CSED 502
Deep Learning
Winter 2026 Quiz 2

SOLUTIONS

March 11, 2026

Full Name (as on Gradescope): _____

UW Net ID: _____

Question	Score
True/False (40 pts)	
Multiple Choice (60 pts)	
Total (100 pts)	

Welcome to the CSED 502 Quiz!

- The exam is 45 min and is **double-sided**.
- You may use a non-graphing calculator and no other electronic devices.
- One handwritten double sided cheat sheet is allowed.

I understand and agree to uphold the University of Washington Student Conduct Code during this exam.

1 True/False (30 points) - Recommended 20 Minutes

Fill in the circle next to your choice (like this: ●). No explanations are required.

Each question is worth 4 points. You will receive partial credit if reasonable short explanation is provided, even if the answer is wrong.

1.1 Two 3×3 convolutions with D input and D output channels have more parameters than one 5×5 convolution with D input and D output channels.

◯ True

◯ False **SOLUTION:**

False. Two 3×3 convolutions have $2 \times (3 \times 3 \times D \times D) = 18D^2$ parameters. One 5×5 convolution has $5 \times 5 \times D \times D = 25D^2$ parameters. $18D^2 < 25D^2$.

1.2 Two models with the same number of parameters but different architectures will require roughly the same amount of memory to perform a forward and backward pass on the same data.

◯ True

◯ False **SOLUTION:**

False. Memory during training is dominated by storing activations (intermediate values) for backpropagation, not by the number of parameters. A deeper model with fewer parameters per layer can require significantly more activation memory than a shallow, wide model with the same total parameter count.

1.3 Generally, larger vocabulary sizes for tokenizers will lead to shorter sequence lengths.

◯ True

◯ False **SOLUTION:**

True. With a larger vocabulary, each token can represent a longer substring of text, so fewer tokens are needed to encode the same input.

1.4 The number of parameters in an RNN increases linearly with the sequence length.

◯ True

◯ False **SOLUTION:**

False, the number of params remains the same

1.5 Increasing the number of attention heads in a multi-head attention layer increases the total number of attention maps computed, since each head produces its own separate attention map.

◯ True

◯ False **SOLUTION:**

True

1.6 Imagine you are training a language model that consists of a single stack of attention and MLP blocks, rather than an encoder-decoder. You should use unmasked (bidirectional) attention during training so the model can see the full context, and then switch to causal (masked) attention at generation time so the model only attends to previously generated tokens.

◯ True

◯ False **SOLUTION:**

False, you need to use the same type during training and testing

1.7 If you want a network to produce two different types of outputs simultaneously, such as a probability distribution over class labels and bounding box coordinates, you can achieve this by attaching multiple separate fully connected heads to the same shared feature representation, each with its own output dimension and loss function

◯ True

◯ False **SOLUTION:**

True

1.8 A key idea in contrastive self-supervised learning (e.g., SimCLR) is that two augmented views of the same image should have similar representations, while views of different images should be pushed apart.

◯ True

◯ False **SOLUTION:**

True

1.9 Transfer learning only works when the source task and target task have the same set of output classes, because the learned features are specific to the classes the network was originally trained on

◯ True

◯ False **SOLUTION:**

False, you can transfer previous layers and reinitialize the final layer

1.10 Self-supervised learning models are <u>defined</u> by the fact that the model's output is a similarity score between embeddings or a reconstructed input, rather than a probability distribution over discrete class labels.

◯ True

◯ False **SOLUTION:**

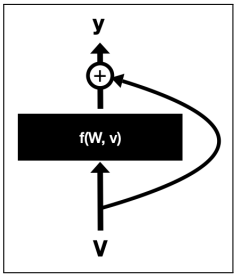
False, its defined by how the labels are obtained, not by the format of the output

2 Multiple Choice (60 points) - Recommended 25 Minutes

Fill in the circle next to the letter(s) of your choice (like this: ●). No explanations are required. **Choose ALL options that apply.**

Each question is worth 15 points and the answer may contain multiple correct options, or none at all. Selecting all of the correct options and none of the incorrect options will get full credit. For questions with multiple correct options, each incorrect or missing selection gets a 5-point deduction (up to 15 points).

2.1 You are training a neural network. At some point in the network, there is an activation vector $\mathbf{v}$ of shape $(1 \times d)$. This vector is fed into a linear layer f with weight matrix $\mathbf{W}$ of shape $(d \times d)$ (i.e., $f(\mathbf{v}) = \mathbf{v}\mathbf{W}$). There is a residual connection from just before f to just after f, as shown to the right.



The output is:

$$\mathbf{y} = f(\mathbf{v}) + \mathbf{v} = \mathbf{v}\mathbf{W} + \mathbf{v}$$

Assume that $\frac{\partial \mathcal{L}}{\partial \mathbf{y}}$ is given (shape $1 \times d$). Let $\mathbf{I}$ denote the identity matrix. What is $\frac{\partial \mathcal{L}}{\partial \mathbf{v}}$?

- ◯ A: $\frac{\partial \mathcal{L}}{\partial \mathbf{y}}\mathbf{W}^T$
- ◯ B: $\frac{\partial \mathcal{L}}{\partial \mathbf{y}}\mathbf{W}^T + \mathbf{I}$
- ◯ C: $\frac{\partial \mathcal{L}}{\partial \mathbf{y}}(\mathbf{I} + \mathbf{W}^T)$
- ◯ D: $(\mathbf{I} + \frac{\partial \mathcal{L}}{\partial \mathbf{y}})\mathbf{W}^T$

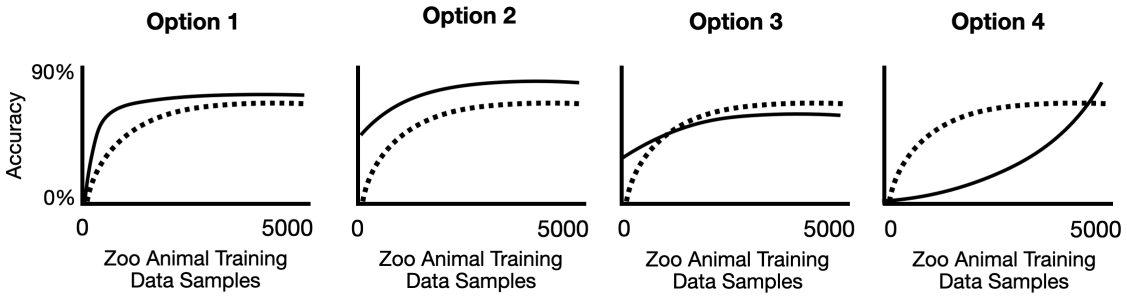
SOLUTION:
C. Since $\mathbf{y} = \mathbf{v}\mathbf{W} + \mathbf{v} = \mathbf{v}(\mathbf{W} + \mathbf{I})$, we have $\frac{\partial \mathbf{y}}{\partial \mathbf{v}} = (\mathbf{W} + \mathbf{I})^T = \mathbf{I} + \mathbf{W}^T$, so by the chain rule:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{v}} = \frac{\partial \mathcal{L}}{\partial \mathbf{y}}(\mathbf{I} + \mathbf{W}^T)$$

This has shape $(1 \times d)(d \times d) = (1 \times d)$ ✓**(A)** would be correct if there were no residual connection—it is missing the gradient from the skip path. **(B)** incorrectly adds the identity matrix as a standalone term—the skip connection contributes $\frac{\partial \mathcal{L}}{\partial \mathbf{y}}$, not $\mathbf{I}$, to the gradient. **(D)** incorrectly adds $\mathbf{I}$ to the upstream gradient $\frac{\partial \mathcal{L}}{\partial \mathbf{y}}$ rather than to $\mathbf{W}^T$.

2.2 You pretrain a convolutional neural network on ImageNet. You then transfer it to a smaller Zoo Animal classification dataset with 500 classes by removing the final fully-connected classification layer and replacing it with a new, randomly initialized fully-connected layer for your target classes. You then fine-tune the entire network on your target data. The **dashed line** in each plot below shows the accuracy curve when training the same architecture from **random initialization** on the target dataset. The **solid line** shows the accuracy curve of the fine-tuned model.

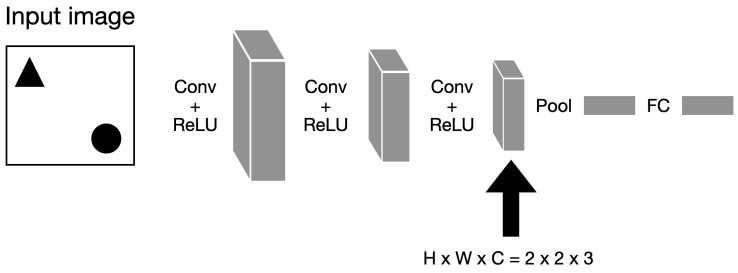
Which plot best represents the **expected training curve** of the fine-tuned model compared to training from scratch? **Select one.**



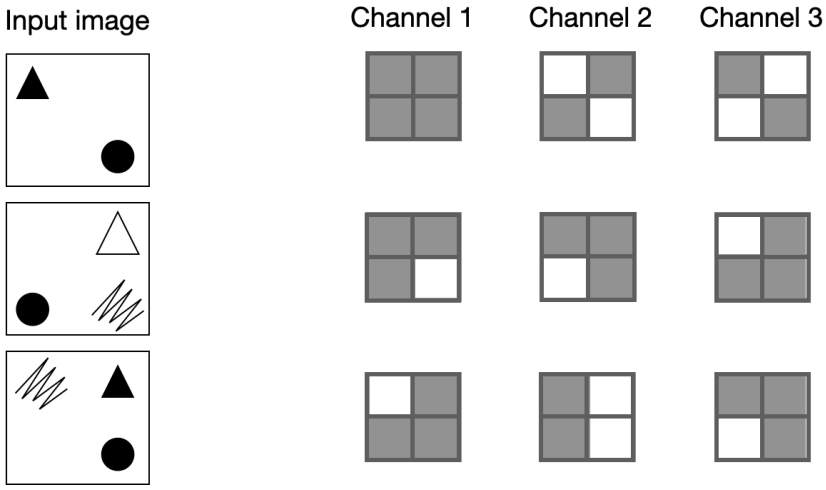
- ◯ A: Option 1
- ◯ B: Option 2
- ◯ C: Option 3
- ◯ D: Option 4

SOLUTION:
A. Since the final fully-connected layer is replaced with a new, randomly initialized one, both models start at a similarly low accuracy. However, the fine-tuned model benefits from pretrained features in the earlier layers, leading to two key differences: **Faster learning:** The pretrained convolutional features are already useful for visual recognition—especially since ImageNet contains many animal classes. The new classification head only needs to learn a mapping from these good features to the target classes, so accuracy rises much more quickly. **Higher final accuracy:** With a smaller target dataset, training from scratch is more prone to underfitting or learning weaker features. The pretrained initialization converges to a better solution. Option 1 correctly shows both effects: same starting point, faster rise, and higher final accuracy. **(B)** Incorrectly shows the fine-tuned model starting at high accuracy. Since the final FC layer is randomly initialized, accuracy at step 0 should be near chance regardless of pretraining. **(C)** Shows the from-scratch model learning faster initially and outperforming the fine-tuned model early on. This is unlikely when transferring from a closely related domain—pretrained features should help immediately, not hinder early training. **(D)** Shows the fine-tuned model learning slower than from-scratch early on, only overtaking it late. Pretrained features should accelerate learning from the start, not slow it down.

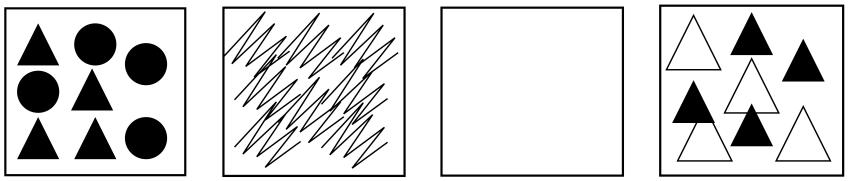
2.3 You have trained the CNN shown below and want to understand what features it has learned at a specific layer. You examine the activations at the layer indicated by the arrow, which has shape $H \times W \times C = 2 \times 2 \times 3$.



You pass three different input images through the network and visualize the resulting activations across all 3 channels. In the activation grids below, **white squares indicate high activation values** and **gray squares indicate low activation values**.



You then perform **gradient ascent** on a randomly initialized input image to maximize the sum of all 4 spatial values in **Channel 3** of this activation. Which of the following images would you most expect gradient ascent to produce? **Select one.**



Option 1 Option 2 Option 3 Option 4

- ◯ A: Option 1
- ◯ B: Option 2
- ◯ C: Option 3
- ◯ D: Option 4

SOLUTION:
C. To determine what Channel 3 detects, we examine where it has high activation (white) across the three input images.

In every case, Channel 3 activates exactly where there is *no object*—it has learned to detect empty space. To maximize all 4 spatial values of Channel 3 via gradient ascent, we want an input with nothing in any quadrant. Option 3 (a blank image) achieves this.

2.4 For this question, use the following notation and assumptions:

- Let $T(f(x))$ denote the time it takes to compute $f(x)$
- Let $\text{Encode}_m(n)$ denote the process of encoding a sequence of length n into a vector representation using model m (e.g., converting a user's input question into a hidden representation for later processing)
- Let $\text{Decode}_m(n)$ denote the process of autoregressively generating n new tokens using model m (e.g., generating a response to a user's question, or captioning an unseen image)
- Let tf denote a Transformer and let rnn denote an RNN

Assume we have **infinite compute and memory**, so any operations that can be parallelized are fully parallelized. Under this assumption:

- Independent operations run in parallel: $T(f(x) + g(x)) = \max\big(T(f(x)),\ T(g(x))\big)$
- Sequential dependencies still require waiting: $T(g(f(x))) \approx T(f(x)) + T(g(y))$ where $y = f(x)$

Select all statements that are TRUE:

◯ A: $T(\text{Encode}_{tf}(n)) \approx T(\text{Encode}_{tf}(2n))$

◯ B: $T(\text{Encode}_{rnn}(n)) \approx T(\text{Encode}_{rnn}(2n))$

◯ C: $T(\text{Decode}_{tf}(n)) \approx T(\text{Decode}_{tf}(2n))$

◯ D: $T(\text{Decode}_{rnn}(n)) \approx T(\text{Decode}_{rnn}(2n))$

SOLUTION:
A. **(A) TRUE:** Transformer encoding processes all tokens in parallel via self-attention. With infinite compute, doubling the sequence length does not significantly increase wall-clock time. **(B) FALSE:** RNN encoding is inherently sequential—each hidden state depends on the previous one. Doubling the sequence length approximately doubles the encoding time: $T(\text{Encode}_r(2n)) \approx 2 \cdot T(\text{Encode}_r(n))$. **(C) FALSE:** Autoregressive decoding in a Transformer generates tokens one at a time, where each new token depends on all previously generated tokens. Doubling the number of generated tokens approximately doubles the decoding time. **(D) FALSE:** RNN decoding is also autoregressive—each generated token depends on the previous hidden state. Doubling the output length approximately doubles the decoding time.