Deep Learning: ECE-7123

Homework 3

Saad Zubairi
shz2020

May 2, 2025

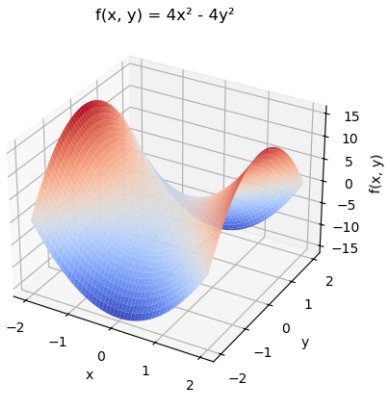
Contents

1 Problem 1: Minimax optimization

Given the function:

$$\min_x \max_y f(x, y) = 4x^2 - 4y^2.$$

Which can be plotted as:



a) Saddle point

To find the saddle point, we can take the partial derivatives with respect to x and y:

$$\nabla f(x, y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right) = (8x, -8y)$$

Setting $\nabla f(x, y) = 0$, we solve:

$$8x = 0 \quad \text{and} \quad -8y = 0$$

This gives the saddle point at $(x, y) = (0, 0)$.

b) Gradient descent/ascent update rules

To derive the update rules for gradient descent in x and gradient ascent in y, we start with the function $f(x, y) = 4x^2 - 4y^2$. The gradient of $f(x, y)$ is given by:

$$\nabla f(x, y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right) = (8x, -8y).$$

For gradient descent in x, we update x by moving in the direction opposite to the gradient:

$$x_{t+1} = x_t - \eta \frac{\partial f}{\partial x}.$$

Substituting $\frac{\partial f}{\partial x} = 8x_t$, we get:

$$x_{t+1} = x_t - \eta(8x_t) = x_t(1 - 8\eta).$$

For gradient ascent in y, we update y by moving in the direction of the gradient:

$$y_{t+1} = y_t + \eta \frac{\partial f}{\partial y}.$$

Substituting $\frac{\partial f}{\partial y} = -8y_t$, we get:

$$y_{t+1} = y_t + \eta(-8y_t) = y_t(1 + 8\eta).$$

Thus, the update rules are:

$$x_{t+1} = (1 - 8\eta)x_t, \quad y_{t+1} = (1 + 8\eta)y_t.$$

c) Convergence of the update rules

To ensure convergence, we analyze the update rules:

$$x_{t+1} = (1 - 8\eta)x_t, \quad y_{t+1} = (1 + 8\eta)y_t.$$

For convergence of x_t in gradient descent, the factor $(1 - 8\eta)$ must satisfy:

$$|1 - 8\eta| < 1.$$

Solving this inequality:

$$-1 < 1 - 8\eta < 1.$$

Subtracting 1 from all sides:

$$-2 < -8\eta < 0.$$

Dividing through by -8 (reversing the inequality):

$$0 < \eta < \frac{1}{4}.$$

For y_t in gradient ascent, the factor $(1 + 8\eta)$ must satisfy:

$$|1 + 8\eta| < 1.$$

Solving this inequality:

$$-1 < 1 + 8\eta < 1.$$

Subtracting 1 from all sides:

$$-2 < 8\eta < 0.$$

Dividing through by 8:

$$-\frac{1}{4} < \eta < 0.$$

Since η must be positive for gradient descent and ascent to make sense, we combine the results:

$$0 < \eta < \frac{1}{4}.$$

Thus, the range of allowable step sizes is:

$$\boxed{0 < \eta < \frac{1}{4}}$$

d) Regular gradient descent on both variables

To analyze regular gradient descent on both variables x and y, we update both using the gradient of $f(x, y)$. The gradient is:

$$\nabla f(x, y) = (8x, -8y).$$

The update rules for gradient descent on x and y are:

$$x_{t+1} = x_t - \eta \frac{\partial f}{\partial x}, \quad y_{t+1} = y_t - \eta \frac{\partial f}{\partial y}.$$

Substituting the partial derivatives:

$$x_{t+1} = x_t - \eta(8x_t) = x_t(1 - 8\eta),$$
$$y_{t+1} = y_t - \eta(-8y_t) = y_t(1 + 8\eta).$$

Thus, the update rules are:

$$x_{t+1} = (1 - 8\eta)x_t, \quad y_{t+1} = (1 + 8\eta)y_t.$$

Dynamics of the updates

- For x_t, the factor $(1 - 8\eta)$ determines the convergence. If $0 < \eta < \frac{1}{4}$, then $|1 - 8\eta| < 1$, ensuring that x_t decays to 0.
- For y_t, the factor $(1 + 8\eta)$ determines the behavior. Since $(1 + 8\eta) > 1$ for $\eta > 0$, y_t grows exponentially, leading to divergence.

2 Vision-Language models

Overview

In this experiment, we evaluate three different combinations of image and text encoders for CLIP-style contrastive learning. The goal is to assess how different architectural choices affect the model's ability to align images and captions.

Model Configurations

Combination 1: ResNet-50 + DistilBERT

- Image Encoder: `resnet50` (output dim: 2048)
- Text Encoder: `distilbert-base-uncased` (output dim: 768)
- Projection Dim: 256

Combination 2: ResNet-18 + BERT

- Image Encoder: `resnet18` (output dim: 512)
- Text Encoder: `bert-base-uncased` (output dim: 768)
- Projection Dim: 256

Combination 3: ResNet-34 + RoBERTa

- Image Encoder: `resnet34` (output dim: 512)
- Text Encoder: `roberta-base` (output dim: 768)
- Projection Dim: 256

a) Rationale

The encoder combinations were selected semi-randomly to reflect varying model scales. The purpose is to observe whether larger encoders (e.g., ResNet-50, RoBERTa) yield visibly better alignment without tuning hyperparameters.

b) Implementation Changes

Config Changes

All modifications are made in the `config` class.

```
model_name = 'resnet50'
image_embedding = 2048
text_encoder_model = 'roberta-base'
text_tokenizer = 'roberta-base'
text_embedding = 768
projection_dim = 512
```

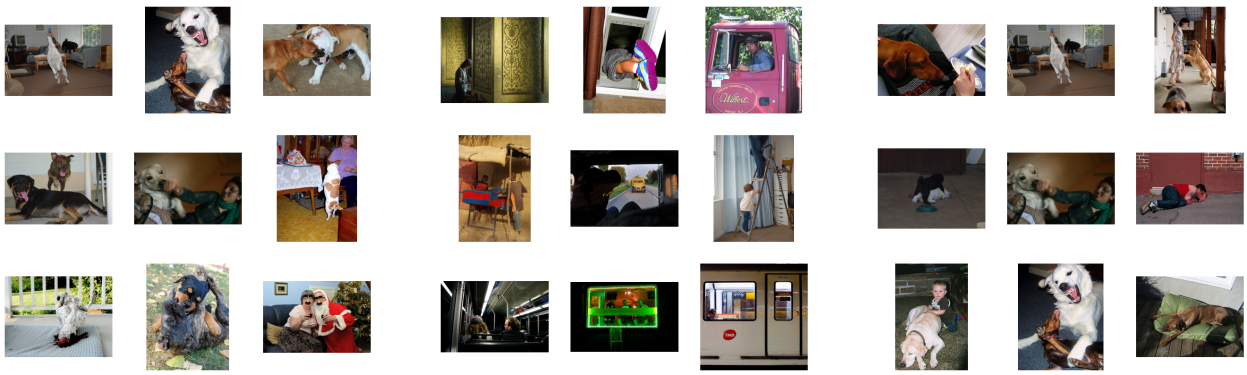
Encoder Changes

In addition to updating the `config` class, the encoders were modified throughout the implementation to support RoBERTa. This included:

- Updating the forward pass to handle RoBERTa's tokenization and embedding outputs.
- Adjusting the projection layers to match the output dimensions of RoBERTa (768) and the specified projection dimension (256).
- Updating the `get_image_embeddings` function to use `AutoTokenizer` for tokenization; compaitble with the updated `text_tokenizer`.

c) Inference Examples

i. Prompt: "A cat sitting with a human"



(a) ResNet-50 + DistilBERT (b) ResNet-18 + BERT (c) ResNet-34 + RoBERTa

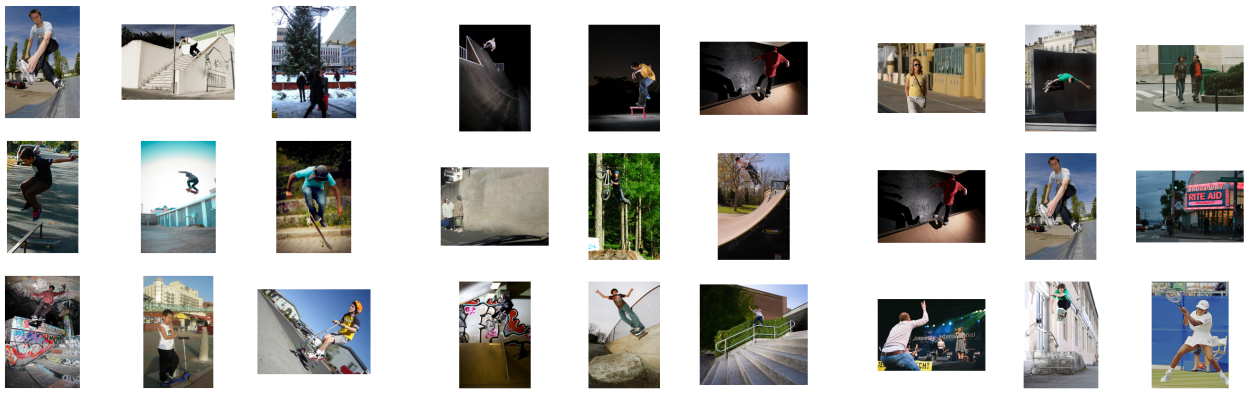
ii. Prompt: "A crowd gathered in a park"

(a) ResNet-50 + DistilBERT (b) ResNet-18 + BERT (c) ResNet-34 + RoBERTa

iii. Prompt: "A dog sitting alone in a park"

(a) ResNet-50 + DistilBERT (b) ResNet-18 + BERT (c) ResNet-34 + RoBERTa

iv. Prompt: "A man skateboarding in the city"



(a) ResNet-50 + DistilBERT (b) ResNet-18 + BERT (c) ResNet-34 + RoBERTa

v. Prompt: "All kinds of food on a table"

(a) ResNet-50 + DistilBERT (b) ResNet-18 + BERT (c) ResNet-34 + RoBERTa

d) Evaluation

i. Prompt: "A cat sitting with a human"

This prompt revealed major limitations across all models.

- **Model A:** Returned dog images—incorrect, but at least animal-adjacent.
- **Model B:** Performed the worst: no animals, minimal context alignment, and often returned scenes with no semantic relevance.
- **Model C:** Came up with a range of unrelated content, including vehicles and humans, with no cats in sight.

Observation: The models seem to generalize poorly to cat-related queries, possibly due to underrepresentation in the dataset or visual confusion with dogs.

ii. Prompt: "A crowd gathered in a park"

All models demonstrated some degree of success.

- **Model A:** Retrieved crowd scenes, several of which included greenery and open spaces resembling parks.
- **Model B:** Managed decent crowd representations, albeit with minimal park-like settings.
- **Model C:** Captured the crowd element well but failed to recognize the park context, retrieving urban or event scenes instead.

Observation: Crowd detection appears robust, but fine-grained location understanding (e.g., park vs street) varies by model.

iii. Prompt: "A dog sitting alone in a park"

Performance improved noticeably here.

- **Model A:** The most consistent in retrieving visually relevant scenes.
- **Model B:** Adequate but unremarkable.
- **Model C:** Despite earlier underperformance, correctly surfaced a few images matching the "sitting dog" description.

Observation: The models exhibit strong prior alignment for the concept of "dog + park," likely due to frequent co-occurrence in training data.

iv. Prompt: "A man skateboarding in the city"

A clear separation in model quality emerged.

- **Model A:** Provided highly accurate results, with multiple images of urban skateboarding scenes.

- **Model B:** More ambiguous—some city scenes, limited motion context.
- **Model C:** Performed poorly, retrieving unrelated activities (including what appeared to be a tennis match).

Observation: Motion-related prompts require both object and action recognition; only Model A appears capable of handling this dual requirement reliably.

v. Prompt: "All kinds of food on a table"

Mixed results with some surprises.

- **Model A:** Showed multiple dining scenes, often including food, people at tables, or floral arrangements. Semantically close, though not perfect.
- **Model B:** Failed entirely, presenting irrelevant images such as trucks and individuals using phones.
- **Model C:** Returned vague or partial matches—some table elements, few actual food visuals.

Observation: Abstract or multi-object prompts (e.g., "all kinds of food") seem more challenging for the smaller models. Model A again showed the strongest contextual alignment.

e) Final Observations

- Model A (ResNet-50 + DistilBERT): This model consistently outperformed the others in both scene and object understanding. It handled composite prompts well and recovered semantically aligned images even when fine-grained details were missing. The larger image encoder (ResNet-50) likely contributed to its superior performance, compensating for the relatively smaller text encoder (DistilBERT).
- Model B (ResNet-18 + BERT): This model lagged significantly behind the others. While BERT is more powerful than DistilBERT, the weak image encoder (ResNet-18) likely limited the overall performance. The smaller image encoder struggled to extract sufficient visual features, leading to poor alignment with text embeddings.
- Model C (ResNet-34 + RoBERTa): This model was inconsistent—sometimes retrieving promising matches (especially for prompts involving dogs and parks), but often drifting semantically. The larger text encoder (RoBERTa) provided strong language representations, but the mid-sized image encoder (ResNet-34) may have been insufficient to fully leverage this capability, resulting in unstable grounding between image and text modalities.

f) Impact of Scaling

Scaling plays a critical role in the performance of vision-language models. Larger image encoders, such as ResNet-50, provide richer visual feature representations, which are crucial for aligning with text embeddings. Similarly, more powerful text encoders, such as RoBERTa, offer better semantic understanding of textual prompts. However, the benefits of scaling are not uniform across modalities—imbalances between the capacities of the image and text encoders can lead to suboptimal performance. For example, Model C's strong text encoder was not fully utilized due to its mid-sized image encoder.

In general, scaling both the image and text encoders proportionally tends to yield the best results, as seen with Model A. However, this comes at the cost of increased computational requirements, which must be balanced against the desired performance and available resources.

```
    def sexy_func(x):
        """Return squares of x."""
        return [i**2 for i in x]
```