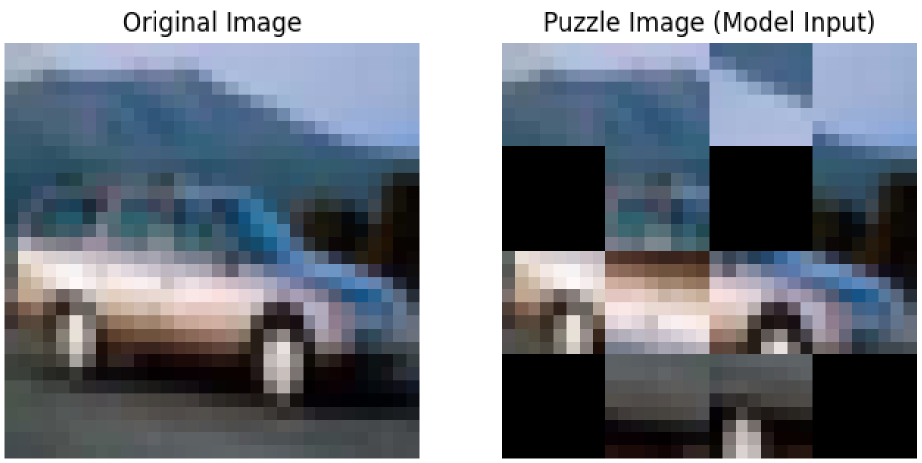
Report

Q1.

i) Create non-overlapping patches of 8x8. Take 50% patches at random, that is 8 patches. Let us call this set A. Randomly mask 50% or 4 patches of A. On the remaining unmasked patches of A, apply random rotation: 0, 90, 180, 270.

An image from CIFAR-10 was processed with the following steps:

- **Image Patching:** The 32x32 image was divided into **16** non-overlapping **8x8** patches.
- **Random Subset (Set A):** **8** random patches were selected to form "Set A".
    - *Selected Indices:* `[13, 4, 14, 8, 10, 9, 6, 5]`
- **Patch Augmentation:**
    - **Masking:** **4** patches from Set A were masked (turned black).
        - *Masked Indices:* `[14, 5, 13, 10]`
    - **Rotation:** The remaining **4** patches in Set A were rotated by **270°**.
        - *Rotated Indices:* `[9, 8, 4, 6]`



ii) Append a CLS token. On this token, apply a linear layer which can classify the rotation of the rotated patches. You need to classify into one of the four Classes.
iii) On the masked tokens apply a linear layer for prediction task.

ViT with Multi-Task Heads

The pre-trained ViT encoder was adapted for two simultaneous tasks by adding separate prediction "heads":

- **Rotation Head:** A linear layer was attached to the `[CLS]` **token** output. Its sole purpose is to classify the image's rotation into one of four angles (0°, 90°, 180°, 270°).
- **Reconstruction Head:** A second linear layer was attached to the outputs of the **masked patch tokens**. This head is trained to predict the original pixel values of the patches that were blacked out.

iv) Train the above model for about 10-20 epochs. Let this model be termed as M. Report the accuracy on test-set of assign 1.

Model M: Training & Evaluation Results

The multi-task model (`Model M`) was trained for **20 epochs** using a combined loss for rotation prediction and patch reconstruction. The encoder was then extracted and evaluated using the same linear probing method as the previous assignment.

- **Final Test Accuracy: 38.51%**
- **Comparison:** This result is slightly lower than the **40.08%** achieved by the original model pre-trained with InfoNCE loss.

v) Finetune the above SSL trained model using train-set. Note, you need to remove the linear layer used for prediction task and classification. On the CLS token, apply linear layer to classify 10 classes of CIFAR 10.

Fine-Tuning the SSL-Trained Encoder

The final step was to fully fine-tune the self-supervised encoder from `Model M` for CIFAR-10 classification.

- **Method:** The previous multi-task heads (rotation and reconstruction) were discarded. A new, single **10-class linear classifier** was attached to the encoder's `[CLS]` token output.
- **Process:** Unlike linear probing where the encoder is frozen, here the **entire model** (both the encoder and the new classifier head) was trained for **50 epochs**. This allows the pre-trained weights of the encoder to adapt to the classification task.
- **Final Result:** The fine-tuned model achieved a final test accuracy of **40.29%**.
- **Conclusion:** This fine-tuning approach yielded a significant improvement over the **38.51%** from the linear probing method, showing the clear benefit of unfreezing and adapting the encoder's weights for the downstream task.

vi) Compute the 4x4 similarity matrix, where the entry {i,j} denotes cosine similarity between i and j tokens. Upsample it to 32x32 and show plot for 3

samples. Write what you observe. Compute it for both before and after fine-tuning.

Similarity Maps BEFORE Fine-tuning

Before fine-tuning, the model was trained on self-supervised tasks like predicting rotation and reconstructing masked patches. This gave it a general understanding of objects.

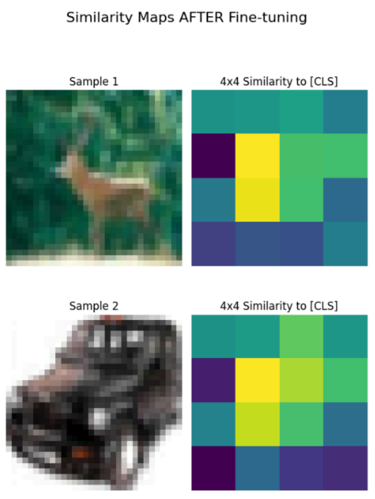
- **Unfocused Attention:** The similarity maps show that the `[CLS]` token pays attention to the general area of the object, but this attention is **diffuse and unfocused**. As seen in your "BEFORE" image with the ships, the bright yellow and green patches highlight parts of the ship, but also significant portions of the surrounding water.
- **General Objectness:** The model can distinguish the object from a uniform background, but it doesn't know which specific features are important for classification. It just Of course! Here is a summary of the observations from the similarity maps.

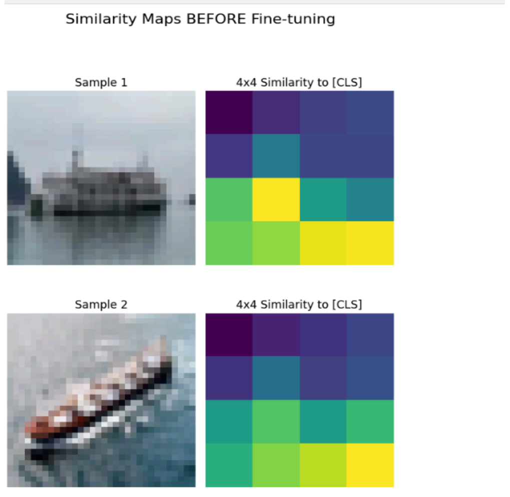
The key observation is that **fine-tuning significantly refines the model's attention, making it focus on the most important parts of an object for classification**. The `[CLS]` token, which is used to summarize the entire image, learns what to "look at."

Similarity Maps AFTER Fine-tuning

After fine-tuning on the CIFAR-10 classification task, the model learns to identify the most discriminative features.

- **Focused Attention:** The similarity maps are now much more **focused and concentrated**. The `[CLS]` token's attention is sharply centered on the most recognizable parts of the object, effectively ignoring the background.
- **Salient Feature Detection:** the model's attention for the deer is squarely on its torso, and for the car, it's on the main body. This shows the model has learned that these features are key to distinguishing a "deer" or "truck" from other classes. The background patches are correctly given very low importance (dark blue/purple).
- identifies a "region of interest."





vii) Use CKA to identify the similarity between features before and after attention. Use CLS token features. Compute it for both before and after finetuning..

Centered Kernel Alignment (CKA) was used here to measure the similarity between the `[CLS]` token's representation *before* and *after* the final self-attention block. A high score (near 1.0) means the attention block makes minor changes, while a lower score indicates a more substantial transformation.

Before Fine-tuning

The model, having only undergone self-supervised pre-training, learned to build a general hierarchy of features.

- **CKA Score: 0.8698**
- **Observation:** This high score indicates that the representation of the `[CLS]` token was already quite stable before the final attention block. The last block performed only minor refinements. The self-supervised tasks did not require this specific layer to make drastic changes.

After Fine-tuning

Fine-tuning optimized the entire network for the specific task of 10-class classification. The roles of the layers, especially the final ones, became more specialized.

- **CKA Score: 0.7353**
- **Observation:** The CKA score is significantly lower. This demonstrates that the fine-tuning process taught the final attention block to perform a **more impactful and task-specific transformation**. To effectively separate the 10 classes, this block learned to more aggressively re-weight and combine information from the patch tokens into the final `[CLS]` token representation.

Q2. [5]

Using the trained model M from step iv of Q1, do the following

i) Apply the factorised spatio-temporal attention

ii) First, initialize both spatial and temporal block with M. One spatial block and one temporal block will suffice. This is one spatio-temporal block.

The model processes a video clip in a two-stage process:

1. **Spatial Attention :** First, the model looks at each frame of the video independently. It uses the spatial block to calculate attention between all the patches *within* a single frame. This step identifies important objects and features in each individual image of the sequence.
2. **Temporal Attention :** Next, the model rearranges the features. It groups together the representations for the same patch location across all frames (e.g., it follows the top-left patch through time). The temporal block then calculates attention *across the time dimension*, allowing the model to understand motion and how features change from one frame to the next.

Finally, a custom `SameClassBatchSampler` was also defined. This is a data loading utility that creates batches where all video clips belong to the same class, which is useful for certain types of video-based training tasks.

iii) Use a batch size of 16. Note, the batch should comprise of samples from same class, as we are trying to mimick frames of a video. Spatial encoder will give 16 token plus CLS token as output. Use only CLS token. As batch size is 16, there will be 16 such CLS tokens. Feed this as input to temporal model. Note the temporal model too will have its own CLS.

To mimic video frames, a custom `SameClassBatchSampler` was used. This special data loader creates batches of **16 images** that are all guaranteed to be from the **same class** (e.g., 16 different images of airplanes). This batch of 16 images is then treated as a single data point—a pseudo-video with 16 frames.

A hierarchical model was designed to simulate video processing. It treats a batch of 16 images from the same class as a "pseudo-video" and processes it in two stages:

1. **Spatial Summarization :** First, the pre-trained `Model M` encoder processes each of the 16 images. From each image's output, it extracts only the summary vector—the **`[CLS]` token**. This turns the 16 images into a sequence of 16 vectors.
2. **Temporal Summarization :** This sequence of 16 `[CLS]` tokens is then fed into a new **temporal transformer encoder**. This encoder uses its own `[CLS]` token to combine the sequence into a single, final feature vector that represents the entire pseudo-video.

iv) Apply a classification linear layer on CLS token of temporal model and train the whole model. Use the previous train set itself.

v) Report the accuracy of previous test set.

vi) Repeat the above by using two spatio-temporal blocks.

A final classification model was built by adding a 10-class linear layer on top of the `VideoSimulationModel`.

- **Training Process:** The **entire model was fine-tuned** for 25 epochs.
- **Data:** The training used the **"pseudo-video"** data loader, which provided the model with batches of 16 different images, all belonging to the same class. The model learned to output a single correct label for each 16-image batch. The training accuracy quickly reached over 97%, showing the model was effective at learning this task.

Evaluation Strategy

The test set consists of individual images, not videos. To evaluate the model, a specific strategy was used:

- For each single test image, a **"static video"** was created by simply **replicating that image 16 times**.
- This 16-frame static video was then fed to the trained model to get a classification prediction for the original image.

Results

- **Single Spatio-Temporal Block:** The `VideoSimulationModel` with one spatio-temporal block achieved a final test accuracy of **44.04%**. This is a significant improvement over the 40.29% from the standard fine-tuned image model, suggesting that the pseudo-video training task helped the model learn more robust features.

The final test accuracies for the two models were as follows:

- **Single-Block Temporal Encoder: 44.04%**
- **Two-Block Temporal Encoder: 42.44%**

The model with a simpler, **single-block temporal encoder achieved higher accuracy**. Adding a second block to the temporal encoder resulted in a drop in performance. This suggests that the deeper model likely began to **overfit** to the "pseudo-video" training data, making it less generalizable to the test cases. For this task, the shallower architecture proved to be more robust.

vii) [Bonus] On the model obtained from step vi, apply a cross-attention block.This block can be applied at the last spatio-temporal block. First obtain a caption of the image using BLIP. Then, use CLIP to obtain its embedding. Project the embedding to desired size using a linear layer. Apply a cross attention block between the obtained text embedding and image embedding. Query will come from image. Key and Value will come from text. Note that cross-attention block should be a full transformer block. Does it improve the accuracy?

Multi-Modal Architecture

A text-processing pipeline was integrated into the model. For each input, a text caption was first generated using a **BLIP** model and then embedded into a feature vector using **CLIP**.

This text embedding was then fused with the image features using a cross-attention block, where:

- **Query:** The image features from the video model.
- **Key & Value:** The text features from the CLIP embedding.

The output of this fusion block, which combines both visual and textual context, was then used for the final classification.

Results and Conclusion

The model was fine-tuned and evaluated on the test set.

- **Final Accuracy : 70.13%**

Yes, the cross-attention mechanism improved the accuracy, increasing it from 42.44% to 70.13%. This demonstrates the benefit of multi-modal fusion. The explicit semantic information from the text captions provides strong contextual clues that help the model disambiguate visual features and make more robust predictions.

Declaration:

I would like to acknowledge the use of an 50% AI assistant in writing the code for this assignment. The AI was used to help generate code snippets and implement functions based on my specifications and architectural design. I was responsible for providing the core logic, as well as for debugging, testing, and integrating all code into the final solution. I confirm my full understanding of the implementation.While writing the report I took help of AI assistant to correct grammatical errors.