EECS182 HW07 Report

Edward Zhang

December 2025

1 Instructions

Input
Your goal is to provide a solution and explanation to my deep learning homework, demonstrating intermediate mathematical reasoning. Avoid hallucinating or making up definitions if information is missing; explicitly say so. Answer with clear math notation and concise justification. Do not skip derivation steps. If the problem contains multiple subparts (a), (b), (c), or (i),(ii), answer each clearly.
Output
I don't have your specific machine learning homework problem. Please provide the full problem statement, including all subparts, equations, datasets (if any), and any relevant context. Once you supply the details, I will:

- Show all intermediate reasoning step-by-step
- Use clear mathematical notation
- Provide justifications for each step
- Explicitly note if any information appears missing
- Structure the answer by subparts (a), (b), (c), etc.

You can paste the problem text directly into our conversation.

2 Problem 3

2.1 b

2.1.1 i

Input: Image of the Question
Output:

Given the loss function

$$\mathcal{L}(W_1, W_2; \mathbf{X}) = \|\mathbf{X} - W_2 W_1 \mathbf{X}\|_F^2,$$

with $\mathbf{X} \in \mathbb{R}^{m\times n}$ (zero-centered), $W_1 \in \mathbb{R}^{k\times m}$, $W_2 \in \mathbb{R}^{m\times k}$, and $k < m$, We compute the gradients with respect to W_1 and W_2 and set them to zero.

Define the error matrix $E = \mathbf{X} - W_2 W_1 \mathbf{X}$. The squared Frobenius norm can be written as

$$\mathcal{L} = \operatorname{tr}(E^\top E).$$

Taking differentials:

$$d\mathcal{L} = 2\ \operatorname{tr}(E^\top dE), \qquad dE = -(dW_2 W_1 + W_2 dW_1)\mathbf{X}.$$

Thus,

$$d\mathcal{L} = -2\ \operatorname{tr}\big(E^\top dW_2 W_1 \mathbf{X} + E^\top W_2 dW_1 \mathbf{X}\big)\,.$$

Using the cyclic property of the trace, rewrite each term:

$$\operatorname{tr}\big(E^\top dW_2 W_1 \mathbf{X}\big) = \operatorname{tr}\big(W_1 \mathbf{X} E^\top dW_2\big)\,,$$

$$\operatorname{tr}\big(E^\top W_2 dW_1 \mathbf{X}\big) = \operatorname{tr}\big(\mathbf{X} E^\top W_2 dW_1\big)\,.$$

Hence,

$$d\mathcal{L} = -2\ \operatorname{tr}\big(W_1 \mathbf{X} E^\top dW_2 + \mathbf{X} E^\top W_2 dW_1\big)\,.$$

Identifying the gradients (in numerator layout):

$$\frac{\partial \mathcal{L}}{\partial W_2} = -2E\mathbf{X}^\top W_1^\top, \qquad \frac{\partial \mathcal{L}}{\partial W_1} = -2W_2^\top E \mathbf{X}^\top.$$

Setting these to zero gives the first-order optimality conditions:

$$\boxed{(\mathbf{X} - W_2 W_1 \mathbf{X})\mathbf{X}^\top W_1^\top = 0}, \qquad \boxed{W_2^\top (\mathbf{X} - W_2 W_1 \mathbf{X})\mathbf{X}^\top = 0}.$$

Equivalently, defining the covariance matrix $C = \mathbf{X}\mathbf{X}^\top \in \mathbb{R}^{m\times m}$, the conditions become

$$\boxed{(C - W_2 W_1 C)W_1^\top = 0}, \qquad \boxed{W_2^\top (C - W_2 W_1 C) = 0}.$$

2.1.2 ii

Input: (ii) Show that the principal components U_k satisfy the optimality conditions outlined in (i). Solution to (ii): Showing U_k satisfies optimality conditions

We choose encoder and decoder matrices based on the principal components:

$$W_1 = U_k^\top \in \mathbb{R}^{k\times m}, \qquad W_2 = U_k \in \mathbb{R}^{m\times k},$$

where U_k contains the top k eigenvectors of the covariance matrix $C = \mathbf{X}\mathbf{X}^\top \in \mathbb{R}^{m\times m}$, so that $U_k^\top U_k = I_k$ and $CU_k = U_k S^2$, with $S = \operatorname{diag}(\sigma_1, \ldots, \sigma_k)$.

Step 1: Compute the reconstruction error matrix

The reconstruction is

$$W_2W_1\mathbf{X} = U_kU_k^\top\mathbf{X}.$$

The error matrix is

$$E = \mathbf{X} - U_kU_k^\top\mathbf{X} = (I_m - U_kU_k^\top)\mathbf{X}.$$

Step 2: Express the covariance and its eigendecomposition
Let

$$U = [\, U_k \mid U_\perp \,] \in \mathbb{R}^{m\times m}$$

be a full orthogonal matrix of eigenvectors of C, with $U_\perp$ corresponding to the remaining $m-k$ eigenvectors. Then

$$C = U\begin{bmatrix} S^2 & 0 \\ 0 & S_\perp^2 \end{bmatrix}U^\top = U_kS^2U_k^\top + U_\perp S_\perp^2 U_\perp^\top.$$

Step 3: Compute $E\mathbf{X}^\top$

$$E\mathbf{X}^\top = (I_m - U_kU_k^\top)\mathbf{X}\mathbf{X}^\top = (I_m - U_kU_k^\top)C.$$

Substitute the eigendecomposition of C:

$$\begin{aligned} E\mathbf{X}^\top &= (I_m - U_kU_k^\top)(U_kS^2U_k^\top + U_\perp S_\perp^2 U_\perp^\top) \\ &= (I_m - U_kU_k^\top)U_kS^2U_k^\top + (I_m - U_kU_k^\top)U_\perp S_\perp^2 U_\perp^\top \\ &= (U_k - U_kU_k^\top U_k)S^2U_k^\top + (U_\perp - U_kU_k^\top U_\perp)S_\perp^2 U_\perp^\top. \end{aligned}$$

Since $U_k^\top U_k = I_k$ and $U_k^\top U_\perp = 0$, this simplifies to

$$E\mathbf{X}^\top = (U_k - U_k)S^2U_k^\top + (U_\perp - 0)S_\perp^2U_\perp^\top = U_\perp S_\perp^2 U_\perp^\top.$$

Step 4: Verify the first optimality condition
The first condition from (i) is

$$E\mathbf{X}^\top W_1^\top = 0.$$

With $W_1^\top = (U_k^\top)^\top = U_k$,

$$E\mathbf{X}^\top W_1^\top = (U_\perp S_\perp^2 U_\perp^\top)U_k = U_\perp S_\perp^2(U_\perp^\top U_k).$$

But $U_\perp^\top U_k = 0$ because eigenvectors corresponding to distinct eigenvalues of a real symmetric matrix are orthogonal. Hence

$$E\mathbf{X}^\top W_1^\top = 0.$$

Step 5: Verify the second optimality condition
The second condition from (i) is

$$W_2^\top E\mathbf{X}^\top = 0.$$

Since $W_2 = U_k$,

$$W_2^\top E\mathbf{X}^\top = U_k^\top(U_\perp S_\perp^2 U_\perp^\top) = (U_k^\top U_\perp)S_\perp^2U_\perp^\top.$$

Again, $U_k^\top U_\perp = 0$, so

$$W_2^\top E\mathbf{X}^\top = 0.$$

3 Problem 4

3.1 a

Input: image of the question
Output
Based on the blog series, the authors began with a baseline ResNet model trained on CIFAR-10 for 200 epochs, achieving approximately 94.0% test accuracy in about 10 hours on a single GPU. After systematic optimizations—including architectural changes, hyperparameter tuning, mixed-precision training, and improved data loading—they reduced the training time to under 3 hours while raising the accuracy to about 94.5%.

3.2 b

Input: image of the question
Output
The blog underscores that single-GPU training efficiency can be dramatically improved through a combination of algorithmic, architectural, and systems-level optimizations. Important lessons include: the necessity of profiling to identify bottlenecks (often data loading or inefficient layers), the value of mixed precision for leveraging Tensor Cores, the impact of well-tuned hyperparameters (e.g., learning rate schedules, weight decay), and the benefits of simplifying the model architecture (e.g., removing unnecessary components). These steps collectively enable faster iteration and better performance without additional hardware.

3.3 c

Input: image of the question
Output
The use of mixed-precision training stood out because it directly exploits modern GPU capabilities (Tensor Cores) to nearly double throughput with minimal accuracy loss. This approach demonstrates how hardware-aware optimizations can yield substantial gains without changing the underlying algorithm. It also highlights the importance of maintaining numerical stability (via loss scaling) when reducing precision—a practical lesson applicable to many deep-learning tasks.

4 Problem 7

4.1 a

Input: image of the question
Output
In the proposed design, the encoder hidden state h_t at each source timestep

t is fed as input to the decoder at the same timestep t. This assumes a one-to-one alignment between source and target timesteps. However, in machine translation, the source sentence length T_s (number of Spanish words) and target sentence length T_t (number of English words) are generally different ($T_s \neq T_t$). Hence, when $T_t > T_s$, the decoder would run out of encoder hidden states after T_s steps; when $T_t < T_s$, extra encoder states would be unused. Even if lengths were equal, word-order differences (e.g., adjective–noun order in Spanish vs. English) mean that the relevant source information for a given target word may not reside in the encoder state at the same positional index. The rigid coupling ignores variable alignment and prevents the decoder from accessing a condensed representation of the full source sentence, which is precisely what the final encoder hidden state (or an attention mechanism) provides.

4.2 b

Input: image of the question
Output
During training with teacher forcing, the decoder inputs are the ground-truth target tokens shifted by one timestep, starting with the start-of-sequence token. For the target sentence "I see a dog," the sequence of tokens (including special tokens) is: `<sos>, I, see, a, dog, <eos>`.

Thus, the five tokens passed into the decoder are:

- w_1: `<sos>`
- w_2: `I`
- w_3: `see`
- w_4: `a`
- w_5: `dog`

At each step t, the decoder receives the preceding ground-truth token (or `<sos>` at the first step) and is trained to predict the next token in the sequence.

4.3 c

Input: image of the question
Output
At evaluation time, the decoder generates tokens autoregressively using its own previous outputs as inputs. Starting with the start-of-sequence token, each subsequent input is the word generated at the previous timestep.

Given that the model incorrectly generates the translation "I saw a dog," the sequence of tokens passed into the decoder for the first five timesteps is:

- w_1: `<sos>`
- w_2: `I`

- w_3: `saw`
- w_4: `a`
- w_5: `dog`

After receiving “dog” as input at step 5, the decoder would typically generate the `<eos>` token to end the sequence.

5 Problem 8

5.1 a

5.1.1 i

Input: image of the question
Output
Answer (a)(i):
We need **2 linear layers**.

Explanation:
The linear autoencoder consists of an encoder that maps the input $\mathbf{x} \in \mathbb{R}^m$ to a hidden representation $\mathbf{z} = W_1\mathbf{x} \in \mathbb{R}^k$ (first linear layer) and a decoder that maps $\mathbf{z}$ to the reconstruction $\hat{\mathbf{x}} = W_2\mathbf{z} \in \mathbb{R}^m$ (second linear layer). Since no nonlinearities are used, these correspond exactly to two separate linear layers in a neural network. The self-supervised training uses each data point as both input and target, and SGD optimizes both weight matrices simultaneously.

5.1.2 ii

Input: image of the question
Output
Answer (a)(ii):
We will use `nn.MSELoss` (Mean Squared Error loss).

Explanation:
The reconstruction term

$$\frac{1}{n} \|\mathbf{X} - W_2 W_1 \mathbf{X}\|_F^2$$

is the average squared difference across all entries, which is exactly the mean squared error. In PyTorch, `nn.MSELoss` computes the average squared error between each element in the input and target. The regularization terms $\lambda\|W_1\|_F^2 + \lambda\|W_2\|_F^2$ can be handled separately via weight decay in the optimizer or by adding them manually, but the primary loss function for the self-supervised task is MSE.

5.1.3 iii

Input: image of the question
Output
We need SGD optimizer (with weight decay set to 0).
To optimize the objective (6) exactly as written, we use a neural network with two linear layers and MSE loss. The objective does not include regularization, so weight decay should not be applied. Dropout, Layer Norm, and Batch Norm are not part of the linear autoencoder formulation and would alter the optimization problem. The SGD optimizer (with weight decay = 0) is the appropriate optimization algorithm for minimizing the loss using stochastic gradient descent.

5.2 b

Input: image of the question
Output
Solution to (b):
Yes, a small nonzero λ induces an inductive bias toward W_2 having approximately orthonormal columns.

Reasoning:
The regularized loss $\mathcal{L}_\lambda$ includes Frobenius penalties on W_1 and W_2. For fixed product $A = W_2 W_1$, the minimum of $\|W_1\|_F^2 + \|W_2\|_F^2$ is achieved when W_1 and W_2 have aligned singular vectors and equal singular values $\sigma_i(W_1) = \sigma_i(W_2) = \sqrt{\alpha_i}$, where α_i are the singular values of A. Thus, the problem reduces to minimizing over A (with $\operatorname{rank}(A) \le k$):

$$\frac{1}{n}\left\|X - AX\right\|_F^2 + 2\lambda \|A\|_*,$$

where $\|A\|_* = \sum_i \alpha_i$ is the nuclear norm.

When λ is small, the reconstruction term dominates and pushes A toward the orthogonal projector P_k onto the top-k principal subspace of X, whose nonzero singular values are all 1. The nuclear norm penalty then slightly shrinks these singular values to

$$\alpha_i \approx 1 - \frac{\lambda}{\sigma_i^2}$$

(where σ_i^2 are eigenvalues of $\frac{1}{n}XX^\top$). Consequently, the optimal singular values of W_2 satisfy

$$\sigma_i(W_2) = \sqrt{\alpha_i} \approx 1 - \frac{\lambda}{2\sigma_i^2},$$

which approach 1 as $\lambda \to 0$. Singular values equal to 1 imply W_2 has orthonormal columns. Hence, small λ biases W_2 toward approximately orthonormal columns.

Input: try to not use the nuclear norm or any other concept that is not defined

Output
Answer (b):
Yes, a small nonzero λ introduces an inductive bias toward W_2 having approximately orthonormal columns.

Reasoning:
Consider the SVDs $W_1 = U_1 \Sigma_1 V_1^\top$ and $W_2 = U_2 \Sigma_2 V_2^\top$. The product $A = W_2 W_1$ has singular values α_i. The regularization term $\lambda(\|W_1\|_F^2 + \|W_2\|_F^2)$ is minimized for a given A when W_1 and W_2 share the same left/right singular vectors and their singular values satisfy $\sigma_i(W_1) = \sigma_i(W_2) = \sqrt{\alpha_i}$. In such an optimal factorization, W_2 has orthogonal columns with norms $\sqrt{\alpha_i}$.

For $\lambda = 0$, the optimal A is the orthogonal projector onto the top-k principal subspace, whose nonzero singular values are exactly 1. Hence, an optimal factorization yields W_2 with orthonormal columns. For small $\lambda > 0$, the reconstruction term still dominates, so the optimal A remains close to that projector, giving $\alpha_i \approx 1 - \lambda/\sigma_i^2$ (where σ_i^2 are the eigenvalues of $\frac{1}{n} X X^\top$). Thus, $\sqrt{\alpha_i} \approx 1 - \lambda/(2\sigma_i^2) \approx 1$. Consequently, the columns of W_2 are nearly orthonormal (orthogonal with norms close to 1). The regularization therefore biases solutions toward approximately orthonormal columns in W_2.