r = \text{rank}(\mathbf{W}_{OV}) \leq d_{\text{head}}$. Consider a source token with residual stream vector $\mathbf{X}_{\text{src}} \in \mathbb{R}^{d_{\text{model}}}$. Show that the output of the OV circuit can be decomposed as:

$$\mathbf{W}_{OV}\mathbf{X}_{\text{src}} = \sum_{k=1}^{r} \sigma_k(\mathbf{V}_k^T \mathbf{X}_{\text{src}})\mathbf{U}_k \tag{7}$$

where $\mathbf{U}_k$ and $\mathbf{V}_k$ are the k-th columns of $\mathbf{U}$ and $\mathbf{V}$, respectively, and σ_k is the k-th singular value.

(iii) Based on the decomposition above, define precisely:

(1) The **read subspace**: Which directions in $\mathbf{X}_{\text{src}}$ can the head extract information from?

(2) The **write subspace**: Which directions in the destination residual stream can the head write to?

(3) What happens to information in $\mathbf{X}_{\text{src}}$ that is orthogonal to the read subspace?

Hint: Think about which matrix "touches" the input first (this determines what can be read) and which matrix determines the form of the output (this determines what can be written).

(iv) **[Connecting to the Value Projection]** The value vector computed by the head is $\mathbf{v} = \mathbf{W}_V\mathbf{X}_{\text{src}}$, which lives in $\mathbb{R}^{d_{\text{head}}}$. Explain in 2-3 sentences how the read subspace $\text{span}(\mathbf{V}_1, \ldots, \mathbf{V}_r)$ relates to what $\mathbf{W}_V$ can "see" in the source token. Why is it natural that the read subspace dimension is at most d_{head}?

(v) **[Synthesis]** The paper describes attention heads as moving information "from the residual stream of one token to another." In 2-3 sentences, explain how the SVD $\mathbf{W}_{OV} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ provides a complete characterization of this information movement. What role does each component ($\mathbf{V}$, $\mathbf{\Sigma}$, $\mathbf{U}$) play, and how does this relate to the attention pattern $\mathbf{A}$?

6. LoRA

A common strategy for adapting a large pre-trained model to a new task is to update only a subset of its parameters, keeping the rest frozen. Low-Rank Adaptation (LoRA) offers a more flexible approach to this idea. In this problem, we focus on a single weight matrix W with m rows and ℓ columns, where W_0 is the pre-trained value. During LoRA-based training, W is replaced by $W_0 + AB$, where W_0 remains frozen and only A and B are learnable. Here, A is an $m \times k$ matrix and B is a $k \times \ell$ matrix, typically with $k \ll \min(m, \ell)$.

(a) Suppose you are using LoRA to adapt a pretrained deep neural net to a new task and observe that the model is "underfitting" the training data. **What would you try to adjust in the LoRA to get better performance?**

(b) Suppose both A and B are initialized to all zeros. **Why will this cause problems for LoRA-based finetuning?**

Remember, this is going to be trained using SGD-style updates over a training set with a loss function.

(c) Consider the following pseudocode for LoRA initialization:

```
A = torch.nn.Parameter(torch.empty(m, k))
B = torch.nn.Parameter(torch.empty(k, l))
torch.nn.init.xavier_uniform_(A)
torch.nn.init.xavier_uniform_(B)
```

 All Rights Reserved. This may not be publicly shared without explicit permission.

Why might LoRA fine-tuning not work well with this initialization?

(d) **How much memory is required to store the LoRA adaptation weights (A and B)?** Assume we are using floats (4 bytes per real number) and give your answer in bytes. **How does this compare to storing a single full-rank adaptation matrix?**

7. Homework Process and Study Group

Citing sources and collaborators are an important part of life, including being a student!
We also want to understand what resources you find helpful and how much time homework is taking, so we can change things in the future if possible.

(a) **What sources (if any) did you use as you worked through the homework?**

(b) **If you worked with someone on this homework, who did you work with?**
List names and student ID's. (In case of homework party, you can also just describe the group.)

(c) **Roughly how many total hours did you work on this homework?**

Contributors: Gireeja Ranade, Sheng Shen, Shaojie Bai, Angelos Katharopoulos, Hao Peng, Olivia Watkins, Jake Austin, Dhruv Shah, Anant Sahai, Linyuan Gong, Sultan Daniels, Kevin Frans, Patrick Mendoza, Naman Jain.

 All Rights Reserved. This may not be publicly shared without explicit permission.