Data 188 Introduction to Deep Learning
Spring 2026 Eric Kim

Discussion 11

This discussion covers masked autoencoders and recommendation systems.

1. Supervised vs. Self-Supervised Learning

(a) As discussed in this class, what is the difference between a supervised learning task vs. a self-supervised learning task?

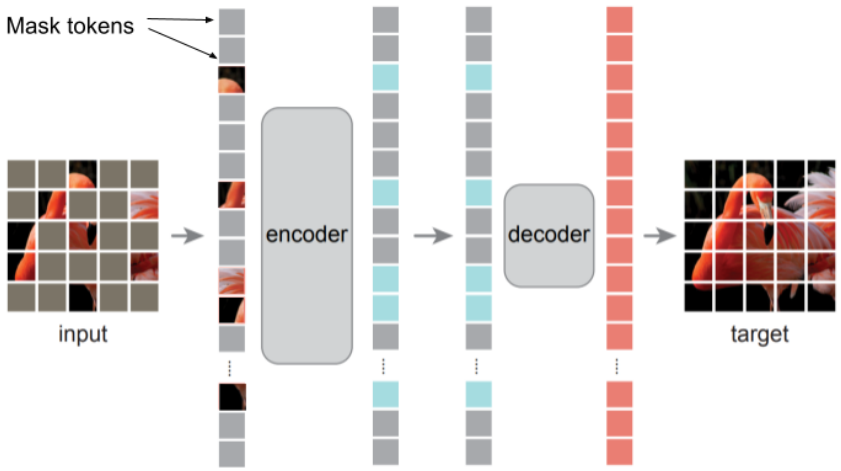
Give an example of a supervised learning task and an example of a self-supervised learning task for both the computer vision and NLP domain.

(b) Suppose we had an audio dataset consisting of millions of songs from a diverse range of genres and artists. Describe a self-supervised learning task to learn a strong audio-clip embedding representation using an audio encoder model (say, an audio-clip embedding of duration 5 seconds).

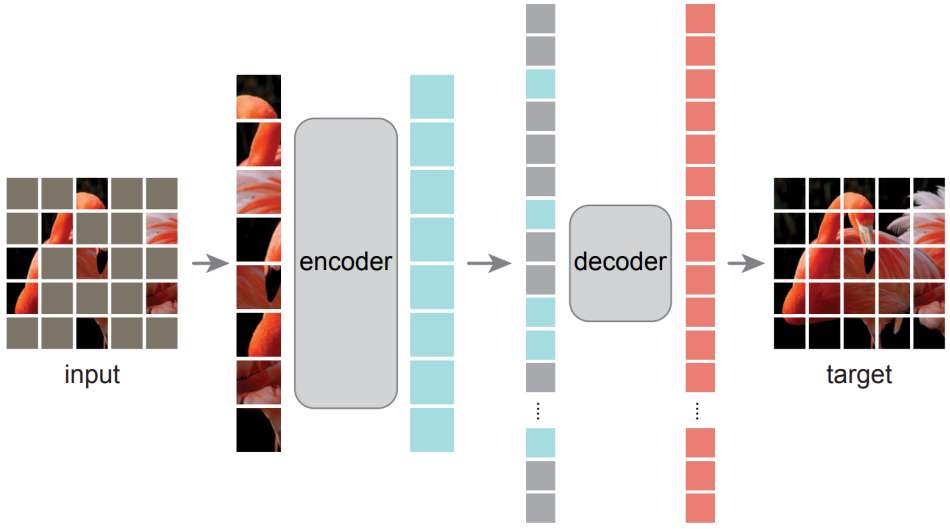
2. Encoder and Mask Tokens

In the masked autoencoder (MAE) architecture, recall that we employ the "fill in the image patch" self-supervised task, using a transformer encoder-decoder as our autoencoder.

Consider the following realization of this idea, where we pass all image patches to the encoder (i.e., both pink and gray boxes):



Compare it to the following approach that the MAE paper actually took:



Why do you think the authors chose to omit the masked image patches to the encoder? Recall that the authors found that an aggressive masking percentage (masking 75% of the image) performed well.

3. Distributed Training

It's rumored that GPT-4 has on the order of 1.8 trillion model parameters. If we stored them as the float16 data type (2 bytes per parameter), this would require 3600 GB GPU memory.

For comparison, in 2026 one of the most powerful GPU servers that AWS EC2 offers is the p5en.48xlarge, that (with 8 H200 GPU's) has a combined total of 1128 GB GPU memory, and costs around $554K per year to rent. Suppose we don't have access to these machines, and we have to instead settle for machines that only have 384 GB GPU memory (ex: g6e.48xlarge).

Which distributed training technique would be most appropriate for us to train a GPT-4 model: Distributed Data Parallel (DDP), or Fully Sharded Data Parallel (FSDP)?

4. Two-Stage Recommendation System

Recall that, during lecture, we covered a two-stage recommendation system architecture, where we first have a lightweight retrieval component ("candidate generation"), followed by a heavier-duty ranking component that ultimately produces the final results.

Why do we need this two-stage architecture? Suppose we discarded the first stage ("candidate generation"), and used my ranking model to rerank my entire corpus. Why is this usually a bad idea? When might this be OK?

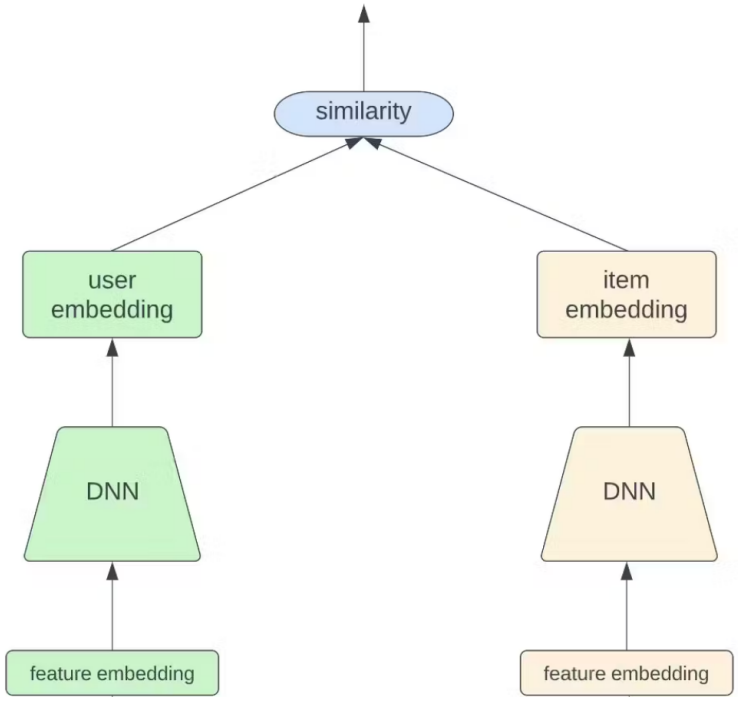
5. Two Tower Models

Suppose we manage a photosharing app where users engage with photos (like, comment, share) that other users have uploaded.

Assume that the app has been running long enough (and with enough users) that we have plentiful user engagement data.

We would like to train a model that, given a User and a query image, outputs the probability that the User will like the image. We want our User and Item embeddings to be 64-dimensional. Use the cosine similarity function as our similarity function.

Describe how to build a two-tower model that achieves this. Be specific about the model architecture details. How would you generate the training dataset? Use this diagram from lecture as reference:



Assume we have the following User features: `int age`, `int account_age`, `string country`, `string gender`, `string favorite_photo_category`.

Contributors:

- Eric Kim, Andria Xu.

