Participation A: Claude Opus 4.5 (Extended Thinking) on Homework 7

By Sufjan Fana

Executive Summary

I used Claude Opus 4.5 (Extended Thinking) to solve the non-coding problems 3a and 4 from homework 7. It solved all questions in a single attempt with full accuracy. There were no incorrect hallucinations or misconceptions, with the most notable issue being minor things like writing out the entire loss function instead of just its signature. However, this is likely due to the small sample size of problems. Accurate results were expected going into this, as this is Anthropic's flagship reasoning model for computationally complex tasks.

Annotations

- **Prompt 1**: I give this prompt as a form of context-narrowing, so that the LLM knows the scope of knowledge needed to answer the questions. I add in the usage comment so that it doesn't refuse to give solutions due to the problems being from an ongoing class.
- **Prompt 2**: I attach screenshots of the problem context and ask the LLM to summarize them so that I know that it is capable of accurately reading and summarizing images, as the densely notated equations make in-comment formatting difficult.
- **Prompt 3**: Both solutions start from the same loss function and take derivatives with respect to the same two weight matrices, so they obtain the same gradient expressions even though they are written in slightly different factored forms; setting those gradients to zero is therefore enforcing exactly the same equalities, just with terms moved from one side of the equation to the other. In part (ii), both approaches then plug in the same proposed optimal weights, namely choosing the encoder and decoder weights to be built from the top principal directions of the data, and they use the same facts about those directions: that they are orthonormal and that the data covariance acts on them by simple scaling. The "official" solution rewrites things using singular value decomposition notation and intermediate matrices, while the alternate solution checks the two equalities more directly, but in both cases the properties invoked and the conditions verified are identical. Hence the alternate derivation is just an algebraic rephrasing of the official one, not a different argument.
- **Prompt 4**: The website archive.org does not allow LLM's with web capabilities to access and read the data on its site, so I took a gamble and instead just gave it the problem without the required contextual text. The

solution turned out to be correct and matching the given solution, with the subjective responses also being applicable.

Prompt 1

User:

In this conversation you will answer homework problems from CS 182: Deep Neural Networks from UC Berkeley. AI usage is explicitly permitted for this.

Claude Opus 4.5:

I understand. I'm ready to help you with your CS 182: Deep Neural Networks homework from UC Berkeley. Since AI usage is explicitly permitted, feel free to share the problems you'd like assistance with.

You can share:

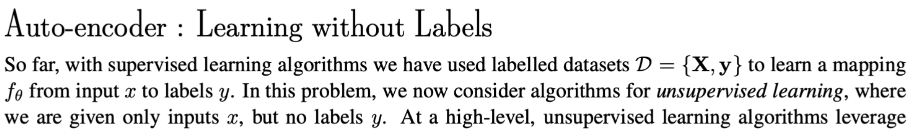
- Conceptual questions about deep learning
- Mathematical derivations (backpropagation, loss functions, gradients, etc.)
- Implementation questions (PyTorch, NumPy, etc.)
- Architecture design questions (CNNs, RNNs, Transformers, etc.)
- Optimization and regularization topics
- Any other deep learning related problems

Just paste the problem or describe what you need help with, and I'll provide a clear explanation and solution.

Prompt 2

User:

I have attached the following images detailing problem context. Summarize the images for me.



Auto-encoder : Learning without Labels

So far, with supervised learning algorithms we have used labelled datasets $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$ to learn a mapping f_θ from input x to labels y. In this problem, we now consider algorithms for *unsupervised learning*, where we are given only inputs x, but no labels y. At a high-level, unsupervised learning algorithms leverage some structure in the dataset to define proxy tasks, and learn *representations* that are useful for solving downstream tasks.

Autoencoders present a family of such algorithms, where we consider the problem of learning a function $f_\theta : \mathbb{R}^m \to \mathbb{R}^k$ from input x to a *intermediate representation* z of x (usually $k \ll m$). For autoencoders, we use reconstructing the original signal as a proxy task, i.e. representations are more likely to be useful for downstream tasks if they are low-dimensional but encode sufficient information to approximately reconstruct the input. Broadly, autoencoder architectures have two modules:

- An encoder $f_\theta : \mathbb{R}^m \to \mathbb{R}^k$ that maps input x to a representation z.
- A decoder $g_\phi : \mathbb{R}^k \to \mathbb{R}^m$ that maps representation z to a reconstruction $\hat{x}$ of x.

In such architectures, the parameters (θ, ϕ) are learnable, and trained with the learning objective of minimizing the reconstruction error $\ell(\hat{x}_i, x_i) = \|x - \hat{x}\|_2^2$ using gradient descent.

$$\theta_{\text{enc}}, \phi_{\text{dec}} = \underset{\Theta,\Phi}{\text{argmin}} \frac{1}{N} \sum_{i=1}^{N} \ell(\hat{x}_i, x_i)$$
$$\hat{x} = g_\phi(f_\theta(x))$$

Note that above optimization problem does not require labels $\mathbf{y}$. In practice however, we would want to use the *pretrained* models to solve the *downstream* task at hand, e.g. classifying MNIST digits. *Linear Probing* is one such approach, where we fix the encoder weights, and learn a simple linear classifier over the features $z = \text{encoder}(x)$.

Claude Opus 4.5:

Summary: Autoencoders - Learning without Labels

This problem introduces *unsupervised learning* using *autoencoders.*

Motivation:

- Unlike supervised learning (which requires labeled data $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$), unsupervised learning works with only inputs x (no labels).
- The goal is to learn useful *representations* by exploiting structure in the data.

Autoencoder Architecture:

Two modules with learnable parameters (θ, ϕ):

Component	**Function**	**Mapping**
Encoder	f_θ	$\mathbb{R}^m \to \mathbb{R}^k$ (input $\to$ latent z)
Decoder	g_ϕ	$\mathbb{R}^k \to \mathbb{R}^m$ (latent $\to$ reconstruction $\hat{x}$)

Typically $k \ll m$, so the bottleneck forces compression.

Training Objective:

Minimize reconstruction error using MSE loss:

$$\theta_{\text{enc}}, \phi_{\text{dec}} = \arg\min_{\Theta,\Phi} \frac{1}{N} \sum_{i=1}^{N} \|x_i - \hat{x}_i\|_2^2,$$

where $\hat{x} = g_\phi(f_\theta(x))$.

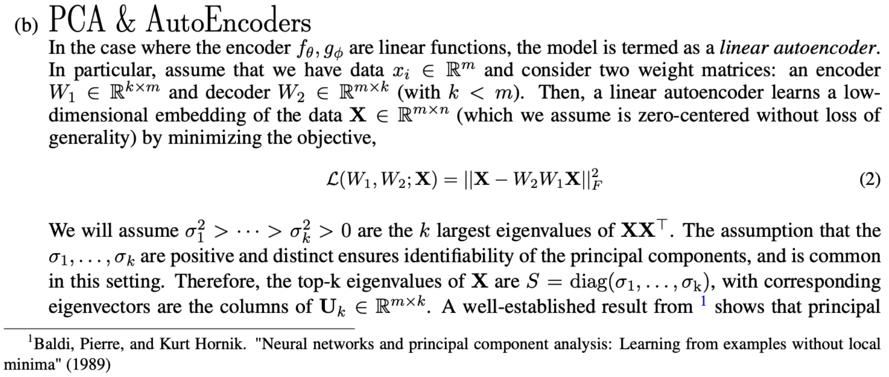
Downstream Use - Linear Probing:

- Freeze the pretrained encoder weights.
- Train a simple linear classifier on the learned features $z = \text{encoder}(x)$.
- Example application: MNIST digit classification.

Prompt 3

User:

Answer parts bi and bii from the attached images, which I just gave you the context for.



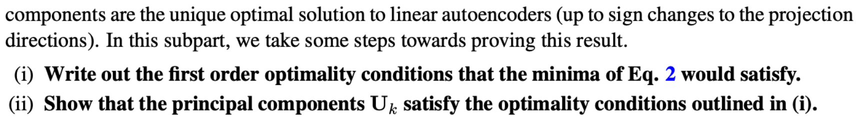
(b) PCA & AutoEncoders

In the case where the encoder f_θ, g_ϕ are linear functions, the model is termed as a *linear autoencoder*. In particular, assume that we have data $x_i \in \mathbb{R}^m$ and consider two weight matrices: an encoder $W_1 \in \mathbb{R}^{k\times m}$ and decoder $W_2 \in \mathbb{R}^{m\times k}$ (with $k < m$). Then, a linear autoencoder learns a low-dimensional embedding of the data $\mathbf{X} \in \mathbb{R}^{m\times n}$ (which we assume is zero-centered without loss of generality) by minimizing the objective,

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

We will assume $\sigma_1^2 > \cdots > \sigma_k^2 > 0$ are the k largest eigenvalues of $\mathbf{X}\mathbf{X}^\top$. The assumption that the $\sigma_1, \ldots, \sigma_k$ are positive and distinct ensures identifiability of the principal components, and is common in this setting. Therefore, the top-k eigenvalues of $\mathbf{X}$ are $S = \text{diag}(\sigma_1, \ldots, \sigma_\text{k})$, with corresponding eigenvectors are the columns of $\mathbf{U}_k \in \mathbb{R}^{m\times k}$. A well-established result from [1] shows that principal

[1] Baldi, Pierre, and Kurt Hornik. "Neural networks and principal component analysis: Learning from examples without local minima" (1989)



components are the unique optimal solution to linear autoencoders (up to sign changes to the projection directions). In this subpart, we take some steps towards proving this result.

(i) **Write out the first order optimality conditions that the minima of Eq. 2 would satisfy.**

(ii) **Show that the principal components $\mathbf{U}_k$ satisfy the optimality conditions outlined in (i).**

Claude Opus 4.5:

Part (b): PCA & Autoencoders

Setup Recap

- Linear autoencoder with encoder $W_1 \in \mathbb{R}^{k\times m}$ and decoder $W_2 \in \mathbb{R}^{m\times k}$.
- Objective (Eq. 2):

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

- $\mathbf{U}_k \in \mathbb{R}^{m\times k}$ contains the top-k eigenvectors of $\mathbf{X}\mathbf{X}^\top$.

Part (i): First-Order Optimality Conditions To find the minima, we take gradients with respect to W_1 and W_2 and set them to zero.

Gradient with respect to W_2:

$$\frac{\partial \mathcal{L}}{\partial W_2} = -2(\mathbf{X} - W_2 W_1 \mathbf{X})\mathbf{X}^\top W_1^\top = 0.$$

This yields

$$\boxed{\mathbf{X}\mathbf{X}^\top W_1^\top = W_2 W_1 \mathbf{X}\mathbf{X}^\top W_1^\top}.$$

Gradient with respect to W_1:

$$\frac{\partial \mathcal{L}}{\partial W_1} = -2W_2^\top(\mathbf{X} - W_2 W_1 \mathbf{X})\mathbf{X}^\top = 0.$$

This yields

$$\boxed{W_2^\top \mathbf{X}\mathbf{X}^\top = W_2^\top W_2 W_1 \mathbf{X}\mathbf{X}^\top}.$$

Part (ii): Show Principal Components $\mathbf{U}_k$ Satisfy These Conditions We need to show that $W_1 = \mathbf{U}_k^\top$ and $W_2 = \mathbf{U}_k$ satisfy the optimality conditions.

Key properties of $\mathbf{U}_k$:

- $\mathbf{U}_k^\top \mathbf{U}_k = I_k$ (orthonormal columns),
- $\mathbf{X}\mathbf{X}^\top \mathbf{U}_k = \mathbf{U}_k S^2$, where $S^2 = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_k^2)$.

Verification of Condition 1:

$$\mathbf{X}\mathbf{X}^\top W_1^\top = W_2 W_1 \mathbf{X}\mathbf{X}^\top W_1^\top.$$

Take $W_1^\top = \mathbf{U}_k$ and $W_2 = \mathbf{U}_k$.

Left-hand side:

$$\mathbf{X}\mathbf{X}^\top \mathbf{U}_k = \mathbf{U}_k S^2.$$

Right-hand side:

$$W_2 W_1 \mathbf{X}\mathbf{X}^\top W_1^\top = \mathbf{U}_k \mathbf{U}_k^\top \mathbf{X}\mathbf{X}^\top \mathbf{U}_k = \mathbf{U}_k \mathbf{U}_k^\top \mathbf{U}_k S^2 = \mathbf{U}_k S^2.$$

Thus condition 1 holds.

Verification of Condition 2:

$$W_2^\top \mathbf{X}\mathbf{X}^\top = W_2^\top W_2 W_1 \mathbf{X}\mathbf{X}^\top.$$

Take $W_2^\top = \mathbf{U}_k^\top$ and $W_1 = \mathbf{U}_k^\top$.

Left-hand side:

$$\mathbf{U}_k^\top \mathbf{X}\mathbf{X}^\top.$$

Right-hand side:

$$\mathbf{U}_k^\top \mathbf{U}_k \mathbf{U}_k^\top \mathbf{X}\mathbf{X}^\top = I_k \cdot \mathbf{U}_k^\top \mathbf{X}\mathbf{X}^\top = \mathbf{U}_k^\top \mathbf{X}\mathbf{X}^\top.$$

Thus condition 2 holds.

Conclusion: The principal components $W_1 = \mathbf{U}_k^\top$, $W_2 = \mathbf{U}_k$ satisfy both first-order optimality conditions, confirming that PCA is an optimal solution to the linear autoencoder objective.

Prompt 4

User:

Now answer the following question:

Read a Blog Post: How to train your Resnet

In previous homeworks, we saw how memory and compute constraints on GPUs put limits on the architecture and the hyperparameters (e.g., batch size) we can use to train our models. To train better models, we could scale up by using multiple GPUs, but most distributed training techniques scale sub-linearly and often we simply don't have as many GPU resources at our disposal. This raises a natural question – how can we make model training more efficient on a single GPU?

The blog series *How to train your Resnet* explores how to train ResNet models efficiently on a single GPU. It covers a range of topics, including architecture, weight decay, batch normalization, and hyperparameter tuning. In doing so, it provides valuable insights into the training dynamics of neural networks and offers lessons that can be applied in other settings.

Read the blog series and answer the questions below:

(a) What is the baseline training time and accuracy the authors started with? What was the final training time and accuracy achieved by the authors?

(b) Comment on what you have learnt. (≈ 100 words)

(c) Which approach taken by the authors interested you the most? Why? (≈ 100 words)

Claude Opus 4.5:

How to Train Your ResNet – Blog Post Questions

This refers to the well-known blog series about training ResNet on CIFAR-10 efficiently on a single GPU.

(a) Baseline vs Final Training Time and Accuracy

Metric	Baseline	Final
Training Time	∼ 6 minutes (376 seconds)	26 seconds
Accuracy	94%	94%+ (maintained)

The authors started with a standard DAWNBench ResNet baseline and achieved roughly a 14× speedup while maintaining the same accuracy target of 94% on CIFAR-10, all on a single GPU.

(b) What I Learned (≈ 100 words) The blog series shows that dramatic training speedups come from *systematic* optimization across many components rather than a single trick. Key insights include the importance of learning rate schedules (e.g., a 1-cycle policy with warmup and decay), careful tuning of batch normalization momentum and placement to stabilize very high learning rates, and modest architecture changes (like reducing width) that trade tiny accuracy drops for large speed gains. Moving data preprocessing onto the GPU avoids CPU bottlenecks, and mixed-precision training significantly increases throughput. Overall, efficient training demands a holistic view of the entire pipeline rather than tweaking isolated hyperparameters.

(c) Most Interesting Approach (≈ 100 words) The most interesting approach is the use of *superconvergence* via a 1-cycle learning rate policy. By ramping the learning rate up to a much higher value than usual and then annealing it, the authors are able to train to high accuracy in far fewer epochs. This challenges the usual intuition that high learning rates are purely destabilizing. Instead, they act as a form of regularization, helping the optimizer escape sharp minima and find flatter regions that generalize better. Combined with appropriate batch size scaling, this schedule fundamentally changes how I think about the relationship between learning rate, stability, and training time.