GENERAL QUESTIONS

1. What are activation functions and why are they needed?
2. What are underfitting and overfitting, why do they occur, and how can we address them?
3. Why is a validation dataset necessary?
4. How does gradient descent work?
5. How does batch size affect training and inference, and how should we choose it?
6. How can we accelerate SGD convergence?
7. Why use convolutional layers instead of fully connected layers for images?
8. Why add residual connections?
9. Why did we transition from RNNs to Transformers?
10. What is the attention mechanism?
11. Which loss function is commonly used for classification, and why not MSE?
12. What is the difference between MSE and MAE, and when would you use each?

QUESTIONS ABOUT THE PAPER

1. **Level**: Easy

   Question: Why don't we train only an image embedding model when text embedding models like BERT already exist?

   Answer: We need a joint model so we can compare embeddings across modalities.
2. **Level**: Intermediate

   Question: How can we use ALIGN at inference time? What inputs does it take, and what outputs does it produce?

Answer: ALIGN supports text-to-text, text-to-image, image-to-text, image-to-image and combined text + image retrieval.

3. **Level**: Intermediate

   Question: How do you train a model like ALIGN? What are the inputs and what do we optimize?

   Answer: We train on text–image pairs using a text encoder and a visual encoder, optimizing for high similarity on matching pairs and low similarity on all non-matching combinations.
4. **Level**: Hard

   Question: What does the contrastive loss do and what does it optimize in this setting?

   Answer: It maximizes cosine similarity for each matching text–image pair while minimizing similarity between each text and all other images, and each image and all other texts. There is no text–text or image–image optimization.
5. **Level**: Hard

   Question: What is the role of the temperature parameter in the contrastive loss?

   Answer: It scales cosine similarities (which lie between –1 and 1), ensuring the cross-entropy over those similarities remains well-behaved.
6. **Level**: Easy

   Question: How does ALIGN's approach differ from CLIP's?

   Answer: ALIGN uses a much larger, unfiltered dataset that includes noisy examples.
7. **Level**: Intermediate

   Question: What tasks can this type of model perform?

   Answer: Retrieval, zero-shot classification, and serving as a pre-trained encoder.
8. **Level**: Intermediate

   Question: How can we use ALIGN for zero-shot image classification?

   Answer: Encode each candidate label with the text encoder, encode the image with the visual encoder, then select the label whose embedding is closest to the image embedding.

9. **Level**: Intermediate

   Question: Why use two contrastive losses during training?

   Answer: One loss treats each text–image pair as positive against all other images as negatives; the other treats the same pair as positive against all other texts.

10. **Level**: Hard

    Question: Why use a very large batch size during training?

    Answer: To leverage hardware efficiently and include hard negatives in each batch, which improves contrastive learning.

11. **Level**: Hard

    Question: When training on multiple TPUs, do we compute loss and gradients per TPU and average them, or compute a single loss across all TPUs?

    Answer: A single global loss is computed across all TPUs so that all examples contribute hard negatives before backpropagation.

12. **Level**: Intermediate

    Question: The convolutional backbone is an EfficientNet with global pooling trained on 289 × 289 images. Can we use different image sizes?

    Answer: Yes. Global pooling makes the embedding size independent of spatial dimensions, though very different sizes may degrade performance.

13. **Level**: Intermediate

    Question: Is ALIGN a good choice for text-only search?

    Answer: No. It's optimized for cross-modal retrieval, lacks a text–text loss, and uses only 64 input tokens. A dedicated text model performs better.

14. **Level**: Easy

    Question: The paper describes searching with both text and image (e.g. "pandas" image + "red" text retrieves red pandas image). How does this work?

    Answer: At the embedding level: compute the image embedding, add or subtract the text embedding to form a new query embedding, then retrieve the nearest image.