Homework 3

Graded

Student

Rohan Gore

Total Points

15 / 15 pts

Question 1

Min max optimization — **5** / 5 pts

1.1 **Saddle point** — **1** / 1 pt

- ✔ **– 0 pts** Correct
- **– 1 pt** Incorrect
- **– 0.5 pts** Steps not clear

1.2 **GDA** — **1** / 1 pt

- ✔ **– 0 pts** Correct
- **– 0.5 pts** Incomplete answer
- **– 0.5 pts** Minor mistake in signs
- **– 1 pt** Not attempted

1.3 **Step sizes** — **1** / 1 pt

- ✔ **– 0 pts** Correct
- **– 0.5 pts** Wrong sign for gradient ascent
- **– 1 pt** wrong

1.4 **Convergence** — **2** / 2 pts

- ✔ **– 0 pts** Correct
- **– 2 pts** Wrong answer
- **– 1 pt** Wrong convergence/divergence.
- **– 1 pt** Missing special case
- **– 2 pts** Missing Solution
- **– 1 pt** Missing update factor/step size

Question 2

CLIP models **5** / 5 pts

2.1 **Training** **3** / 3 pts

✔ **– 0 pts** Correct

– 1 pt trained only 2 configurations

– 2 pts trained only one configuration

– 3 pts no training code

– 1 pt no training logs for verification on some or all configurations

2.2 **Evals + prompting** **1** / 1 pt

✔ **– 0 pts** Correct

– 1 pt Inference images not shown for any of the models

– 0.5 pts Inference images shown for less than 3 models

– 0.5 pts Inference not shown for same text prompt across different models

– 0.5 pts Incomplete

– 1 pt Incorrect

2.3 **Trends** **1** / 1 pt

✔ **– 0 pts** Correct

– 1 pt No trends/observation reported

Question 3

DCGAN **5** / 5 pts

3.1 **Training** **3** / 3 pts

- ✔ **– 0 pts** Correct
- **– 0.5 pts** Incorrect/missing loss curve
- **– 1 pt** Images of epoch 10,30,50 not displayed/ inaccurate
- **– 0.5 pts** Incorrect architecture
- **– 0.5 pts** Missing output
- **– 1 pt** Code not shown
- **– 3 pts** No answer given

3.2 **Inference** **2** / 2 pts

- ✔ **– 0 pts** Correct
- **– 2 pts** Intermediate images missing/inaccurate

Question 4

Late submission **0** / 0 pts

- ✔ **– 0 pts** Submission on time
- **– 1 pt** Late submission

Question assigned to the following page: 1.1

NYU: Deep Learning CS 6953 Spring 2025

Homework 3

Name: Rohan Mahesh Gore (N19332535)

Problem 1: Minimax Optimization

Collaborators: Perplexity - Sonar Huge & DeepSeek-R1 , ChatGPT 4o.

a. Minimax Optimization

We are given a bivariate function $f(x, y) = 4x^2 - 4y^2$ of which we need to find a saddle point, which is basically a point where the function is minimized in one direction (x^2) and maximized in another (y^2).

Step 1: Finding critical points using partial derivatives

Calculating the partial derivatives of $f(x, y)$ with respect to each variable

$$\frac{\partial f}{\partial x} = \frac{d}{dx}(4x^2) = 8x \qquad \frac{\partial f}{\partial y} = \frac{d}{dy}(-4y^2) = -8y$$

Equating them to 0:

$$8x = 0 \Rightarrow x = 0 \qquad -8y = 0 \Rightarrow y = 0$$

So we got a critical point which is:

$$(x^*, y^*) = (0, 0)$$

Step 2: Confirming the saddle point using Hessian matrix

For calcualting this we comppute Hessian matrix which is a matrix of second-order partial derivatives.

This can be calculated as:

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix} = \begin{bmatrix} 8 & 0 \\ 0 & -8 \end{bmatrix}$$

On observation of eigenvalues of the Hessian MatriX, we can see that:

- Positive eigenvalue (8) means convex in that direction (x-direction).
- Negative eigenvalue (8) means concave in that direction (y-direction).

This observation confirms that $(0, 0)$ is a saddle point, because the function is increasing in one direction (x) and decreasing in another (y).

Hence, Solved,

Question assigned to the following page: 1.2

b. Gradient descent/ascent Update Equations

As this is a minimax problem, we need to:
- Minimize the function over one variable (here x) $\rightarrow$ using gradient descent
- Maximize the function over another variable (here y) $\rightarrow$ using gradient ascent

So, we will create two separate update equations for x and y, using the gradients from question(a).

Step 1: Gradient descent on x

$$x_{t+1} \quad = \quad x_t - \eta \cdot \frac{\partial f}{\partial x} \quad = \quad x_t - \eta \cdot 8x_t = x_t(1 - 8\eta)$$

which means, that in every step, x shrinks by a factor of $(1 - 8\eta)$.

It also means that, if η is chosen properly, this will lead to convergence of x_t toward 0.

Step 2: Gradient ascent on y

As we are maximizing the function with respect to y, we move in opposite direction as compared to the traditional gradient descent, by having a "plus" sign in the equation, such as:

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

So as we can see, both x and y update via the same scaling rule.

Therefore the final gradient descent and gradient ascent equations are:

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

Hence, Solved.

Question assigned to the following page: 1.3

c. Determining Range of allowable Step Sizes

As we have calculated the gradient descent and ascent quations, we now need to analyze whether and how do they converge.

As in both the equations, we are multiplying each variable by $(1 - 8\eta)$ at every step, we want it to shrink the values towards zero.

Lets denote this multipler as: $r = 1 - 8\eta$

The variables to converge to 0, which happens if the magnitude of this factor is less than 1. Therefore:

$$|r| = |1 - 8\eta| < 1$$

We need to solve this for exact bounds.

Step 1: Splitting into inequalities

1. $1 - 8\eta < 1 \quad \Rightarrow \quad always true$

2. $1 - 8\eta > -1 \quad \Rightarrow \quad 8\eta < 2 \quad \Rightarrow \quad \eta < \frac{1}{4}$

3. Also, since we want the sequence to actually decrease toward 0, we must avoid negative scaling (which causes oscillation), so we also want:

$$1 - 8\eta > 0 \quad \Rightarrow \quad \eta < \frac{1}{8}$$

4. But as we know, oscillation doesn't prevent convergence — it just makes it alternate signs. So we need to take the final condition into consideration, which is:

$$|1 - 8\eta| < 1 \quad \Rightarrow \quad -1 < 1 - 8\eta < 1$$

Step 2: Solving the compound inequalities

For Lower bound: $1 - 8\eta > -1 \quad \Rightarrow \quad 8\eta < 2 \quad \Rightarrow \quad \eta < \frac{1}{4}$

For Upper bound: $1 - 8\eta < 1 \quad \Rightarrow \quad$ (always true)

Step 3: Combining Conditions

Therefore by combining conditions from above step, we get a range, which is: $0 < \eta < \frac{1}{4}$

Therefore, for the gradient descent and gradient ascent functions to converge to the saddle point, the learning rate must satisfy the condition:

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

Hence, Determined.

Question assigned to the following page: 1.4

d. Demonstration & Comments on an Approach

Lets demo the results in the case if we apply gradient descent to both variables, instead of doing descent-ascent as intended in minimax settings.

Step 1: Gradient descent on x

Similar to the calculations in question(b), we have: $x_{t+1} = x_t(1 - 8\eta)$

Step 2: Gradient descent on y

Here's the different approach which is basically a **mistake**.

WE need to consider descent and hence the sign would be "negative" for the equation. Therefore, we have:

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

Comments:

- Based on both these equations, it means that the magnitude of y will grow exponentially, because the factor $(1 + 8\eta) > 1$ for any $\eta > 0$.
- Therefore, with every update, y increases farther away from 0.
- Hence, the optimization will diverge.

Analyzing Special Case 1: $\eta = 0$

Now, we want both x_t and y_t to converge, so the update factors $|1 - 8\eta|$ & $|1 + 8\eta|$ must both be less than 1.

Therefore we have:

1. $|1 - 8\eta| < 1 \quad \Rightarrow \quad 0 < \eta < \frac{1}{4}$ (η must be positive)
2. $|1 + 8\eta| < 1 \quad \Rightarrow \quad \eta < 0$ (η must be negative)

But, these two conditions contradict each other as η can't be both positive and negative simultaneously.

Therefore, **there does not exist a valid positive value** of η that causes both x_t and y_t to converge simultaneously under simultaneous gradient descent on both variables, which basically leads to no updates.

Analyzing Special Case 2: $y_0 = 0$

In this special case, since the initial value is zero, and each update depends linearly on the previous value, the future values will also be zero:

$$y_1 = y_0(1 + 8\eta) = 0, \quad y_2 = 0, \quad ...$$

But this is very fragile as, even a tiny **noise will push it away** from 0, and then it will start diverging again.

Hence, applying regular gradient descent to both x and y leads to divergent behavior, where **x** converges but **y** would keep diverging further.

Questions assigned to the following page: 1.4 and 3.1

We may only converge to the saddle point in special situations (e.g., $y_0 = 0$), but this is not generalizable.

Hence, Demonstrated.

Problem 2: Vision-Language models

Collaborators: Gemini 2.5 Flash & Perplexity - Claude 3.7 Sonnet

Attached ahead.

Problem 3: GANS

Collaborators: Gemini 2.5 Flash & Perplexity - Claude 3.7 Sonnet

Attached ahead.

No questions assigned to the following page.

Question 2 - Vision-Language models

May 4, 2025

1 *Question 2 - Vision-Language models*

2

[1]:
```
!pip install timm
!pip install transformers
!pip install opencv-python
!pip install pandas
!pip install albumentations
!pip install matplotlib
!pip install kaggle --upgrade1
!pip install --upgrade kaggle tqdm
```

```
Requirement already satisfied: timm in /usr/local/lib/python3.11/dist-packages
(1.0.14)
Requirement already satisfied: torch in /usr/local/lib/python3.11/dist-packages
(from timm) (2.5.1+cu124)
Requirement already satisfied: torchvision in /usr/local/lib/python3.11/dist-
packages (from timm) (0.20.1+cu124)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.11/dist-packages
(from timm) (6.0.2)
Requirement already satisfied: huggingface_hub in
/usr/local/lib/python3.11/dist-packages (from timm) (0.30.2)
Requirement already satisfied: safetensors in /usr/local/lib/python3.11/dist-
packages (from timm) (0.5.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-
packages (from huggingface_hub->timm) (3.18.0)
Requirement already satisfied: fsspec>=2023.5.0 in
/usr/local/lib/python3.11/dist-packages (from huggingface_hub->timm) (2025.3.2)
Requirement already satisfied: packaging>=20.9 in
/usr/local/lib/python3.11/dist-packages (from huggingface_hub->timm) (24.2)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-
packages (from huggingface_hub->timm) (2.32.3)
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.11/dist-
packages (from huggingface_hub->timm) (4.67.1)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.11/dist-packages (from huggingface_hub->timm) (4.13.1)
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-
```

No questions assigned to the following page.

```
packages (from torch->timm) (3.4.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages
(from torch->timm) (3.1.6)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.4.127 in
/usr/local/lib/python3.11/dist-packages (from torch->timm) (12.4.127)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.4.127 in
/usr/local/lib/python3.11/dist-packages (from torch->timm) (12.4.127)
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.4.127 in
/usr/local/lib/python3.11/dist-packages (from torch->timm) (12.4.127)
Collecting nvidia-cudnn-cu12==9.1.0.70 (from torch->timm)
  Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-
manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cublas-cu12==12.4.5.8 (from torch->timm)
  Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cufft-cu12==11.2.1.3 (from torch->timm)
  Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-curand-cu12==10.3.5.147 (from torch->timm)
  Downloading nvidia_curand_cu12-10.3.5.147-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cusolver-cu12==11.6.1.9 (from torch->timm)
  Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-
manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cusparse-cu12==12.3.1.170 (from torch->timm)
  Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-
manylinux2014_x86_64.whl.metadata (1.6 kB)
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in
/usr/local/lib/python3.11/dist-packages (from torch->timm) (2.21.5)
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in
/usr/local/lib/python3.11/dist-packages (from torch->timm) (12.4.127)
Collecting nvidia-nvjitlink-cu12==12.4.127 (from torch->timm)
  Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Requirement already satisfied: triton==3.1.0 in /usr/local/lib/python3.11/dist-
packages (from torch->timm) (3.1.0)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.11/dist-
packages (from torch->timm) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.11/dist-packages (from sympy==1.13.1->torch->timm)
(1.3.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages
(from torchvision->timm) (1.26.4)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in
/usr/local/lib/python3.11/dist-packages (from torchvision->timm) (11.1.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.11/dist-packages (from jinja2->torch->timm) (3.0.2)
Requirement already satisfied: mkl_fft in /usr/local/lib/python3.11/dist-
```

No questions assigned to the following page.

```
packages (from numpy->torchvision->timm) (1.3.8)
Requirement already satisfied: mkl_random in /usr/local/lib/python3.11/dist-
packages (from numpy->torchvision->timm) (1.2.4)
Requirement already satisfied: mkl_umath in /usr/local/lib/python3.11/dist-
packages (from numpy->torchvision->timm) (0.1.1)
Requirement already satisfied: mkl in /usr/local/lib/python3.11/dist-packages
(from numpy->torchvision->timm) (2025.1.0)
Requirement already satisfied: tbb4py in /usr/local/lib/python3.11/dist-packages
(from numpy->torchvision->timm) (2022.1.0)
Requirement already satisfied: mkl-service in /usr/local/lib/python3.11/dist-
packages (from numpy->torchvision->timm) (2.4.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests->huggingface_hub->timm)
(3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-
packages (from requests->huggingface_hub->timm) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests->huggingface_hub->timm)
(2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from requests->huggingface_hub->timm)
(2025.1.31)
Requirement already satisfied: intel-openmp<2026,>=2024 in
/usr/local/lib/python3.11/dist-packages (from mkl->numpy->torchvision->timm)
(2024.2.0)
Requirement already satisfied: tbb==2022.* in /usr/local/lib/python3.11/dist-
packages (from mkl->numpy->torchvision->timm) (2022.1.0)
Requirement already satisfied: tcmlib==1.* in /usr/local/lib/python3.11/dist-
packages (from tbb==2022.*->mkl->numpy->torchvision->timm) (1.2.0)
Requirement already satisfied: intel-cmplr-lib-rt in
/usr/local/lib/python3.11/dist-packages (from
mkl_umath->numpy->torchvision->timm) (2024.2.0)
Requirement already satisfied: intel-cmplr-lib-ur==2024.2.0 in
/usr/local/lib/python3.11/dist-packages (from intel-
openmp<2026,>=2024->mkl->numpy->torchvision->timm) (2024.2.0)
Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl (363.4
MB)
                         363.4/363.4 MB
4.7 MB/s eta 0:00:000:00:0100:01
Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl
(664.8 MB)
                         664.8/664.8 MB
2.5 MB/s eta 0:00:000:00:0100:01
Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl
(211.5 MB)
                         211.5/211.5 MB
8.0 MB/s eta 0:00:000:00:0100:01
Downloading nvidia_curand_cu12-10.3.5.147-py3-none-
```

No questions assigned to the following page.

```
manylinux2014_x86_64.whl (56.3 MB)
                         56.3/56.3 MB
30.9 MB/s eta 0:00:00:00:0100:01
Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-
manylinux2014_x86_64.whl (127.9 MB)
                         127.9/127.9 MB
8.5 MB/s eta 0:00:000:00:0100:01
Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-
manylinux2014_x86_64.whl (207.5 MB)
                         207.5/207.5 MB
8.3 MB/s eta 0:00:000:00:0100:01
Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl (21.1 MB)
                         21.1/21.1 MB
86.4 MB/s eta 0:00:00:00:0100:01
Installing collected packages: nvidia-nvjitlink-cu12, nvidia-curand-cu12,
nvidia-cufft-cu12, nvidia-cublas-cu12, nvidia-cusparse-cu12, nvidia-cudnn-cu12,
nvidia-cusolver-cu12
  Attempting uninstall: nvidia-nvjitlink-cu12
    Found existing installation: nvidia-nvjitlink-cu12 12.8.93
    Uninstalling nvidia-nvjitlink-cu12-12.8.93:
      Successfully uninstalled nvidia-nvjitlink-cu12-12.8.93
  Attempting uninstall: nvidia-curand-cu12
    Found existing installation: nvidia-curand-cu12 10.3.9.90
    Uninstalling nvidia-curand-cu12-10.3.9.90:
      Successfully uninstalled nvidia-curand-cu12-10.3.9.90
  Attempting uninstall: nvidia-cufft-cu12
    Found existing installation: nvidia-cufft-cu12 11.3.3.83
    Uninstalling nvidia-cufft-cu12-11.3.3.83:
      Successfully uninstalled nvidia-cufft-cu12-11.3.3.83
  Attempting uninstall: nvidia-cublas-cu12
    Found existing installation: nvidia-cublas-cu12 12.8.4.1
    Uninstalling nvidia-cublas-cu12-12.8.4.1:
      Successfully uninstalled nvidia-cublas-cu12-12.8.4.1
  Attempting uninstall: nvidia-cusparse-cu12
    Found existing installation: nvidia-cusparse-cu12 12.5.8.93
    Uninstalling nvidia-cusparse-cu12-12.5.8.93:
      Successfully uninstalled nvidia-cusparse-cu12-12.5.8.93
  Attempting uninstall: nvidia-cudnn-cu12
    Found existing installation: nvidia-cudnn-cu12 9.3.0.75
    Uninstalling nvidia-cudnn-cu12-9.3.0.75:
      Successfully uninstalled nvidia-cudnn-cu12-9.3.0.75
  Attempting uninstall: nvidia-cusolver-cu12
    Found existing installation: nvidia-cusolver-cu12 11.7.3.90
    Uninstalling nvidia-cusolver-cu12-11.7.3.90:
      Successfully uninstalled nvidia-cusolver-cu12-11.7.3.90
```

No questions assigned to the following page.

```
ERROR: pip's dependency resolver does not currently take into account all
the packages that are installed. This behaviour is the source of the following
dependency conflicts.
pylibcugraph-cu12 24.12.0 requires pylibraft-cu12==24.12.*, but you have
pylibraft-cu12 25.2.0 which is incompatible.
pylibcugraph-cu12 24.12.0 requires rmm-cu12==24.12.*, but you have rmm-cu12
25.2.0 which is incompatible.
Successfully installed nvidia-cublas-cu12-12.4.5.8 nvidia-cudnn-
cu12-9.1.0.70 nvidia-cufft-cu12-11.2.1.3 nvidia-curand-cu12-10.3.5.147 nvidia-
cusolver-cu12-11.6.1.9 nvidia-cusparse-cu12-12.3.1.170 nvidia-nvjitlink-
cu12-12.4.127
Requirement already satisfied: transformers in /usr/local/lib/python3.11/dist-
packages (4.51.1)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-
packages (from transformers) (3.18.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.30.0 in
/usr/local/lib/python3.11/dist-packages (from transformers) (0.30.2)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dist-
packages (from transformers) (1.26.4)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.11/dist-packages (from transformers) (24.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-
packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.11/dist-packages (from transformers) (2024.11.6)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-
packages (from transformers) (2.32.3)
Requirement already satisfied: tokenizers<0.22,>=0.21 in
/usr/local/lib/python3.11/dist-packages (from transformers) (0.21.0)
Requirement already satisfied: safetensors>=0.4.3 in
/usr/local/lib/python3.11/dist-packages (from transformers) (0.5.2)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.11/dist-
packages (from transformers) (4.67.1)
Requirement already satisfied: fsspec>=2023.5.0 in
/usr/local/lib/python3.11/dist-packages (from huggingface-
hub<1.0,>=0.30.0->transformers) (2025.3.2)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.11/dist-packages (from huggingface-
hub<1.0,>=0.30.0->transformers) (4.13.1)
Requirement already satisfied: mkl_fft in /usr/local/lib/python3.11/dist-
packages (from numpy>=1.17->transformers) (1.3.8)
Requirement already satisfied: mkl_random in /usr/local/lib/python3.11/dist-
packages (from numpy>=1.17->transformers) (1.2.4)
Requirement already satisfied: mkl_umath in /usr/local/lib/python3.11/dist-
```

No questions assigned to the following page.

```
packages (from numpy>=1.17->transformers) (0.1.1)
Requirement already satisfied: mkl in /usr/local/lib/python3.11/dist-packages
(from numpy>=1.17->transformers) (2025.1.0)
Requirement already satisfied: tbb4py in /usr/local/lib/python3.11/dist-packages
(from numpy>=1.17->transformers) (2022.1.0)
Requirement already satisfied: mkl-service in /usr/local/lib/python3.11/dist-
packages (from numpy>=1.17->transformers) (2.4.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests->transformers) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-
packages (from requests->transformers) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests->transformers) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from requests->transformers)
(2025.1.31)
Requirement already satisfied: intel-openmp<2026,>=2024 in
/usr/local/lib/python3.11/dist-packages (from mkl->numpy>=1.17->transformers)
(2024.2.0)
Requirement already satisfied: tbb==2022.* in /usr/local/lib/python3.11/dist-
packages (from mkl->numpy>=1.17->transformers) (2022.1.0)
Requirement already satisfied: tcmlib==1.* in /usr/local/lib/python3.11/dist-
packages (from tbb==2022.*->mkl->numpy>=1.17->transformers) (1.2.0)
Requirement already satisfied: intel-cmplr-lib-rt in
/usr/local/lib/python3.11/dist-packages (from
mkl_umath->numpy>=1.17->transformers) (2024.2.0)
Requirement already satisfied: intel-cmplr-lib-ur==2024.2.0 in
/usr/local/lib/python3.11/dist-packages (from intel-
openmp<2026,>=2024->mkl->numpy>=1.17->transformers) (2024.2.0)
Requirement already satisfied: opencv-python in /usr/local/lib/python3.11/dist-
packages (4.11.0.86)
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.11/dist-
packages (from opencv-python) (1.26.4)
Requirement already satisfied: mkl_fft in /usr/local/lib/python3.11/dist-
packages (from numpy>=1.21.2->opencv-python) (1.3.8)
Requirement already satisfied: mkl_random in /usr/local/lib/python3.11/dist-
packages (from numpy>=1.21.2->opencv-python) (1.2.4)
Requirement already satisfied: mkl_umath in /usr/local/lib/python3.11/dist-
packages (from numpy>=1.21.2->opencv-python) (0.1.1)
Requirement already satisfied: mkl in /usr/local/lib/python3.11/dist-packages
(from numpy>=1.21.2->opencv-python) (2025.1.0)
Requirement already satisfied: tbb4py in /usr/local/lib/python3.11/dist-packages
(from numpy>=1.21.2->opencv-python) (2022.1.0)
Requirement already satisfied: mkl-service in /usr/local/lib/python3.11/dist-
packages (from numpy>=1.21.2->opencv-python) (2.4.1)
Requirement already satisfied: intel-openmp<2026,>=2024 in
/usr/local/lib/python3.11/dist-packages (from mkl->numpy>=1.21.2->opencv-python)
(2024.2.0)
```

No questions assigned to the following page.

```
Requirement already satisfied: tbb==2022.* in /usr/local/lib/python3.11/dist-
packages (from mkl->numpy>=1.21.2->opencv-python) (2022.1.0)
Requirement already satisfied: tcmlib==1.* in /usr/local/lib/python3.11/dist-
packages (from tbb==2022.*->mkl->numpy>=1.21.2->opencv-python) (1.2.0)
Requirement already satisfied: intel-cmplr-lib-rt in
/usr/local/lib/python3.11/dist-packages (from mkl_umath->numpy>=1.21.2->opencv-
python) (2024.2.0)
Requirement already satisfied: intel-cmplr-lib-ur==2024.2.0 in
/usr/local/lib/python3.11/dist-packages (from intel-
openmp<2026,>=2024->mkl->numpy>=1.21.2->opencv-python) (2024.2.0)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages
(2.2.3)
Requirement already satisfied: numpy>=1.23.2 in /usr/local/lib/python3.11/dist-
packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.11/dist-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-
packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-
packages (from pandas) (2025.2)
Requirement already satisfied: mkl_fft in /usr/local/lib/python3.11/dist-
packages (from numpy>=1.23.2->pandas) (1.3.8)
Requirement already satisfied: mkl_random in /usr/local/lib/python3.11/dist-
packages (from numpy>=1.23.2->pandas) (1.2.4)
Requirement already satisfied: mkl_umath in /usr/local/lib/python3.11/dist-
packages (from numpy>=1.23.2->pandas) (0.1.1)
Requirement already satisfied: mkl in /usr/local/lib/python3.11/dist-packages
(from numpy>=1.23.2->pandas) (2025.1.0)
Requirement already satisfied: tbb4py in /usr/local/lib/python3.11/dist-packages
(from numpy>=1.23.2->pandas) (2022.1.0)
Requirement already satisfied: mkl-service in /usr/local/lib/python3.11/dist-
packages (from numpy>=1.23.2->pandas) (2.4.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-
packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
Requirement already satisfied: intel-openmp<2026,>=2024 in
/usr/local/lib/python3.11/dist-packages (from mkl->numpy>=1.23.2->pandas)
(2024.2.0)
Requirement already satisfied: tbb==2022.* in /usr/local/lib/python3.11/dist-
packages (from mkl->numpy>=1.23.2->pandas) (2022.1.0)
Requirement already satisfied: tcmlib==1.* in /usr/local/lib/python3.11/dist-
packages (from tbb==2022.*->mkl->numpy>=1.23.2->pandas) (1.2.0)
Requirement already satisfied: intel-cmplr-lib-rt in
/usr/local/lib/python3.11/dist-packages (from mkl_umath->numpy>=1.23.2->pandas)
(2024.2.0)
Requirement already satisfied: intel-cmplr-lib-ur==2024.2.0 in
/usr/local/lib/python3.11/dist-packages (from intel-
openmp<2026,>=2024->mkl->numpy>=1.23.2->pandas) (2024.2.0)
Requirement already satisfied: albumentations in /usr/local/lib/python3.11/dist-
```

No questions assigned to the following page.

```
packages (2.0.4)
Requirement already satisfied: numpy>=1.24.4 in /usr/local/lib/python3.11/dist-
packages (from albumentations) (1.26.4)
Requirement already satisfied: scipy>=1.10.0 in /usr/local/lib/python3.11/dist-
packages (from albumentations) (1.15.2)
Requirement already satisfied: PyYAML in /usr/local/lib/python3.11/dist-packages
(from albumentations) (6.0.2)
Requirement already satisfied: pydantic>=2.9.2 in
/usr/local/lib/python3.11/dist-packages (from albumentations) (2.11.3)
Requirement already satisfied: albucore==0.0.23 in
/usr/local/lib/python3.11/dist-packages (from albumentations) (0.0.23)
Requirement already satisfied: opencv-python-headless>=4.9.0.80 in
/usr/local/lib/python3.11/dist-packages (from albumentations) (4.11.0.86)
Requirement already satisfied: stringzilla>=3.10.4 in
/usr/local/lib/python3.11/dist-packages (from albucore==0.0.23->albumentations)
(3.11.3)
Requirement already satisfied: simsimd>=5.9.2 in /usr/local/lib/python3.11/dist-
packages (from albucore==0.0.23->albumentations) (6.2.1)
Requirement already satisfied: mkl_fft in /usr/local/lib/python3.11/dist-
packages (from numpy>=1.24.4->albumentations) (1.3.8)
Requirement already satisfied: mkl_random in /usr/local/lib/python3.11/dist-
packages (from numpy>=1.24.4->albumentations) (1.2.4)
Requirement already satisfied: mkl_umath in /usr/local/lib/python3.11/dist-
packages (from numpy>=1.24.4->albumentations) (0.1.1)
Requirement already satisfied: mkl in /usr/local/lib/python3.11/dist-packages
(from numpy>=1.24.4->albumentations) (2025.1.0)
Requirement already satisfied: tbb4py in /usr/local/lib/python3.11/dist-packages
(from numpy>=1.24.4->albumentations) (2022.1.0)
Requirement already satisfied: mkl-service in /usr/local/lib/python3.11/dist-
packages (from numpy>=1.24.4->albumentations) (2.4.1)
Requirement already satisfied: annotated-types>=0.6.0 in
/usr/local/lib/python3.11/dist-packages (from pydantic>=2.9.2->albumentations)
(0.7.0)
Requirement already satisfied: pydantic-core==2.33.1 in
/usr/local/lib/python3.11/dist-packages (from pydantic>=2.9.2->albumentations)
(2.33.1)
Requirement already satisfied: typing-extensions>=4.12.2 in
/usr/local/lib/python3.11/dist-packages (from pydantic>=2.9.2->albumentations)
(4.13.1)
Requirement already satisfied: typing-inspection>=0.4.0 in
/usr/local/lib/python3.11/dist-packages (from pydantic>=2.9.2->albumentations)
(0.4.0)
Requirement already satisfied: intel-openmp<2026,>=2024 in
/usr/local/lib/python3.11/dist-packages (from
mkl->numpy>=1.24.4->albumentations) (2024.2.0)
Requirement already satisfied: tbb==2022.* in /usr/local/lib/python3.11/dist-
packages (from mkl->numpy>=1.24.4->albumentations) (2022.1.0)
Requirement already satisfied: tcmlib==1.* in /usr/local/lib/python3.11/dist-
```

No questions assigned to the following page.

```
packages (from tbb==2022.*->mkl->numpy>=1.24.4->albumentations) (1.2.0)
Requirement already satisfied: intel-cmplr-lib-rt in
/usr/local/lib/python3.11/dist-packages (from
mkl_umath->numpy>=1.24.4->albumentations) (2024.2.0)
Requirement already satisfied: intel-cmplr-lib-ur==2024.2.0 in
/usr/local/lib/python3.11/dist-packages (from intel-
openmp<2026,>=2024->mkl->numpy>=1.24.4->albumentations) (2024.2.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-
packages (3.7.5)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-
packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (4.56.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (1.4.8)
Requirement already satisfied: numpy<2,>=1.20 in /usr/local/lib/python3.11/dist-
packages (from matplotlib) (1.26.4)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (24.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.11/dist-
packages (from matplotlib) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (3.2.1)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: mkl_fft in /usr/local/lib/python3.11/dist-
packages (from numpy<2,>=1.20->matplotlib) (1.3.8)
Requirement already satisfied: mkl_random in /usr/local/lib/python3.11/dist-
packages (from numpy<2,>=1.20->matplotlib) (1.2.4)
Requirement already satisfied: mkl_umath in /usr/local/lib/python3.11/dist-
packages (from numpy<2,>=1.20->matplotlib) (0.1.1)
Requirement already satisfied: mkl in /usr/local/lib/python3.11/dist-packages
(from numpy<2,>=1.20->matplotlib) (2025.1.0)
Requirement already satisfied: tbb4py in /usr/local/lib/python3.11/dist-packages
(from numpy<2,>=1.20->matplotlib) (2022.1.0)
Requirement already satisfied: mkl-service in /usr/local/lib/python3.11/dist-
packages (from numpy<2,>=1.20->matplotlib) (2.4.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-
packages (from python-dateutil>=2.7->matplotlib) (1.17.0)
Requirement already satisfied: intel-openmp<2026,>=2024 in
/usr/local/lib/python3.11/dist-packages (from mkl->numpy<2,>=1.20->matplotlib)
(2024.2.0)
Requirement already satisfied: tbb==2022.* in /usr/local/lib/python3.11/dist-
packages (from mkl->numpy<2,>=1.20->matplotlib) (2022.1.0)
Requirement already satisfied: tcmlib==1.* in /usr/local/lib/python3.11/dist-
packages (from tbb==2022.*->mkl->numpy<2,>=1.20->matplotlib) (1.2.0)
```

No questions assigned to the following page.

```
Requirement already satisfied: intel-cmplr-lib-rt in
/usr/local/lib/python3.11/dist-packages (from
mkl_umath->numpy<2,>=1.20->matplotlib) (2024.2.0)
Requirement already satisfied: intel-cmplr-lib-ur==2024.2.0 in
/usr/local/lib/python3.11/dist-packages (from intel-
openmp<2026,>=2024->mkl->numpy<2,>=1.20->matplotlib) (2024.2.0)

Usage:
  pip3 install [options] <requirement specifier> [package-index-options] …
  pip3 install [options] -r <requirements file> [package-index-options] …
  pip3 install [options] [-e] <vcs project url> …
  pip3 install [options] [-e] <local project path> …
  pip3 install [options] <archive url/path> …

no such option: --upgrade1
Requirement already satisfied: kaggle in /usr/local/lib/python3.11/dist-packages
(1.7.4.2)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages
(4.67.1)
Requirement already satisfied: bleach in /usr/local/lib/python3.11/dist-packages
(from kaggle) (6.2.0)
Requirement already satisfied: certifi>=14.05.14 in
/usr/local/lib/python3.11/dist-packages (from kaggle) (2025.1.31)
Requirement already satisfied: charset-normalizer in
/usr/local/lib/python3.11/dist-packages (from kaggle) (3.4.1)
Requirement already satisfied: idna in /usr/local/lib/python3.11/dist-packages
(from kaggle) (3.10)
Requirement already satisfied: protobuf in /usr/local/lib/python3.11/dist-
packages (from kaggle) (3.20.3)
Requirement already satisfied: python-dateutil>=2.5.3 in
/usr/local/lib/python3.11/dist-packages (from kaggle) (2.9.0.post0)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.11/dist-
packages (from kaggle) (8.0.4)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-
packages (from kaggle) (2.32.3)
Requirement already satisfied: setuptools>=21.0.0 in
/usr/local/lib/python3.11/dist-packages (from kaggle) (75.1.0)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.11/dist-
packages (from kaggle) (1.17.0)
Requirement already satisfied: text-unidecode in /usr/local/lib/python3.11/dist-
packages (from kaggle) (1.3)
Requirement already satisfied: urllib3>=1.15.1 in
/usr/local/lib/python3.11/dist-packages (from kaggle) (2.3.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.11/dist-
packages (from kaggle) (0.5.1)
```

No questions assigned to the following page.

[2]:
```
import os
import cv2
import gc
import numpy as np
import pandas as pd
import itertools
from tqdm.autonotebook import tqdm
import albumentations as A
import matplotlib.pyplot as plt
import torch
from torch import nn
import torch.nn.functional as F
import timm
from transformers import DistilBertModel, DistilBertConfig, DistilBertTokenizer
import warnings
from tqdm import tqdm
import zipfile
import textwrap
```

```
/tmp/ipykernel_31/2717143393.py:7: TqdmExperimentalWarning: Using
`tqdm.autonotebook.tqdm` in notebook mode. Use `tqdm.tqdm` instead to force
console mode (e.g. in jupyter console)
  from tqdm.autonotebook import tqdm
/usr/local/lib/python3.11/dist-packages/albumentations/__init__.py:28:
UserWarning: A new version of Albumentations is available: '2.0.6' (you have
'2.0.4'). Upgrade using: pip install -U albumentations. To disable automatic
update checks, set the environment variable NO_ALBUMENTATIONS_UPDATE to 1.
  check_for_updates()
2025-05-04 14:52:35.826952: E
external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:477] Unable to register
cuFFT factory: Attempting to register factory for plugin cuFFT when one has
already been registered
WARNING: All log messages before absl::InitializeLog() is called are written to
STDERR
E0000 00:00:1746370356.000881      31 cuda_dnn.cc:8310] Unable to register cuDNN
factory: Attempting to register factory for plugin cuDNN when one has already
been registered
E0000 00:00:1746370356.053214      31 cuda_blas.cc:1418] Unable to register
cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has
already been registered
```

[3]:
```
# ------------------ KAGGLE CREDENTIALS ------------------

# Set your Kaggle credentials here

os.environ['KAGGLE_USERNAME'] = 'rmg9725'
os.environ['KAGGLE_KEY'] = 'ce53d991a9b10c4f65d0ec1f6794ebf8'
```

No questions assigned to the following page.

```
import kaggle
from kaggle.api.kaggle_api_extended import KaggleApi

api = KaggleApi()
api.authenticate()
api
```

[3]:
```
<kaggle.api.kaggle_api_extended.KaggleApi at 0x7df91ca17d10>
```

[4]:
```
"""
Data Download and Extraction Module

This section handles the downloading and extraction of the Flickr8k dataset.

It performs the following steps:
1. Defines paths for data directory, images, captions, and zip file
2. Downloads and extracts the dataset

"""

data_dir = './flickr8k_data'
images_dir = os.path.join(data_dir, 'Images')
captions_path = os.path.join(data_dir, 'captions.txt')
zip_path = os.path.join(data_dir, 'flickr8k.zip')

if not os.path.exists(images_dir):
    print(f"'{images_dir}' not found. Proceeding with download and extraction.")
    try:
        os.makedirs(data_dir, exist_ok=True)

        # Download the dataset using Kaggle API
        print("Downloading dataset...")
        api.dataset_download_files('adityajn105/flickr8k', path=data_dir,␣
 ↪quiet=False) # quiet=False shows progress
        print("Download complete.")

        # Extract the downloaded zip file
        print("Extracting dataset...")
        if os.path.exists(zip_path):
            import zipfile
            with zipfile.ZipFile(zip_path, 'r') as zip_ref:
                zip_ref.extractall(data_dir)

            # Clean up zip file
            os.remove(zip_path)
            print("Extraction complete and zip file removed.")
```

No questions assigned to the following page.

```
            if not os.path.exists(images_dir):
                print(f"Error: Extraction finished, but '{images_dir}'␣
 ↪directory not found.")
            else:
                print(f"Dataset ready in '{data_dir}'.")

        else:
            print(f"Error: Zip file '{zip_path}' not found after download␣
 ↪command.")

    except Exception as e:
        print(f"An error occurred during download or extraction: {e}")
else:
    print(f"Dataset already exists in '{images_dir}'. Skipping download and␣
 ↪extraction.")
```

```
'./flickr8k_data/Images' not found. Proceeding with download and extraction.
Downloading dataset…
Dataset URL: https://www.kaggle.com/datasets/adityajn105/flickr8k
Download complete.
Extracting dataset…
Extraction complete and zip file removed.
Dataset ready in './flickr8k_data'.
```

[5]:
```
"""
This section loads the captions file and displays sample images with their␣
 ↪captions.
"""

# Check if captions file exists
if not os.path.exists(captions_path):
    print(f"Error: Captions file not found at '{captions_path}'. Cannot display␣
 ↪samples.")
else:
    try:
        # Load captions using pandas - skip header row if present, name columns
        df_captions = pd.read_csv(captions_path, sep=',', header=0) # Assume␣
 ↪header row exists
        # Ensure columns are named 'image' and 'caption' (adjust if needed␣
 ↪based on actual file)
        if list(df_captions.columns) != ['image', 'caption']:
            print("Warning: Columns might not be 'image', 'caption'. Trying␣
 ↪renaming...")
            # Adjust based on the actual first few lines of your captions.txt
            # Example if no header and format is "image,caption":
            # df_captions = pd.read_csv(captions_path, sep=',',␣
 ↪names=['image', 'caption'])
```

No questions assigned to the following page.

```
            # Example if header exists but needs renaming:
            df_captions.columns = ['image', 'caption'] + list(df_captions.
↪columns[2:]) # Rename first two


        print("Captions loaded successfully. DataFrame head:")
        # print(df_captions.head())

        # Get the first 10 unique image filenames
        unique_images = df_captions['image'].unique()[:10]

        # Prepare data for plotting (image file and first caption)
        samples_to_plot = []
        for img_name in unique_images:
            first_caption = df_captions[df_captions['image'] ==␣
↪img_name]['caption'].iloc[0]
            samples_to_plot.append({'image': img_name, 'caption':␣
↪first_caption})

        # Display the images and captions
        fig, axes = plt.subplots(2, 5, figsize=(20, 8)) # Adjusted figsize
        axes = axes.flatten()

        print("\nDisplaying first 10 unique image samples:\n")

        for i, sample in enumerate(samples_to_plot):
            img_path = os.path.join(images_dir, sample['image'])
            ax = axes[i]
            try:
                if os.path.exists(img_path):
                    image = cv2.imread(img_path)
                    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # Convert␣
↪BGR to RGB
                    ax.imshow(image)
                    # Wrap caption text for display
                    wrapped_title = textwrap.fill(sample['caption'], width=45)␣
↪# Adjust width
                    ax.set_title(wrapped_title, fontsize=11) # Adjust fontsize
                else:
                    print(f"Image not found: {img_path}")
                    ax.set_title(f"Image not found:\n{sample['image']}",␣
↪fontsize=8, color='red')

            except Exception as e:
                print(f"Error loading/plotting {sample['image']}: {e}")
```

No questions assigned to the following page.

```
                ax.set_title(f"Error loading:\n{sample['image']}", fontsize=8,␣
↪color='red')

            ax.axis('off') # Hide axes ticks

        plt.tight_layout(pad=1.5) # Add padding between subplots
        plt.show()

    except Exception as e:
        print(f"An error occurred while loading captions or plotting: {e}")
```

```
Captions loaded successfully. DataFrame head:

Displaying first 10 unique image samples:
```

[ ]:

2.1 Hyperparameters

We will use the following hyperparameters.

[6]:
```
class config:
    # Paths updated to match current data directory
    data_dir = 'flickr8k_data'
    image_path = os.path.join(data_dir, "Images")
    captions_path = data_dir # Directory containing captions.txt

    debug = False
    batch_size = 32
    num_workers = 2
    head_lr = 1e-3
    image_encoder_lr = 1e-4
```

No questions assigned to the following page.

```
    text_encoder_lr = 1e-5
    weight_decay = 1e-3
    patience = 1
    factor = 0.8
    epochs = 4

    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

    # will be changed per config. later

    model_name = 'resnet50'
    image_embedding = 2048
    text_encoder_model = "distilbert-base-uncased"
    text_embedding = 768
    text_tokenizer = "distilbert-base-uncased"
    max_length = 200

    pretrained = True # for both image encoder and text encoder
    trainable = True # for both image encoder and text encoder
    temperature = 1.0

    # image size
    size = 224

    # for projection head; used for both image and text encoders
    num_projection_layers = 1
    projection_dim = 256
    dropout = 0.1
```

[7]:
```
print("--- Hyperparameters Defined ---")
print(f"Device: {config.device}")
print(f"Default Image Encoder: {config.model_name}")
print(f"Default Text Encoder: {config.text_encoder_model}")
print(f"Batch Size: {config.batch_size}")
print(f"Epochs: {config.epochs}")
```

```
--- Hyperparameters Defined ---
Device: cuda
Default Image Encoder: resnet50
Default Text Encoder: distilbert-base-uncased
Batch Size: 32
Epochs: 4
```

2.2 Dataset

Whenever we work with text and images, we'll need to do a bit of tedious preprocessing. Here, we'll have to tokenize the sentence descriptions of images before passing them through an embedding and apply a transform to the images before loading them.

No questions assigned to the following page.

[8]:
```
"""
CLIPDataset Class Implementation - Loading and Preprocessing of
image-caption pairs

It does:
1. Efficient initialization and caption tokenization
2. Loads images using OpenCV and converts from BGR to RGB color space
3. Applies transformations to images (resize, normalization, etc.)
4. Converts images to PyTorch tensors with correct channel ordering (CHW)
5. Includes robust error handling for missing or corrupted images
6. Returns a dictionary containing the processed image tensor and tokenized␣
 ↪caption

"""



class CLIPDataset(torch.utils.data.Dataset):
    def __init__(self, image_filenames, captions, tokenizer, transforms):

        self.image_filenames = image_filenames
        self.captions = list(captions)


        # Tokenize captions during initialization

        self.encoded_captions = tokenizer(
            list(captions), padding=True, truncation=True, max_length=config.
 ↪max_length
        )
        self.transforms = transforms


    def __getitem__(self, idx):
        # Get pre-tokenized caption data
        item = {
            key: torch.tensor(values[idx])
            for key, values in self.encoded_captions.items()
        }

        image_file_path = os.path.join(config.image_path, self.
 ↪image_filenames[idx])

        try:
            image = cv2.imread(image_file_path)
            if image is None:
                raise FileNotFoundError(f"Image not found or failed to load:␣
 ↪{image_file_path}")
```

No questions assigned to the following page.

```
            # Convert color BGR -> RGB
            image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

            # Apply transformations (e.g., resize, normalize)
            image = self.transforms(image=image)['image']

            # Convert to tensor and permute dimensions (HWC -> CHW)
            item['image'] = torch.tensor(image).permute(2, 0, 1).float()

            # Include the raw caption text (optional, but can be useful)
            item['caption'] = self.captions[idx]

        except Exception as e:
             print(f"Error loading/processing image {image_file_path} at index␣
↪{idx}: {e}")
             # Handle error: return a dummy image tensor and mark caption
             item['image'] = torch.zeros((3, config.size, config.size)) # Dummy␣
↪image
             item['caption'] = "[IMAGE LOAD ERROR]"

        return item



    def __len__(self):
        return len(self.captions)




print("\n--- CLIPDataset Class Defined ---")
```

```
--- CLIPDataset Class Defined ---
```

2.3 Embeddings

We'll need a way of encoding the images and text. For the images, we'll use a ResNet to get images down to a latent space of 2048. For the text, we'll use a Bert model to get images down to a latent space of 768.

Of course, to compare the embedded texts and images, they'll have to live in the same dimension. We can fix this by adding a projection head on top of the embeddings.

[9]:
```
"""
ImageEncoder Module
```

No questions assigned to the following page.

```
This module defines the image encoding component of the CLIP architecture.
It leverages the timm library to utilize pre-trained vision models with the
following characteristics:

1. Flexible model selection through configuration parameters
2. Outputs feature vectors from the final layer before classification
3. Uses global average pooling to produce fixed-size feature vectors

--------

NOTE: The encoder serves as the vision branch of the CLIP model, transforming
images into a high-dimensional embedding space that can be further projected
to match the text embedding dimension for contrastive learning.

"""


# --- Image Encoder ---
class ImageEncoder(nn.Module):

    def __init__(
        self, model_name=config.model_name, pretrained=config.pretrained,␣
 ↪trainable=config.trainable
    ):
        super().__init__()

        # num_classes=0 and global_pool='avg' ensures we get features before␣
 ↪the final classification layer

        self.model = timm.create_model(
            model_name, pretrained=pretrained, num_classes=0, global_pool="avg"
        )

        for p in self.model.parameters():
            p.requires_grad = trainable

    def forward(self, x):


        # Pass the input image batch through the timm model

        return self.model(x)
```

[10]:
```
"""
TextEncoder Module

This module defines the text encoding component of the CLIP architecture.
```

No questions assigned to the following page.

```
It basically uses DistilBERT as the base model for efficient text encoding,
but it is configurable.

3. Controllable parameter freezing for transfer learning scenarios
4. Extracts the [CLS] token embedding as the sentence representation
5. Processes both input token IDs and attention masks

The encoder serves as the language branch of the CLIP model, transforming
text inputs into a high-dimensional embedding space that can be further
projected to match the image embedding dimension for contrastive learning.
"""

class TextEncoder(nn.Module):

    def __init__(self, model_name=config.text_encoder_model, pretrained=config.
 ↪pretrained, trainable=config.trainable):
        super().__init__()

        if pretrained:
            self.model = DistilBertModel.from_pretrained(model_name)
        else:
            self.model = DistilBertModel(config=DistilBertConfig.
 ↪from_pretrained(model_name))



        for p in self.model.parameters():
            p.requires_grad = trainable


        # We use the hidden state of the first token ([CLS]) as the sentence␣
 ↪embedding

        self.target_token_idx = 0

    def forward(self, input_ids, attention_mask):
        output = self.model(input_ids=input_ids, attention_mask=attention_mask)
        last_hidden_state = output.last_hidden_state
        return last_hidden_state[:, self.target_token_idx, :]
```

[11]:
```
"""

Projects embeddings to the configured projection dimension.
Includes linear layers, activation, dropout, residual connection, and layer␣
 ↪normalization.
```

No questions assigned to the following page.

```
"""


class ProjectionHead(nn.Module):

    def __init__(
        self,
        embedding_dim,                          # Input dimension (from image or
 ↪text encoder)
        projection_dim=config.projection_dim, # Output dimension (shared space)
        dropout=config.dropout
    ):
        super().__init__()
        self.projection = nn.Linear(embedding_dim, projection_dim)
        self.gelu = nn.GELU()
        self.fc = nn.Linear(projection_dim, projection_dim)
        self.dropout = nn.Dropout(dropout)
        self.layer_norm = nn.LayerNorm(projection_dim)

    def forward(self, x):
        projected = self.projection(x)
        x = self.gelu(projected)
        x = self.fc(x)
        x = self.dropout(x)
        x = x + projected
        x = self.layer_norm(x)
        return x

print("\n--- ImageEncoder, TextEncoder, and ProjectionHead Classes Defined ---")
```

```
--- ImageEncoder, TextEncoder, and ProjectionHead Classes Defined ---
```

2.4 CLIP Architecture

Now we're ready to implement the CLIP architecture. As we discussed, we'll first have to embed the texts and images and apply the projection head to get embedded vectors in the same dimension.

Once we have the embedded vectors in hand, we'll compute the loss by taking the outer product of the embedded images with the embedded texts. This gives a Gram matrix where entry (i, j) is the inner product between the ith text embedding and jth image embedding. If we didn't have text or image duplicates, we'd want this matrix to be as close to the identity as possible: entry (i, i) is the inner product between the ith image and text pair which we want to be large while entry (i, j) (for $i \neq j$) should be small.

We can encourage this by making our loss the cross entropy loss between the identity matrix and our gram matrix. The picture gets a little bit more complicated when we have duplicates (each image has five different captions). We can deal with this by replacing the identity matrix with a

No questions assigned to the following page.

matrix encoding whether different texts (images) are the same.

[12]:
```
"""
CLIP Model Implementation

This module defines the complete CLIP (Contrastive Language-Image Pre-training) model
that connects the image and text encoders through contrastive learning.

Key features ofthe function are:

1. Combines image and text encoders with their RESPECTIVE projection heads
2. Configurable temperature parameter to control the sharpness of similarity distributions
3. Calculates cosine similarity between all image-text pairs in a batch
4. Implements a symmetric contrastive loss using cross-entropy

"""


class CLIPModel(nn.Module):


    def __init__(
        self,
        temperature=config.temperature,

        # Pass embedding dimensions explicitly to ensure ProjectionHeads are initialized correctly
        image_embedding=config.image_embedding,
        text_embedding=config.text_embedding,

    ):
        super().__init__()


        # Instantiate encoders and projection heads using current config values

        # Important: These will use the model names set in the config when an instance is created


        self.image_encoder = ImageEncoder(model_name=config.model_name)
        self.text_encoder = TextEncoder(model_name=config.text_encoder_model)
        self.image_projection = ProjectionHead(embedding_dim=image_embedding)
        self.text_projection = ProjectionHead(embedding_dim=text_embedding)
        self.temperature = temperature
```

No questions assigned to the following page.

```
#----


    def forward(self, batch):

        # 1. Get Image and Text Features from encoders

        image_features = self.image_encoder(batch["image"])
        text_features = self.text_encoder(
            input_ids=batch["input_ids"], attention_mask=batch["attention_mask"]
        )

        # 2. Project features to the shared embedding space

        image_embeddings = self.image_projection(image_features)
        text_embeddings = self.text_projection(text_features)

        # 3. Calculate the Contrastive Loss

        # Calculate cosine similarity logits (dot product of embeddings)
        # Scaling by temperature happens here
        # Shape: (batch_size, batch_size)

        logits = (text_embeddings @ image_embeddings.T) / self.temperature


        # Calculate similarity within modalities to create targets
        # Shape: (batch_size, batch_size) for both

        images_similarity = image_embeddings @ image_embeddings.T
        texts_similarity = text_embeddings @ text_embeddings.T


        # Create target distribution based on average similarity across␣
↪modalities
        # Softmax makes it a probability distribution
        # Scaling by temperature here influences the sharpness of the target␣
↪distribution
        # Shape: (batch_size, batch_size)

        targets = F.softmax(
            (images_similarity + texts_similarity) / 2 * self.temperature,␣
↪dim=-1
        )
```

Question assigned to the following page: 2.1

```
        # Calculate cross-entropy loss symmetrically
        # Compare text-image logits with targets

        texts_loss = cross_entropy(logits, targets, reduction='none') # Loss␣
 ↪for each text embedding


        # Compare image-text logits (logits.T) with targets.T
        images_loss = cross_entropy(logits.T, targets.T, reduction='none') #␣
 ↪Loss for each image embedding


        # Combine the losses (average the symmetric losses)
        # Result is a loss value per sample in the batch, shape: (batch_size)
        loss = (images_loss + texts_loss) / 2.0

        # Return the mean loss across the batch
        return loss.mean()


# Helper function for manual cross-entropy calculation

def cross_entropy(preds, targets, reduction='none'):

    log_softmax = nn.LogSoftmax(dim=-1)
    loss = (-targets * log_softmax(preds)).sum(1)

    if reduction == "none":
        return loss
    elif reduction == "mean":
        return loss.mean()
    else:
        raise ValueError(f"Invalid reduction specified: {reduction}. Use 'none'␣
 ↪or 'mean'.")

print("\n--- CLIPModel Class and cross_entropy Function Defined ---")
```

```
--- CLIPModel Class and cross_entropy Function Defined ---
```

2.5 Training

[13]:
```
"""

Data Splitting and DataLoader Functions as given in sample
```

Question assigned to the following page: 2.1

```
1. Splits the caption_df into training (80%) and validation (20%) sets
2. Applies image transformations
3. Creates PyTorch DataLoaders
4. Implements shuffling only for training data to improve model generalization
5. Uses pin_memory and num_workers for optimized data loading performance

"""

def make_train_valid_dfs():

    caption_df = pd.read_csv(f"{config.captions_path}/captions.csv")

    if caption_df.empty:
        print("Warning: Input dataframe is empty in make_train_valid_dfs.")
        return pd.DataFrame(), pd.DataFrame()


    max_id = caption_df["id"].max() + 1
    if config.debug:
         max_id = min(100, max_id) # Limit IDs if debugging
    image_ids = np.arange(0, max_id)



    if len(image_ids) == 0:
        print("Warning: No valid image IDs found in make_train_valid_dfs.")
        return pd.DataFrame(), pd.DataFrame()

    np.random.seed(42)
    valid_ids = np.random.choice(
        image_ids, size=max(1, int(0.2 * len(image_ids))), replace=False #␣
 ↪Ensure at least 1 validation ID if possible
    )
    train_ids = [id_ for id_ in image_ids if id_ not in valid_ids]



    train_dataframe = caption_df[caption_df["id"].isin(train_ids)].
 ↪reset_index(drop=True)
    valid_dataframe = caption_df[caption_df["id"].isin(valid_ids)].
 ↪reset_index(drop=True)

    print(f"Data split: Training samples = {len(train_dataframe)}, Validation␣
 ↪samples = {len(valid_dataframe)}")
    return train_dataframe, valid_dataframe
```

Question assigned to the following page: 2.1

```
def build_loaders(dataframe, tokenizer, mode):


    if dataframe.empty:
        print(f"Warning: Cannot build loader for mode '{mode}' - empty␣
 ↪dataframe.")
        return torch.utils.data.DataLoader(list()) # Return empty loader

    transforms = A.Compose([
        A.Resize(config.size, config.size, always_apply=True),
        A.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225],␣
 ↪max_pixel_value=255.0, always_apply=True),
        ])

    # Create the dataset instance
    dataset = CLIPDataset(
        dataframe["image"].values,
        dataframe["caption"].values,
        tokenizer=tokenizer,
        transforms=transforms,
    )

    # Create the DataLoader
    dataloader = torch.utils.data.DataLoader(
        dataset,
        batch_size=config.batch_size,
        num_workers=config.num_workers,
        shuffle=(mode == "train"), # Shuffle only for training
        pin_memory=True # Helps speed up CPU-to-GPU transfer
    )
    return dataloader

print("\n--- Data splitting (make_train_valid_dfs) and DataLoader␣
 ↪(build_loaders) functions defined ---")
```

```
--- Data splitting (make_train_valid_dfs) and DataLoader (build_loaders)
functions defined ---
```

[14]:
```
"""

MOre helper functions like:

1. AvgMeter class for computing and storing running averages of metrics

2. get_lr function to extract the current learning rate from an optimizer
```

Question assigned to the following page: 2.1

```
"""


# --- Training Helper Utilities ---
class AvgMeter:


    def __init__(self, name="Metric"):
        self.name = name
        self.reset()

    def reset(self):
        self.avg, self.sum, self.count = [0.0] * 3 # Initialize with floats

    def update(self, val, count=1):
        self.count += count
        self.sum += val * count
        self.avg = self.sum / self.count if self.count != 0 else 0.0

    def __repr__(self):
        return f"{self.name}: {self.avg:.4f}"

def get_lr(optimizer):

    for param_group in optimizer.param_groups:
        return param_group["lr"]
    return 0.0

print("\n--- AvgMeter class and get_lr function defined ---")
```

```
--- AvgMeter class and get_lr function defined ---
```

[15]:
```
"""
Training and Validation Epoch Functions

This module implements the core training and evaluation loops:

1. def train_epoch:
   - Executes one complete training epoch
   - Handles batch processing with gradient updates
   - Updates learning rate scheduler when in batch step mode

2. def valid_epoch:
   - Evaluates model performance on validation data

"""
```

Question assigned to the following page: 2.1

```python
def train_epoch(model, train_loader, optimizer, lr_scheduler, step_mode):

    loss_meter = AvgMeter(name="Train Loss")
    model.train() # Set model to training mode
    tqdm_object = tqdm(train_loader, total=len(train_loader), desc="Training")
    for batch in tqdm_object:


        # Move batch to device, excluding non-tensor 'caption' field

        batch_dev = {k: v.to(config.device) for k, v in batch.items() if␣
 ↪isinstance(v, torch.Tensor)}


        # Check if batch is valid

        if 'image' not in batch_dev or batch_dev['image'].nelement() == 0:
             print("Warning: Skipping empty or invalid batch in train_epoch.")
             continue

        loss = model(batch_dev)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        if step_mode == "batch":
            lr_scheduler.step()

        count = batch_dev["image"].size(0)
        loss_meter.update(loss.item(), count)
        tqdm_object.set_postfix(train_loss=loss_meter.avg, lr=get_lr(optimizer))

    return loss_meter


def valid_epoch(model, valid_loader):

    loss_meter = AvgMeter(name="Valid Loss")
    model.eval() # Set model to evaluation mode
    tqdm_object = tqdm(valid_loader, total=len(valid_loader), desc="Validation")

    with torch.no_grad():
        for batch in tqdm_object:
```

Question assigned to the following page: 2.1

```
            batch_dev = {k: v.to(config.device) for k, v in batch.items() if␣
 ↪isinstance(v, torch.Tensor)}

            if 'image' not in batch_dev or batch_dev['image'].nelement() == 0:
                print("Warning: Skipping empty or invalid batch in valid_epoch.
 ↪")
                continue

            loss = model(batch_dev)

            count = batch_dev["image"].size(0)
            loss_meter.update(loss.item(), count)
            tqdm_object.set_postfix(valid_loss=loss_meter.avg)

    return loss_meter

print("\nTraining utilities setup complete. Ready to initialize training run.")
```

```
Training utilities setup complete. Ready to initialize training run.
```

[16]:
```
# Creating df


df = pd.read_csv(f"{config.captions_path}/captions.txt")
df['id'] = [id_ // 5 for id_ in range(df.shape[0])]  # Assign IDs (one ID per 5␣
 ↪captions)
df.to_csv(f"{config.captions_path}/captions.csv", index=False)


caption_df = pd.read_csv(f"{config.captions_path}/captions.csv")
caption_df.head()
```

[16]:

	image	caption	id
0	1000268201_693b08cb0e.jpg	A child in a pink dress is climbing up a set o…	0
1	1000268201_693b08cb0e.jpg	A girl going into a wooden building .	0
2	1000268201_693b08cb0e.jpg	A little girl climbing into a wooden playhouse .	0
3	1000268201_693b08cb0e.jpg	A little girl climbing the stairs to her playh…	0
4	1000268201_693b08cb0e.jpg	A little girl in a pink dress going into a woo…	0

Question assigned to the following page: 2.1

[17]:
```python
# ---  Model Configs ---


from transformers import BertTokenizer, RobertaTokenizer

print("\n--- Preparing to Train Multiple CLIP Models ---")

# Define the three model configurations we'll train
model_configs = [
    {
        'name': 'ResNet18 + RoBERTa',
        'vision_encoder': 'resnet18',
        'text_encoder': 'roberta-base',
        'vision_embedding': 512,  # ResNet18 output dimension
        'text_embedding': 768,
        'tokenizer_class': RobertaTokenizer,
        'output_file': 'best_resnet18_roberta.pt'
    },
    {
        'name': 'ResNet34 + BERT',
        'vision_encoder': 'resnet34',
        'text_encoder': 'bert-base-uncased',
        'vision_embedding': 512,  # ResNet34 output dimension
        'text_embedding': 768,
        'tokenizer_class': BertTokenizer,
        'output_file': 'best_resnet34_bert.pt'
    },
    {
        'name': 'ResNet50 + DistilBERT',
        'vision_encoder': 'resnet50',
        'text_encoder': 'distilbert-base-uncased',
        'vision_embedding': 2048, # ResNet50 output dimension
        'text_embedding': 768,
        'tokenizer_class': DistilBertTokenizer,
        'output_file': 'best_resnet50_distilbert.pt'
    },

]
```

```
--- Preparing to Train Multiple CLIP Models ---
```

[18]:
```python
"""
CLIP Model Training Function

This comprehensive function handles the complete training pipeline for CLIP␣
 ↪models:
```

Question assigned to the following page: 2.1

```
1. Configures model-specific settings for vision and text encoders

2. Constructs the CLIP model with specified embedding dimensions

3. Loads and returns the best model based on validation performance


"""


def train_clip_model(model_config):




    print(f"\n=== Training {model_config['name']} ===")

    # Update config with model-specific settings
    config.model_name = model_config['vision_encoder']
    config.text_encoder_model = model_config['text_encoder']
    config.image_embedding = model_config['vision_embedding']
    config.text_embedding = model_config['text_embedding']


    # Initialize tokenizer
    tokenizer = model_config['tokenizer_class'].from_pretrained(config.
 ↪text_encoder_model)


    # Prepare data loaders
    print("Preparing data loaders...")
    train_df, valid_df = make_train_valid_dfs()
    print(f"Training samples: {len(train_df)}, Validation samples:␣
 ↪{len(valid_df)}")

    train_loader = build_loaders(train_df, tokenizer, mode="train")
    valid_loader = build_loaders(valid_df, tokenizer, mode="valid")



    # Initialize model
    print("Initializing model...")
    model = CLIPModel(
        temperature=config.temperature,
        image_embedding=config.image_embedding,
        text_embedding=config.text_embedding
```

Question assigned to the following page: 2.1

```
    ).to(config.device)



    # Set up optimizer with different learning rates for different components
    params = [
        {"params": model.image_encoder.parameters(), "lr": config.
↪image_encoder_lr},
        {"params": model.text_encoder.parameters(), "lr": config.
↪text_encoder_lr},
        {"params": itertools.chain(
            model.image_projection.parameters(), model.text_projection.
↪parameters()
        ), "lr": config.head_lr, "weight_decay": config.weight_decay}
    ]

    optimizer = torch.optim.AdamW(params, weight_decay=0.)
    lr_scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(
        optimizer, mode="min", patience=config.patience, factor=config.factor
    )


    # Training loop
    print(f"Beginning training for {config.epochs} epochs...")
    step = "epoch"
    best_loss = float('inf')

    # Initialize lists to store loss values for plotting
    train_losses = []
    valid_losses = []
    epochs = []

    for epoch in range(config.epochs):

        epochs.append(epoch + 1)

        print(f"Epoch: {epoch + 1}/{config.epochs}")

        train_loss = train_epoch(model, train_loader, optimizer, lr_scheduler,␣
↪step)

        train_losses.append(train_loss.avg)

        print(f"Train Loss: {train_loss.avg:.4f}")


        # Validate
```

Question assigned to the following page: 2.1

```
        with torch.no_grad():
            valid_loss = valid_epoch(model, valid_loader)
        valid_losses.append(valid_loss.avg)
        print(f"Valid Loss: {valid_loss.avg:.4f}")



        # Save best model

        if valid_loss.avg < best_loss:
            best_loss = valid_loss.avg
            torch.save(model.state_dict(), model_config['output_file'])
            print(f"Saved Best Model! (Loss: {best_loss:.4f})")



        # Update learning rate
        lr_scheduler.step(valid_loss.avg)


        # Clear memory
        gc.collect()
        torch.cuda.empty_cache() if torch.cuda.is_available() else None

    print(f"Training completed for {model_config['name']}. Best validation loss:
↪ {best_loss:.4f}")

    # Load best model
    model.load_state_dict(torch.load(model_config['output_file']))

    # return model, best_loss

    return model, best_loss, {
        'train_losses': train_losses,
        'valid_losses': valid_losses,
        'epochs': epochs
    }
```

[19]:
```
# ------ DRIVER CODE for training all 3 configs -------

trained_models = []
loss_histories = {}

for i, model_config in enumerate(model_configs):
    print(f"\n\n{'='*20} TRAINING MODEL {i+1}/3: {model_config['name']}␣
↪{'='*20}")
```

Question assigned to the following page: 2.1

```
    # Train the model
    model, best_loss, loss_history = train_clip_model(model_config)

    # Store results
    trained_models.append({
        'config': model_config,
        'model': model,
        'best_loss': best_loss
    })

    loss_histories[model_config['name']] = loss_history


    print(f"{'='*20} COMPLETED MODEL {i+1}/3 {'='*20}")




#-----




# Print summary of results
print("\n\n--- TRAINING SUMMARY ---")
print("Three CLIP models trained with the following results:")

for i, model_data in enumerate(trained_models):
    print(f"Model {i+1}: {model_data['config']['name']}")
    print(f"  - Vision Encoder: {model_data['config']['vision_encoder']}")
    print(f"  - Text Encoder: {model_data['config']['text_encoder']}")
    print(f"  - Best Validation Loss: {model_data['best_loss']:.4f}")
    print(f"  - Saved to: {model_data['config']['output_file']}")
    print()
```

```
==================== TRAINING MODEL 1/3: ResNet18 + RoBERTa ====================

=== Training ResNet18 + RoBERTa ===
tokenizer_config.json:   0%|          | 0.00/25.0 [00:00<?, ?B/s]
vocab.json:   0%|          | 0.00/899k [00:00<?, ?B/s]
merges.txt:   0%|          | 0.00/456k [00:00<?, ?B/s]
```

Question assigned to the following page: 2.1

```
tokenizer.json:   0%|          | 0.00/1.36M [00:00<?, ?B/s]

config.json:   0%|          | 0.00/481 [00:00<?, ?B/s]

Preparing data loaders…
Data split: Training samples = 32365, Validation samples = 8090
Training samples: 32365, Validation samples: 8090

/tmp/ipykernel_31/373562709.py:56: UserWarning: Argument(s) 'always_apply' are
not valid for transform Resize
  A.Resize(config.size, config.size, always_apply=True),
/tmp/ipykernel_31/373562709.py:57: UserWarning: Argument(s) 'always_apply' are
not valid for transform Normalize
  A.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225],
max_pixel_value=255.0, always_apply=True),

Initializing model…

model.safetensors:   0%|          | 0.00/46.8M [00:00<?, ?B/s]

You are using a model of type roberta to instantiate a model of type distilbert.
This is not supported for all configurations of models and can yield errors.
Xet Storage is enabled for this repo, but the 'hf_xet' package is not installed.
Falling back to regular HTTP download. For better performance, install the
package with: `pip install huggingface_hub[hf_xet]` or `pip install hf_xet`

model.safetensors:   0%|          | 0.00/499M [00:00<?, ?B/s]

Some weights of DistilBertModel were not initialized from the model checkpoint
at roberta-base and are newly initialized: ['embeddings.LayerNorm.bias',
'embeddings.LayerNorm.weight', 'embeddings.position_embeddings.weight',
'embeddings.word_embeddings.weight', 'transformer.layer.0.attention.k_lin.bias',
'transformer.layer.0.attention.k_lin.weight',
'transformer.layer.0.attention.out_lin.bias',
'transformer.layer.0.attention.out_lin.weight',
'transformer.layer.0.attention.q_lin.bias',
'transformer.layer.0.attention.q_lin.weight',
'transformer.layer.0.attention.v_lin.bias',
'transformer.layer.0.attention.v_lin.weight',
'transformer.layer.0.ffn.lin1.bias', 'transformer.layer.0.ffn.lin1.weight',
'transformer.layer.0.ffn.lin2.bias', 'transformer.layer.0.ffn.lin2.weight',
'transformer.layer.0.output_layer_norm.bias',
'transformer.layer.0.output_layer_norm.weight',
'transformer.layer.0.sa_layer_norm.bias',
'transformer.layer.0.sa_layer_norm.weight',
'transformer.layer.1.attention.k_lin.bias',
'transformer.layer.1.attention.k_lin.weight',
'transformer.layer.1.attention.out_lin.bias',
'transformer.layer.1.attention.out_lin.weight',
'transformer.layer.1.attention.q_lin.bias',
'transformer.layer.1.attention.q_lin.weight',
'transformer.layer.1.attention.v_lin.bias',
```

Question assigned to the following page: 2.1

```
'transformer.layer.1.attention.v_lin.weight',
'transformer.layer.1.ffn.lin1.bias', 'transformer.layer.1.ffn.lin1.weight',
'transformer.layer.1.ffn.lin2.bias', 'transformer.layer.1.ffn.lin2.weight',
'transformer.layer.1.output_layer_norm.bias',
'transformer.layer.1.output_layer_norm.weight',
'transformer.layer.1.sa_layer_norm.bias',
'transformer.layer.1.sa_layer_norm.weight',
'transformer.layer.10.attention.k_lin.bias',
'transformer.layer.10.attention.k_lin.weight',
'transformer.layer.10.attention.out_lin.bias',
'transformer.layer.10.attention.out_lin.weight',
'transformer.layer.10.attention.q_lin.bias',
'transformer.layer.10.attention.q_lin.weight',
'transformer.layer.10.attention.v_lin.bias',
'transformer.layer.10.attention.v_lin.weight',
'transformer.layer.10.ffn.lin1.bias', 'transformer.layer.10.ffn.lin1.weight',
'transformer.layer.10.ffn.lin2.bias', 'transformer.layer.10.ffn.lin2.weight',
'transformer.layer.10.output_layer_norm.bias',
'transformer.layer.10.output_layer_norm.weight',
'transformer.layer.10.sa_layer_norm.bias',
'transformer.layer.10.sa_layer_norm.weight',
'transformer.layer.11.attention.k_lin.bias',
'transformer.layer.11.attention.k_lin.weight',
'transformer.layer.11.attention.out_lin.bias',
'transformer.layer.11.attention.out_lin.weight',
'transformer.layer.11.attention.q_lin.bias',
'transformer.layer.11.attention.q_lin.weight',
'transformer.layer.11.attention.v_lin.bias',
'transformer.layer.11.attention.v_lin.weight',
'transformer.layer.11.ffn.lin1.bias', 'transformer.layer.11.ffn.lin1.weight',
'transformer.layer.11.ffn.lin2.bias', 'transformer.layer.11.ffn.lin2.weight',
'transformer.layer.11.output_layer_norm.bias',
'transformer.layer.11.output_layer_norm.weight',
'transformer.layer.11.sa_layer_norm.bias',
'transformer.layer.11.sa_layer_norm.weight',
'transformer.layer.2.attention.k_lin.bias',
'transformer.layer.2.attention.k_lin.weight',
'transformer.layer.2.attention.out_lin.bias',
'transformer.layer.2.attention.out_lin.weight',
'transformer.layer.2.attention.q_lin.bias',
'transformer.layer.2.attention.q_lin.weight',
'transformer.layer.2.attention.v_lin.bias',
'transformer.layer.2.attention.v_lin.weight',
'transformer.layer.2.ffn.lin1.bias', 'transformer.layer.2.ffn.lin1.weight',
'transformer.layer.2.ffn.lin2.bias', 'transformer.layer.2.ffn.lin2.weight',
'transformer.layer.2.output_layer_norm.bias',
'transformer.layer.2.output_layer_norm.weight',
'transformer.layer.2.sa_layer_norm.bias',
```

Question assigned to the following page: 2.1

```
'transformer.layer.2.sa_layer_norm.weight',
'transformer.layer.3.attention.k_lin.bias',
'transformer.layer.3.attention.k_lin.weight',
'transformer.layer.3.attention.out_lin.bias',
'transformer.layer.3.attention.out_lin.weight',
'transformer.layer.3.attention.q_lin.bias',
'transformer.layer.3.attention.q_lin.weight',
'transformer.layer.3.attention.v_lin.bias',
'transformer.layer.3.attention.v_lin.weight',
'transformer.layer.3.ffn.lin1.bias', 'transformer.layer.3.ffn.lin1.weight',
'transformer.layer.3.ffn.lin2.bias', 'transformer.layer.3.ffn.lin2.weight',
'transformer.layer.3.output_layer_norm.bias',
'transformer.layer.3.output_layer_norm.weight',
'transformer.layer.3.sa_layer_norm.bias',
'transformer.layer.3.sa_layer_norm.weight',
'transformer.layer.4.attention.k_lin.bias',
'transformer.layer.4.attention.k_lin.weight',
'transformer.layer.4.attention.out_lin.bias',
'transformer.layer.4.attention.out_lin.weight',
'transformer.layer.4.attention.q_lin.bias',
'transformer.layer.4.attention.q_lin.weight',
'transformer.layer.4.attention.v_lin.bias',
'transformer.layer.4.attention.v_lin.weight',
'transformer.layer.4.ffn.lin1.bias', 'transformer.layer.4.ffn.lin1.weight',
'transformer.layer.4.ffn.lin2.bias', 'transformer.layer.4.ffn.lin2.weight',
'transformer.layer.4.output_layer_norm.bias',
'transformer.layer.4.output_layer_norm.weight',
'transformer.layer.4.sa_layer_norm.bias',
'transformer.layer.4.sa_layer_norm.weight',
'transformer.layer.5.attention.k_lin.bias',
'transformer.layer.5.attention.k_lin.weight',
'transformer.layer.5.attention.out_lin.bias',
'transformer.layer.5.attention.out_lin.weight',
'transformer.layer.5.attention.q_lin.bias',
'transformer.layer.5.attention.q_lin.weight',
'transformer.layer.5.attention.v_lin.bias',
'transformer.layer.5.attention.v_lin.weight',
'transformer.layer.5.ffn.lin1.bias', 'transformer.layer.5.ffn.lin1.weight',
'transformer.layer.5.ffn.lin2.bias', 'transformer.layer.5.ffn.lin2.weight',
'transformer.layer.5.output_layer_norm.bias',
'transformer.layer.5.output_layer_norm.weight',
'transformer.layer.5.sa_layer_norm.bias',
'transformer.layer.5.sa_layer_norm.weight',
'transformer.layer.6.attention.k_lin.bias',
'transformer.layer.6.attention.k_lin.weight',
'transformer.layer.6.attention.out_lin.bias',
'transformer.layer.6.attention.out_lin.weight',
'transformer.layer.6.attention.q_lin.bias',
```

Question assigned to the following page: 2.1

```
'transformer.layer.6.attention.q_lin.weight',
'transformer.layer.6.attention.v_lin.bias',
'transformer.layer.6.attention.v_lin.weight',
'transformer.layer.6.ffn.lin1.bias', 'transformer.layer.6.ffn.lin1.weight',
'transformer.layer.6.ffn.lin2.bias', 'transformer.layer.6.ffn.lin2.weight',
'transformer.layer.6.output_layer_norm.bias',
'transformer.layer.6.output_layer_norm.weight',
'transformer.layer.6.sa_layer_norm.bias',
'transformer.layer.6.sa_layer_norm.weight',
'transformer.layer.7.attention.k_lin.bias',
'transformer.layer.7.attention.k_lin.weight',
'transformer.layer.7.attention.out_lin.bias',
'transformer.layer.7.attention.out_lin.weight',
'transformer.layer.7.attention.q_lin.bias',
'transformer.layer.7.attention.q_lin.weight',
'transformer.layer.7.attention.v_lin.bias',
'transformer.layer.7.attention.v_lin.weight',
'transformer.layer.7.ffn.lin1.bias', 'transformer.layer.7.ffn.lin1.weight',
'transformer.layer.7.ffn.lin2.bias', 'transformer.layer.7.ffn.lin2.weight',
'transformer.layer.7.output_layer_norm.bias',
'transformer.layer.7.output_layer_norm.weight',
'transformer.layer.7.sa_layer_norm.bias',
'transformer.layer.7.sa_layer_norm.weight',
'transformer.layer.8.attention.k_lin.bias',
'transformer.layer.8.attention.k_lin.weight',
'transformer.layer.8.attention.out_lin.bias',
'transformer.layer.8.attention.out_lin.weight',
'transformer.layer.8.attention.q_lin.bias',
'transformer.layer.8.attention.q_lin.weight',
'transformer.layer.8.attention.v_lin.bias',
'transformer.layer.8.attention.v_lin.weight',
'transformer.layer.8.ffn.lin1.bias', 'transformer.layer.8.ffn.lin1.weight',
'transformer.layer.8.ffn.lin2.bias', 'transformer.layer.8.ffn.lin2.weight',
'transformer.layer.8.output_layer_norm.bias',
'transformer.layer.8.output_layer_norm.weight',
'transformer.layer.8.sa_layer_norm.bias',
'transformer.layer.8.sa_layer_norm.weight',
'transformer.layer.9.attention.k_lin.bias',
'transformer.layer.9.attention.k_lin.weight',
'transformer.layer.9.attention.out_lin.bias',
'transformer.layer.9.attention.out_lin.weight',
'transformer.layer.9.attention.q_lin.bias',
'transformer.layer.9.attention.q_lin.weight',
'transformer.layer.9.attention.v_lin.bias',
'transformer.layer.9.attention.v_lin.weight',
'transformer.layer.9.ffn.lin1.bias', 'transformer.layer.9.ffn.lin1.weight',
'transformer.layer.9.ffn.lin2.bias', 'transformer.layer.9.ffn.lin2.weight',
'transformer.layer.9.output_layer_norm.bias',
```

Question assigned to the following page: 2.1

```
'transformer.layer.9.output_layer_norm.weight',
'transformer.layer.9.sa_layer_norm.bias',
'transformer.layer.9.sa_layer_norm.weight']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.

Beginning training for 4 epochs…
Epoch: 1/4

Training: 100%|     | 1012/1012 [03:18<00:00,  5.10it/s, lr=0.0001,
train_loss=3.62]

Train Loss: 3.6234

Validation: 100%|     | 253/253 [00:20<00:00, 12.57it/s, valid_loss=3.12]

Valid Loss: 3.1156
Saved Best Model! (Loss: 3.1156)
Epoch: 2/4

Training: 100%|     | 1012/1012 [03:17<00:00,  5.13it/s, lr=0.0001,
train_loss=2.74]

Train Loss: 2.7429

Validation: 100%|     | 253/253 [00:19<00:00, 12.93it/s, valid_loss=2.78]

Valid Loss: 2.7835
Saved Best Model! (Loss: 2.7835)
Epoch: 3/4

Training: 100%|     | 1012/1012 [03:17<00:00,  5.13it/s, lr=0.0001,
train_loss=1.95]

Train Loss: 1.9515

Validation: 100%|     | 253/253 [00:19<00:00, 12.77it/s, valid_loss=2.5]

Valid Loss: 2.4984
Saved Best Model! (Loss: 2.4984)
Epoch: 4/4

Training: 100%|     | 1012/1012 [03:17<00:00,  5.14it/s, lr=0.0001,
train_loss=1.35]

Train Loss: 1.3530

Validation: 100%|     | 253/253 [00:19<00:00, 12.66it/s, valid_loss=2.45]

Valid Loss: 2.4499
Saved Best Model! (Loss: 2.4499)
Training completed for ResNet18 + RoBERTa. Best validation loss: 2.4499

/tmp/ipykernel_31/165364774.py:121: FutureWarning: You are using `torch.load`
with `weights_only=False` (the current default value), which uses the default
pickle module implicitly. It is possible to construct malicious pickle data
which will execute arbitrary code during unpickling (See
```

Question assigned to the following page: 2.1

```
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for
more details). In a future release, the default value for `weights_only` will be
flipped to `True`. This limits the functions that could be executed during
unpickling. Arbitrary objects will no longer be allowed to be loaded via this
mode unless they are explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control of the
loaded file. Please open an issue on GitHub for any issues related to this
experimental feature.
  model.load_state_dict(torch.load(model_config['output_file']))

==================== COMPLETED MODEL 1/3 ====================


==================== TRAINING MODEL 2/3: ResNet34 + BERT ====================

=== Training ResNet34 + BERT ===

tokenizer_config.json:   0%|          | 0.00/48.0 [00:00<?, ?B/s]

vocab.txt:   0%|          | 0.00/232k [00:00<?, ?B/s]

tokenizer.json:   0%|          | 0.00/466k [00:00<?, ?B/s]

config.json:   0%|          | 0.00/570 [00:00<?, ?B/s]

Preparing data loaders…
Data split: Training samples = 32365, Validation samples = 8090
Training samples: 32365, Validation samples: 8090

/tmp/ipykernel_31/373562709.py:56: UserWarning: Argument(s) 'always_apply' are
not valid for transform Resize
  A.Resize(config.size, config.size, always_apply=True),
/tmp/ipykernel_31/373562709.py:57: UserWarning: Argument(s) 'always_apply' are
not valid for transform Normalize
  A.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225],
max_pixel_value=255.0, always_apply=True),

Initializing model…

model.safetensors:   0%|          | 0.00/87.3M [00:00<?, ?B/s]

You are using a model of type bert to instantiate a model of type distilbert.
This is not supported for all configurations of models and can yield errors.
Xet Storage is enabled for this repo, but the 'hf_xet' package is not installed.
Falling back to regular HTTP download. For better performance, install the
package with: `pip install huggingface_hub[hf_xet]` or `pip install hf_xet`

model.safetensors:   0%|          | 0.00/440M [00:00<?, ?B/s]

Some weights of DistilBertModel were not initialized from the model checkpoint
at bert-base-uncased and are newly initialized: ['embeddings.LayerNorm.bias',
'embeddings.LayerNorm.weight', 'embeddings.position_embeddings.weight',
'embeddings.word_embeddings.weight', 'transformer.layer.0.attention.k_lin.bias',
```

Question assigned to the following page: 2.1

```
'transformer.layer.0.attention.k_lin.weight',
'transformer.layer.0.attention.out_lin.bias',
'transformer.layer.0.attention.out_lin.weight',
'transformer.layer.0.attention.q_lin.bias',
'transformer.layer.0.attention.q_lin.weight',
'transformer.layer.0.attention.v_lin.bias',
'transformer.layer.0.attention.v_lin.weight',
'transformer.layer.0.ffn.lin1.bias', 'transformer.layer.0.ffn.lin1.weight',
'transformer.layer.0.ffn.lin2.bias', 'transformer.layer.0.ffn.lin2.weight',
'transformer.layer.0.output_layer_norm.bias',
'transformer.layer.0.output_layer_norm.weight',
'transformer.layer.0.sa_layer_norm.bias',
'transformer.layer.0.sa_layer_norm.weight',
'transformer.layer.1.attention.k_lin.bias',
'transformer.layer.1.attention.k_lin.weight',
'transformer.layer.1.attention.out_lin.bias',
'transformer.layer.1.attention.out_lin.weight',
'transformer.layer.1.attention.q_lin.bias',
'transformer.layer.1.attention.q_lin.weight',
'transformer.layer.1.attention.v_lin.bias',
'transformer.layer.1.attention.v_lin.weight',
'transformer.layer.1.ffn.lin1.bias', 'transformer.layer.1.ffn.lin1.weight',
'transformer.layer.1.ffn.lin2.bias', 'transformer.layer.1.ffn.lin2.weight',
'transformer.layer.1.output_layer_norm.bias',
'transformer.layer.1.output_layer_norm.weight',
'transformer.layer.1.sa_layer_norm.bias',
'transformer.layer.1.sa_layer_norm.weight',
'transformer.layer.10.attention.k_lin.bias',
'transformer.layer.10.attention.k_lin.weight',
'transformer.layer.10.attention.out_lin.bias',
'transformer.layer.10.attention.out_lin.weight',
'transformer.layer.10.attention.q_lin.bias',
'transformer.layer.10.attention.q_lin.weight',
'transformer.layer.10.attention.v_lin.bias',
'transformer.layer.10.attention.v_lin.weight',
'transformer.layer.10.ffn.lin1.bias', 'transformer.layer.10.ffn.lin1.weight',
'transformer.layer.10.ffn.lin2.bias', 'transformer.layer.10.ffn.lin2.weight',
'transformer.layer.10.output_layer_norm.bias',
'transformer.layer.10.output_layer_norm.weight',
'transformer.layer.10.sa_layer_norm.bias',
'transformer.layer.10.sa_layer_norm.weight',
'transformer.layer.11.attention.k_lin.bias',
'transformer.layer.11.attention.k_lin.weight',
'transformer.layer.11.attention.out_lin.bias',
'transformer.layer.11.attention.out_lin.weight',
'transformer.layer.11.attention.q_lin.bias',
'transformer.layer.11.attention.q_lin.weight',
'transformer.layer.11.attention.v_lin.bias',
```

Question assigned to the following page: 2.1

```
'transformer.layer.11.attention.v_lin.weight',
'transformer.layer.11.ffn.lin1.bias', 'transformer.layer.11.ffn.lin1.weight',
'transformer.layer.11.ffn.lin2.bias', 'transformer.layer.11.ffn.lin2.weight',
'transformer.layer.11.output_layer_norm.bias',
'transformer.layer.11.output_layer_norm.weight',
'transformer.layer.11.sa_layer_norm.bias',
'transformer.layer.11.sa_layer_norm.weight',
'transformer.layer.2.attention.k_lin.bias',
'transformer.layer.2.attention.k_lin.weight',
'transformer.layer.2.attention.out_lin.bias',
'transformer.layer.2.attention.out_lin.weight',
'transformer.layer.2.attention.q_lin.bias',
'transformer.layer.2.attention.q_lin.weight',
'transformer.layer.2.attention.v_lin.bias',
'transformer.layer.2.attention.v_lin.weight',
'transformer.layer.2.ffn.lin1.bias', 'transformer.layer.2.ffn.lin1.weight',
'transformer.layer.2.ffn.lin2.bias', 'transformer.layer.2.ffn.lin2.weight',
'transformer.layer.2.output_layer_norm.bias',
'transformer.layer.2.output_layer_norm.weight',
'transformer.layer.2.sa_layer_norm.bias',
'transformer.layer.2.sa_layer_norm.weight',
'transformer.layer.3.attention.k_lin.bias',
'transformer.layer.3.attention.k_lin.weight',
'transformer.layer.3.attention.out_lin.bias',
'transformer.layer.3.attention.out_lin.weight',
'transformer.layer.3.attention.q_lin.bias',
'transformer.layer.3.attention.q_lin.weight',
'transformer.layer.3.attention.v_lin.bias',
'transformer.layer.3.attention.v_lin.weight',
'transformer.layer.3.ffn.lin1.bias', 'transformer.layer.3.ffn.lin1.weight',
'transformer.layer.3.ffn.lin2.bias', 'transformer.layer.3.ffn.lin2.weight',
'transformer.layer.3.output_layer_norm.bias',
'transformer.layer.3.output_layer_norm.weight',
'transformer.layer.3.sa_layer_norm.bias',
'transformer.layer.3.sa_layer_norm.weight',
'transformer.layer.4.attention.k_lin.bias',
'transformer.layer.4.attention.k_lin.weight',
'transformer.layer.4.attention.out_lin.bias',
'transformer.layer.4.attention.out_lin.weight',
'transformer.layer.4.attention.q_lin.bias',
'transformer.layer.4.attention.q_lin.weight',
'transformer.layer.4.attention.v_lin.bias',
'transformer.layer.4.attention.v_lin.weight',
'transformer.layer.4.ffn.lin1.bias', 'transformer.layer.4.ffn.lin1.weight',
'transformer.layer.4.ffn.lin2.bias', 'transformer.layer.4.ffn.lin2.weight',
'transformer.layer.4.output_layer_norm.bias',
'transformer.layer.4.output_layer_norm.weight',
'transformer.layer.4.sa_layer_norm.bias',
```

Question assigned to the following page: 2.1

```
'transformer.layer.4.sa_layer_norm.weight',
'transformer.layer.5.attention.k_lin.bias',
'transformer.layer.5.attention.k_lin.weight',
'transformer.layer.5.attention.out_lin.bias',
'transformer.layer.5.attention.out_lin.weight',
'transformer.layer.5.attention.q_lin.bias',
'transformer.layer.5.attention.q_lin.weight',
'transformer.layer.5.attention.v_lin.bias',
'transformer.layer.5.attention.v_lin.weight',
'transformer.layer.5.ffn.lin1.bias', 'transformer.layer.5.ffn.lin1.weight',
'transformer.layer.5.ffn.lin2.bias', 'transformer.layer.5.ffn.lin2.weight',
'transformer.layer.5.output_layer_norm.bias',
'transformer.layer.5.output_layer_norm.weight',
'transformer.layer.5.sa_layer_norm.bias',
'transformer.layer.5.sa_layer_norm.weight',
'transformer.layer.6.attention.k_lin.bias',
'transformer.layer.6.attention.k_lin.weight',
'transformer.layer.6.attention.out_lin.bias',
'transformer.layer.6.attention.out_lin.weight',
'transformer.layer.6.attention.q_lin.bias',
'transformer.layer.6.attention.q_lin.weight',
'transformer.layer.6.attention.v_lin.bias',
'transformer.layer.6.attention.v_lin.weight',
'transformer.layer.6.ffn.lin1.bias', 'transformer.layer.6.ffn.lin1.weight',
'transformer.layer.6.ffn.lin2.bias', 'transformer.layer.6.ffn.lin2.weight',
'transformer.layer.6.output_layer_norm.bias',
'transformer.layer.6.output_layer_norm.weight',
'transformer.layer.6.sa_layer_norm.bias',
'transformer.layer.6.sa_layer_norm.weight',
'transformer.layer.7.attention.k_lin.bias',
'transformer.layer.7.attention.k_lin.weight',
'transformer.layer.7.attention.out_lin.bias',
'transformer.layer.7.attention.out_lin.weight',
'transformer.layer.7.attention.q_lin.bias',
'transformer.layer.7.attention.q_lin.weight',
'transformer.layer.7.attention.v_lin.bias',
'transformer.layer.7.attention.v_lin.weight',
'transformer.layer.7.ffn.lin1.bias', 'transformer.layer.7.ffn.lin1.weight',
'transformer.layer.7.ffn.lin2.bias', 'transformer.layer.7.ffn.lin2.weight',
'transformer.layer.7.output_layer_norm.bias',
'transformer.layer.7.output_layer_norm.weight',
'transformer.layer.7.sa_layer_norm.bias',
'transformer.layer.7.sa_layer_norm.weight',
'transformer.layer.8.attention.k_lin.bias',
'transformer.layer.8.attention.k_lin.weight',
'transformer.layer.8.attention.out_lin.bias',
'transformer.layer.8.attention.out_lin.weight',
'transformer.layer.8.attention.q_lin.bias',
```

Question assigned to the following page: 2.1

```
'transformer.layer.8.attention.q_lin.weight',
'transformer.layer.8.attention.v_lin.bias',
'transformer.layer.8.attention.v_lin.weight',
'transformer.layer.8.ffn.lin1.bias', 'transformer.layer.8.ffn.lin1.weight',
'transformer.layer.8.ffn.lin2.bias', 'transformer.layer.8.ffn.lin2.weight',
'transformer.layer.8.output_layer_norm.bias',
'transformer.layer.8.output_layer_norm.weight',
'transformer.layer.8.sa_layer_norm.bias',
'transformer.layer.8.sa_layer_norm.weight',
'transformer.layer.9.attention.k_lin.bias',
'transformer.layer.9.attention.k_lin.weight',
'transformer.layer.9.attention.out_lin.bias',
'transformer.layer.9.attention.out_lin.weight',
'transformer.layer.9.attention.q_lin.bias',
'transformer.layer.9.attention.q_lin.weight',
'transformer.layer.9.attention.v_lin.bias',
'transformer.layer.9.attention.v_lin.weight',
'transformer.layer.9.ffn.lin1.bias', 'transformer.layer.9.ffn.lin1.weight',
'transformer.layer.9.ffn.lin2.bias', 'transformer.layer.9.ffn.lin2.weight',
'transformer.layer.9.output_layer_norm.bias',
'transformer.layer.9.output_layer_norm.weight',
'transformer.layer.9.sa_layer_norm.bias',
'transformer.layer.9.sa_layer_norm.weight']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.

Beginning training for 4 epochs…
Epoch: 1/4

Training: 100%|     | 1012/1012 [03:45<00:00,  4.49it/s, lr=0.0001,
train_loss=3.66]

Train Loss: 3.6648

Validation: 100%|     | 253/253 [00:21<00:00, 11.95it/s, valid_loss=3.11]

Valid Loss: 3.1132
Saved Best Model! (Loss: 3.1132)
Epoch: 2/4

Training: 100%|     | 1012/1012 [03:45<00:00,  4.49it/s, lr=0.0001,
train_loss=2.7]

Train Loss: 2.6960

Validation: 100%|     | 253/253 [00:20<00:00, 12.06it/s, valid_loss=2.7]

Valid Loss: 2.6982
Saved Best Model! (Loss: 2.6982)
Epoch: 3/4

Training: 100%|     | 1012/1012 [03:45<00:00,  4.49it/s, lr=0.0001,
train_loss=1.88]
```

Question assigned to the following page: 2.1

```
Train Loss: 1.8809

Validation: 100%|      | 253/253 [00:20<00:00, 12.50it/s, valid_loss=2.49]

Valid Loss: 2.4910
Saved Best Model! (Loss: 2.4910)
Epoch: 4/4

Training: 100%|      | 1012/1012 [03:45<00:00,  4.49it/s, lr=0.0001,
train_loss=1.28]

Train Loss: 1.2788

Validation: 100%|      | 253/253 [00:20<00:00, 12.33it/s, valid_loss=2.47]

Valid Loss: 2.4658
Saved Best Model! (Loss: 2.4658)
Training completed for ResNet34 + BERT. Best validation loss: 2.4658

/tmp/ipykernel_31/165364774.py:121: FutureWarning: You are using `torch.load`
with `weights_only=False` (the current default value), which uses the default
pickle module implicitly. It is possible to construct malicious pickle data
which will execute arbitrary code during unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for
more details). In a future release, the default value for `weights_only` will be
flipped to `True`. This limits the functions that could be executed during
unpickling. Arbitrary objects will no longer be allowed to be loaded via this
mode unless they are explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control of the
loaded file. Please open an issue on GitHub for any issues related to this
experimental feature.
  model.load_state_dict(torch.load(model_config['output_file']))

==================== COMPLETED MODEL 2/3 ====================


==================== TRAINING MODEL 3/3: ResNet50 + DistilBERT
====================

=== Training ResNet50 + DistilBERT ===

tokenizer_config.json:   0%|          | 0.00/48.0 [00:00<?, ?B/s]

vocab.txt:   0%|          | 0.00/232k [00:00<?, ?B/s]

tokenizer.json:   0%|          | 0.00/466k [00:00<?, ?B/s]

config.json:   0%|          | 0.00/483 [00:00<?, ?B/s]

Preparing data loaders…
Data split: Training samples = 32365, Validation samples = 8090
Training samples: 32365, Validation samples: 8090
```

Question assigned to the following page: 2.1

```
/tmp/ipykernel_31/373562709.py:56: UserWarning: Argument(s) 'always_apply' are
not valid for transform Resize
  A.Resize(config.size, config.size, always_apply=True),
/tmp/ipykernel_31/373562709.py:57: UserWarning: Argument(s) 'always_apply' are
not valid for transform Normalize
  A.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225],
max_pixel_value=255.0, always_apply=True),

Initializing model…

model.safetensors:   0%|          | 0.00/102M [00:00<?, ?B/s]

Xet Storage is enabled for this repo, but the 'hf_xet' package is not installed.
Falling back to regular HTTP download. For better performance, install the
package with: `pip install huggingface_hub[hf_xet]` or `pip install hf_xet`

model.safetensors:   0%|          | 0.00/268M [00:00<?, ?B/s]

Beginning training for 4 epochs…
Epoch: 1/4

Training: 100%|     | 1012/1012 [04:01<00:00,  4.20it/s, lr=0.0001,
train_loss=2.16]

Train Loss: 2.1621

Validation: 100%|     | 253/253 [00:20<00:00, 12.37it/s, valid_loss=2.25]

Valid Loss: 2.2485
Saved Best Model! (Loss: 2.2485)
Epoch: 2/4

Training: 100%|     | 1012/1012 [04:01<00:00,  4.20it/s, lr=0.0001,
train_loss=0.659]

Train Loss: 0.6589

Validation: 100%|     | 253/253 [00:20<00:00, 12.19it/s, valid_loss=2.15]

Valid Loss: 2.1459
Saved Best Model! (Loss: 2.1459)
Epoch: 3/4

Training: 100%|     | 1012/1012 [04:00<00:00,  4.20it/s, lr=0.0001,
train_loss=0.378]

Train Loss: 0.3783

Validation: 100%|     | 253/253 [00:20<00:00, 12.38it/s, valid_loss=2.19]

Valid Loss: 2.1912
Epoch: 4/4

Training: 100%|     | 1012/1012 [04:01<00:00,  4.20it/s, lr=0.0001,
train_loss=0.272]

Train Loss: 0.2716
```

Question assigned to the following page: 2.1

```
Validation: 100%|      | 253/253 [00:20<00:00, 12.52it/s, valid_loss=2.32]

Valid Loss: 2.3244
Training completed for ResNet50 + DistilBERT. Best validation loss: 2.1459

/tmp/ipykernel_31/165364774.py:121: FutureWarning: You are using `torch.load`
with `weights_only=False` (the current default value), which uses the default
pickle module implicitly. It is possible to construct malicious pickle data
which will execute arbitrary code during unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for
more details). In a future release, the default value for `weights_only` will be
flipped to `True`. This limits the functions that could be executed during
unpickling. Arbitrary objects will no longer be allowed to be loaded via this
mode unless they are explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control of the
loaded file. Please open an issue on GitHub for any issues related to this
experimental feature.
  model.load_state_dict(torch.load(model_config['output_file']))

==================== COMPLETED MODEL 3/3 ====================


--- TRAINING SUMMARY ---
Three CLIP models trained with the following results:
Model 1: ResNet18 + RoBERTa
  - Vision Encoder: resnet18
  - Text Encoder: roberta-base
  - Best Validation Loss: 2.4499
  - Saved to: best_resnet18_roberta.pt

Model 2: ResNet34 + BERT
  - Vision Encoder: resnet34
  - Text Encoder: bert-base-uncased
  - Best Validation Loss: 2.4658
  - Saved to: best_resnet34_bert.pt

Model 3: ResNet50 + DistilBERT
  - Vision Encoder: resnet50
  - Text Encoder: distilbert-base-uncased
  - Best Validation Loss: 2.1459
  - Saved to: best_resnet50_distilbert.pt
```

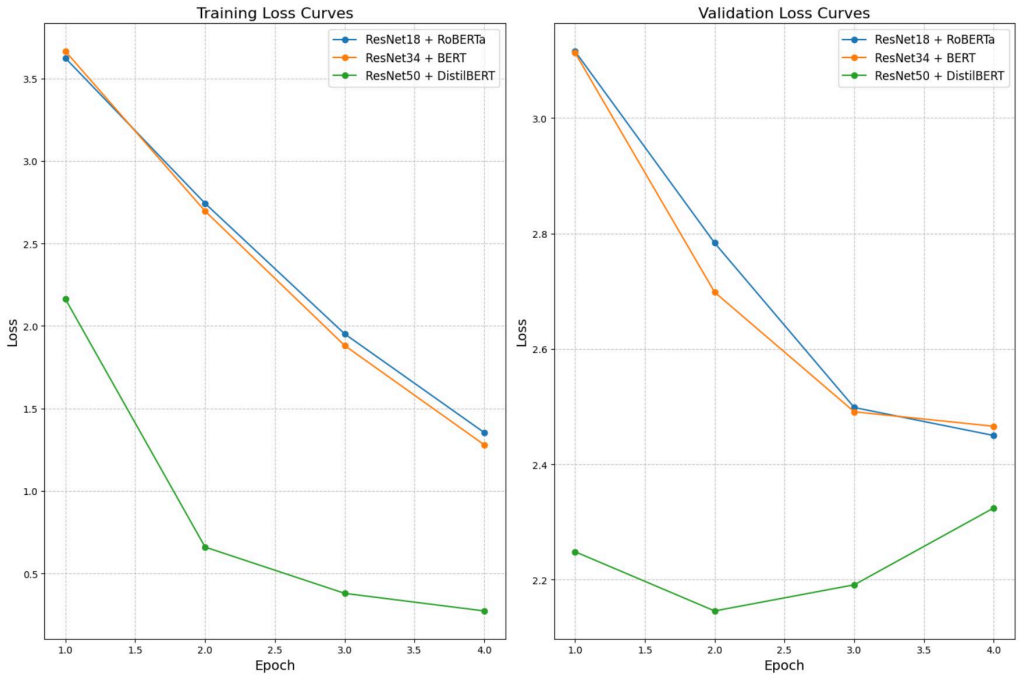
[20]:
```python
# --- Visualizing Loss Curves for All Models ---

plt.figure(figsize=(15, 10))

# Plot training losses
```

Question assigned to the following page: 2.1

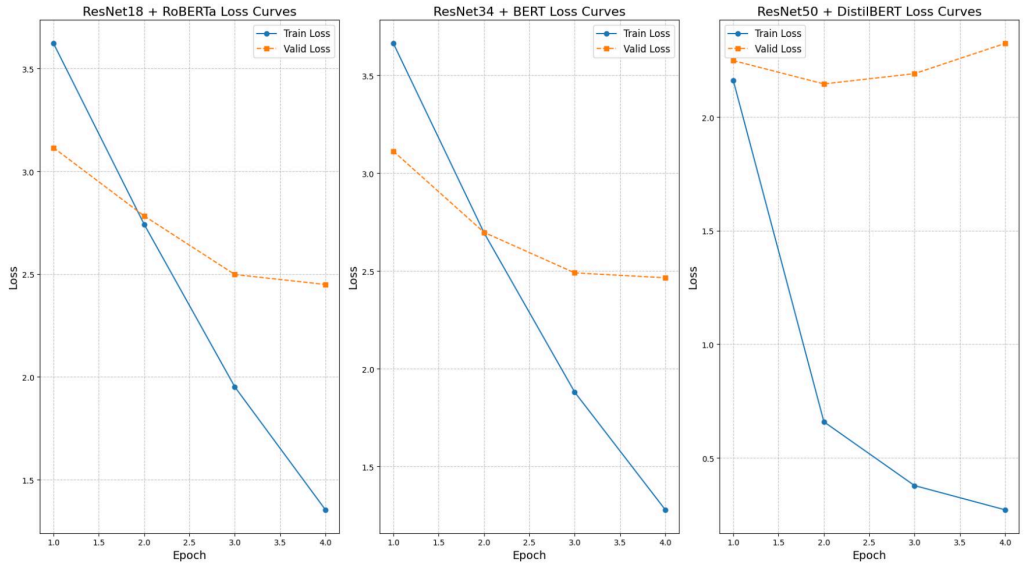
```
plt.subplot(1, 2, 1)
for model_name, history in loss_histories.items():
    plt.plot(history['epochs'], history['train_losses'], marker='o',␣
 ↪linestyle='-', label=f"{model_name}")

plt.title('Training Loss Curves', fontsize=16)
plt.xlabel('Epoch', fontsize=14)
plt.ylabel('Loss', fontsize=14)
plt.grid(True, linestyle='--', alpha=0.7)
plt.legend(fontsize=12)

# Plot validation losses
plt.subplot(1, 2, 2)
for model_name, history in loss_histories.items():
    plt.plot(history['epochs'], history['valid_losses'], marker='o',␣
 ↪linestyle='-', label=f"{model_name}")

plt.title('Validation Loss Curves', fontsize=16)
plt.xlabel('Epoch', fontsize=14)
plt.ylabel('Loss', fontsize=14)
plt.grid(True, linestyle='--', alpha=0.7)
plt.legend(fontsize=12)

plt.tight_layout()
plt.show()

# Plot combined loss curves for each model
plt.figure(figsize=(18, 10))
for i, (model_name, history) in enumerate(loss_histories.items()):
    plt.subplot(1, 3, i+1)
    plt.plot(history['epochs'], history['train_losses'], marker='o',␣
 ↪linestyle='-', label='Train Loss')
    plt.plot(history['epochs'], history['valid_losses'], marker='s',␣
 ↪linestyle='--', label='Valid Loss')
    plt.title(f'{model_name} Loss Curves', fontsize=16)
    plt.xlabel('Epoch', fontsize=14)
    plt.ylabel('Loss', fontsize=14)
    plt.grid(True, linestyle='--', alpha=0.7)
    plt.legend(fontsize=12)

plt.tight_layout()
plt.show()

# Create a table comparing final losses
model_names = []
final_train_losses = []
final_valid_losses = []
```

Question assigned to the following page: 2.1

```
best_valid_losses = []

for model_data in trained_models:
    model_name = model_data['config']['name']
    history = loss_histories[model_name]

    model_names.append(model_name)
    final_train_losses.append(history['train_losses'][-1])
    final_valid_losses.append(history['valid_losses'][-1])
    best_valid_losses.append(model_data['best_loss'])

# Create a DataFrame for the comparison table
comparison_df = pd.DataFrame({
    'Model': model_names,
    'Final Train Loss': final_train_losses,
    'Final Valid Loss': final_valid_losses,
    'Best Valid Loss': best_valid_losses
})

print("Loss Comparison Table:")
display(comparison_df)
```



Questions assigned to the following page: 2.2 and 2.1




```
Loss Comparison Table:
```

	Model	Final Train Loss	Final Valid Loss	Best Valid Loss
0	ResNet18 + RoBERTa	1.353031	2.449854	2.449854
1	ResNet34 + BERT	1.278813	2.465841	2.465841
2	ResNet50 + DistilBERT	0.271578	2.324401	2.145879

2.6 Evaluation

We will test our model by providing it a text and asking for images that are close to it. We'll start by embedding all our images and loading the model.

[39]:
```
"""
Image Embeddings Generation Function

This function generates embeddings for all images in the validation set using a␣
 ↪trained CLIP model

It also:

1. Constructs the CLIP model and loads pre-trained weights

2. Sets the model to evaluation mode for inference

3. Processes batches of images to generate embeddings:
   - Extracts features using the image encoder
   - Projects features to the shared embedding space
   - Handles potential empty or invalid batches
```

Question assigned to the following page: 2.2

```
"""

def get_image_embeddings(valid_df, model_path, model_config):

    print(f"\nGenerating image embeddings for {model_config['name']}...")

    config.model_name = model_config['vision_encoder']
    config.text_encoder_model = model_config['text_encoder']
    config.image_embedding = model_config['vision_embedding']
    config.text_embedding = model_config['text_embedding']


    # Initialize tokenizer
    tokenizer = model_config['tokenizer_class'].from_pretrained(config.
 ↪text_encoder_model)
    valid_loader = build_loaders(valid_df, tokenizer, mode="valid")

    # Initialize model and load weights
    model = CLIPModel(
        temperature=config.temperature,
        image_embedding=config.image_embedding,
        text_embedding=config.text_embedding
    ).to(config.device)

    model.load_state_dict(torch.load(model_path, map_location=config.device))
    model.eval()




    # Generate embeddings
    valid_image_embeddings = []
    with torch.no_grad():
        for batch in tqdm(valid_loader, desc="Embedding images"):
            batch_dev = {k: v.to(config.device) for k, v in batch.items() if␣
 ↪isinstance(v, torch.Tensor)}

            if 'image' not in batch_dev or batch_dev['image'].nelement() == 0:
                continue

            image_features = model.image_encoder(batch_dev["image"])
            image_embeddings = model.image_projection(image_features)
            valid_image_embeddings.append(image_embeddings)
```

Question assigned to the following page: 2.2

```
    return model, torch.cat(valid_image_embeddings)
```

[40]:
```
"""
Unique Image Matching Function

This function finds images that match a given text prompt while ensuring␣
 ↪uniqueness:

1. Encodes the text query using the model's tokenizer

2. Computes cosine similarity between the text embedding and all image␣
 ↪embeddings

3. Sorts images by similarity score in descending order

---------

IMPORTANT & ALSO USED IT FOR DEBUGGING:

4. Filters results to ensure uniqueness through two mechanisms:
   - Filename-based filtering to avoid retrieving previously returned images
   - Embedding-based similarity check to avoid near-duplicate images

5. Uses a configurable similarity threshold (0.95) to determine duplicates

"""


def find_matches_no_repeats(model, image_embeddings, query, image_filenames,␣
 ↪model_config,
                            already_retrieved=None, n=9):


    if already_retrieved is None:
        already_retrieved = set()

    print(f"\nFinding unique matches for prompt: '{query}' using␣
 ↪{model_config['name']}...")


    # Initialize tokenizer based on model config
    tokenizer = model_config['tokenizer_class'].from_pretrained(config.
 ↪text_encoder_model)

    # Encode the query
    encoded_query = tokenizer([query], padding=True, truncation=True,
                              max_length=config.max_length, return_tensors="pt")
```

Question assigned to the following page: 2.2

```
    batch = {k: v.to(config.device) for k, v in encoded_query.items()}


    # Generate text embedding
    with torch.no_grad():
        text_features = model.text_encoder(
            input_ids=batch["input_ids"], attention_mask=batch["attention_mask"]
        )
        text_embeddings = model.text_projection(text_features)

        # Normalize embeddings for cosine similarity
        image_embeddings_n = F.normalize(image_embeddings, p=2, dim=-1)
        text_embeddings_n = F.normalize(text_embeddings, p=2, dim=-1)

        # Compute similarity scores
        dot_similarity = text_embeddings_n @ image_embeddings_n.T


    # Get all similarity scores and sort them
    similarity_scores = dot_similarity.squeeze(0).cpu().numpy()


    # Create a list of (score, index) tuples and sort by score in descending␣
↪order
    scored_indices = [(score, idx) for idx, score in␣
↪enumerate(similarity_scores)]
    scored_indices.sort(reverse=True)


    # Filter out already retrieved images
    filtered_matches = []
    filtered_scores = []
    filtered_indices = []



    # Also keep track of selected embeddings to check for visual similarity
    selected_embeddings = []

    for score, idx in scored_indices:
        filename = image_filenames[idx]
        current_embedding = image_embeddings_n[idx].cpu().numpy()

        # Skip if filename is already retrieved
        if filename in already_retrieved:
            continue
```

Question assigned to the following page: 2.2

```
        # Check for visual similarity with already selected images
        is_similar = False
        for selected_emb in selected_embeddings:
            # Compute cosine similarity between embeddings
            similarity = np.dot(current_embedding, selected_emb)
            # If similarity is above threshold (e.g., 0.95), consider as␣
↪duplicate
            if similarity > 0.95:  # Adjust this threshold as needed
                is_similar = True
                break

        if not is_similar:
            filtered_matches.append(filename)
            filtered_scores.append(score)
            filtered_indices.append(idx)
            selected_embeddings.append(current_embedding)

            # Stop once we have enough matches
            if len(filtered_matches) >= n:
                break



# --------


    # If we couldn't find enough new images, warn
    if len(filtered_matches) < n:
        print(f"Warning: Could only find {len(filtered_matches)} unique images␣
↪for this prompt.")

    # Convert filtered scores back to tensor
    filtered_scores = torch.tensor(filtered_scores, device=config.device)




    # Display results
    if len(filtered_matches) > 0:
        rows = int(np.ceil(np.sqrt(len(filtered_matches))))
        cols = int(np.ceil(len(filtered_matches) / rows))
        fig, axes = plt.subplots(rows, cols, figsize=(12, 12))
        fig.suptitle(f"Top {len(filtered_matches)} unique matches for:␣
↪'{query}'\nModel: {model_config['name']}",
                     fontsize=16)
```

Question assigned to the following page: 2.2

```
        # Handle single image case
        if len(filtered_matches) == 1:
            axes = np.array([axes])

        # Make sure axes is always a flattened array
        axes = np.array(axes).flatten()

        for i, (match, ax) in enumerate(zip(filtered_matches, axes)):
            try:
                image = cv2.imread(f"{config.image_path}/{match}")
                if image is None:
                    raise FileNotFoundError(f"Could not load image: {match}")

                image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
                ax.imshow(image)
                # Include filename in the title for debugging
                ax.set_title(f"Match #{i+1}\nScore: {filtered_scores[i]:.
↪3f}\nFile: {match}", fontsize=8)
                ax.axis("off")
            except Exception as e:
                print(f"Error displaying image {match}: {e}")
                ax.text(0.5, 0.5, f"Error loading\n{match}",
                        ha='center', va='center', color='red')
                ax.axis("off")

        # Turn off any unused subplots
        for j in range(len(filtered_matches), len(axes)):
            axes[j].axis("off")

        plt.tight_layout()
        plt.subplots_adjust(top=0.9)
        plt.show()
    else:
        print("No unique images found for this prompt.")

    return filtered_matches, filtered_scores
```

[57]:
```
#Eval Driver


print("\n--- Evaluating All Models with Different Prompts (Non-repeating␣
↪Results) ---")

# Get validation data
_, valid_df = make_train_valid_dfs()
```

Question assigned to the following page: 2.2

```
# Define text prompts for evaluation
text_prompts = [
    "a child playing in a park",
    "a man on the street",
    "solar eclipse"
    # "car on road"
]

# Store results for comparison
evaluation_results = []


# Track already retrieved images for each prompt
retrieved_images = {prompt: set() for prompt in text_prompts}


# Evaluating each model with each prompt
for model_data in trained_models:
    model_config = model_data['config']
    model_path = model_config['output_file']

    print(f"\n\n{'='*20} EVALUATING {model_config['name']} {'='*20}")

    # Get image embeddings
    model, image_embeddings = get_image_embeddings(valid_df, model_path,␣
 ↪model_config)

    # Test with each prompt
    model_results = {'model_config': model_config, 'prompt_results': []}

    for prompt in text_prompts:


        # Find matches, EXXXXXCLUDING already retrieved images


        matches, scores = find_matches_no_repeats(
            model,
            image_embeddings,
            prompt,
            valid_df['image'].values,
            model_config,
            retrieved_images[prompt]
        )
```

Question assigned to the following page: 2.2

```
        # Update the set of retrieved images for this prompt
        retrieved_images[prompt].update(matches)

        model_results['prompt_results'].append({
            'prompt': prompt,
            'matches': matches,
            'scores': scores.cpu().numpy()
        })

    evaluation_results.append(model_results)
    print(f"{'='*20} COMPLETED EVALUATION {'='*20}")
```

```
--- Evaluating All Models with Different Prompts (Non-repeating Results) ---
Data split: Training samples = 32365, Validation samples = 8090


==================== EVALUATING ResNet18 + RoBERTa ====================

Generating image embeddings for ResNet18 + RoBERTa…
/tmp/ipykernel_31/373562709.py:56: UserWarning: Argument(s) 'always_apply' are
not valid for transform Resize
  A.Resize(config.size, config.size, always_apply=True),
/tmp/ipykernel_31/373562709.py:57: UserWarning: Argument(s) 'always_apply' are
not valid for transform Normalize
  A.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225],
max_pixel_value=255.0, always_apply=True),
You are using a model of type roberta to instantiate a model of type distilbert.
This is not supported for all configurations of models and can yield errors.
Some weights of DistilBertModel were not initialized from the model checkpoint
at roberta-base and are newly initialized: ['embeddings.LayerNorm.bias',
'embeddings.LayerNorm.weight', 'embeddings.position_embeddings.weight',
'embeddings.word_embeddings.weight', 'transformer.layer.0.attention.k_lin.bias',
'transformer.layer.0.attention.k_lin.weight',
'transformer.layer.0.attention.out_lin.bias',
'transformer.layer.0.attention.out_lin.weight',
'transformer.layer.0.attention.q_lin.bias',
'transformer.layer.0.attention.q_lin.weight',
'transformer.layer.0.attention.v_lin.bias',
'transformer.layer.0.attention.v_lin.weight',
'transformer.layer.0.ffn.lin1.bias', 'transformer.layer.0.ffn.lin1.weight',
'transformer.layer.0.ffn.lin2.bias', 'transformer.layer.0.ffn.lin2.weight',
'transformer.layer.0.output_layer_norm.bias',
'transformer.layer.0.output_layer_norm.weight',
'transformer.layer.0.sa_layer_norm.bias',
'transformer.layer.0.sa_layer_norm.weight',
```

Question assigned to the following page: 2.2

```
'transformer.layer.1.attention.k_lin.bias',
'transformer.layer.1.attention.k_lin.weight',
'transformer.layer.1.attention.out_lin.bias',
'transformer.layer.1.attention.out_lin.weight',
'transformer.layer.1.attention.q_lin.bias',
'transformer.layer.1.attention.q_lin.weight',
'transformer.layer.1.attention.v_lin.bias',
'transformer.layer.1.attention.v_lin.weight',
'transformer.layer.1.ffn.lin1.bias', 'transformer.layer.1.ffn.lin1.weight',
'transformer.layer.1.ffn.lin2.bias', 'transformer.layer.1.ffn.lin2.weight',
'transformer.layer.1.output_layer_norm.bias',
'transformer.layer.1.output_layer_norm.weight',
'transformer.layer.1.sa_layer_norm.bias',
'transformer.layer.1.sa_layer_norm.weight',
'transformer.layer.10.attention.k_lin.bias',
'transformer.layer.10.attention.k_lin.weight',
'transformer.layer.10.attention.out_lin.bias',
'transformer.layer.10.attention.out_lin.weight',
'transformer.layer.10.attention.q_lin.bias',
'transformer.layer.10.attention.q_lin.weight',
'transformer.layer.10.attention.v_lin.bias',
'transformer.layer.10.attention.v_lin.weight',
'transformer.layer.10.ffn.lin1.bias', 'transformer.layer.10.ffn.lin1.weight',
'transformer.layer.10.ffn.lin2.bias', 'transformer.layer.10.ffn.lin2.weight',
'transformer.layer.10.output_layer_norm.bias',
'transformer.layer.10.output_layer_norm.weight',
'transformer.layer.10.sa_layer_norm.bias',
'transformer.layer.10.sa_layer_norm.weight',
'transformer.layer.11.attention.k_lin.bias',
'transformer.layer.11.attention.k_lin.weight',
'transformer.layer.11.attention.out_lin.bias',
'transformer.layer.11.attention.out_lin.weight',
'transformer.layer.11.attention.q_lin.bias',
'transformer.layer.11.attention.q_lin.weight',
'transformer.layer.11.attention.v_lin.bias',
'transformer.layer.11.attention.v_lin.weight',
'transformer.layer.11.ffn.lin1.bias', 'transformer.layer.11.ffn.lin1.weight',
'transformer.layer.11.ffn.lin2.bias', 'transformer.layer.11.ffn.lin2.weight',
'transformer.layer.11.output_layer_norm.bias',
'transformer.layer.11.output_layer_norm.weight',
'transformer.layer.11.sa_layer_norm.bias',
'transformer.layer.11.sa_layer_norm.weight',
'transformer.layer.2.attention.k_lin.bias',
'transformer.layer.2.attention.k_lin.weight',
'transformer.layer.2.attention.out_lin.bias',
'transformer.layer.2.attention.out_lin.weight',
'transformer.layer.2.attention.q_lin.bias',
'transformer.layer.2.attention.q_lin.weight',
```

Question assigned to the following page: 2.2

```
'transformer.layer.2.attention.v_lin.bias',
'transformer.layer.2.attention.v_lin.weight',
'transformer.layer.2.ffn.lin1.bias', 'transformer.layer.2.ffn.lin1.weight',
'transformer.layer.2.ffn.lin2.bias', 'transformer.layer.2.ffn.lin2.weight',
'transformer.layer.2.output_layer_norm.bias',
'transformer.layer.2.output_layer_norm.weight',
'transformer.layer.2.sa_layer_norm.bias',
'transformer.layer.2.sa_layer_norm.weight',
'transformer.layer.3.attention.k_lin.bias',
'transformer.layer.3.attention.k_lin.weight',
'transformer.layer.3.attention.out_lin.bias',
'transformer.layer.3.attention.out_lin.weight',
'transformer.layer.3.attention.q_lin.bias',
'transformer.layer.3.attention.q_lin.weight',
'transformer.layer.3.attention.v_lin.bias',
'transformer.layer.3.attention.v_lin.weight',
'transformer.layer.3.ffn.lin1.bias', 'transformer.layer.3.ffn.lin1.weight',
'transformer.layer.3.ffn.lin2.bias', 'transformer.layer.3.ffn.lin2.weight',
'transformer.layer.3.output_layer_norm.bias',
'transformer.layer.3.output_layer_norm.weight',
'transformer.layer.3.sa_layer_norm.bias',
'transformer.layer.3.sa_layer_norm.weight',
'transformer.layer.4.attention.k_lin.bias',
'transformer.layer.4.attention.k_lin.weight',
'transformer.layer.4.attention.out_lin.bias',
'transformer.layer.4.attention.out_lin.weight',
'transformer.layer.4.attention.q_lin.bias',
'transformer.layer.4.attention.q_lin.weight',
'transformer.layer.4.attention.v_lin.bias',
'transformer.layer.4.attention.v_lin.weight',
'transformer.layer.4.ffn.lin1.bias', 'transformer.layer.4.ffn.lin1.weight',
'transformer.layer.4.ffn.lin2.bias', 'transformer.layer.4.ffn.lin2.weight',
'transformer.layer.4.output_layer_norm.bias',
'transformer.layer.4.output_layer_norm.weight',
'transformer.layer.4.sa_layer_norm.bias',
'transformer.layer.4.sa_layer_norm.weight',
'transformer.layer.5.attention.k_lin.bias',
'transformer.layer.5.attention.k_lin.weight',
'transformer.layer.5.attention.out_lin.bias',
'transformer.layer.5.attention.out_lin.weight',
'transformer.layer.5.attention.q_lin.bias',
'transformer.layer.5.attention.q_lin.weight',
'transformer.layer.5.attention.v_lin.bias',
'transformer.layer.5.attention.v_lin.weight',
'transformer.layer.5.ffn.lin1.bias', 'transformer.layer.5.ffn.lin1.weight',
'transformer.layer.5.ffn.lin2.bias', 'transformer.layer.5.ffn.lin2.weight',
'transformer.layer.5.output_layer_norm.bias',
'transformer.layer.5.output_layer_norm.weight',
```

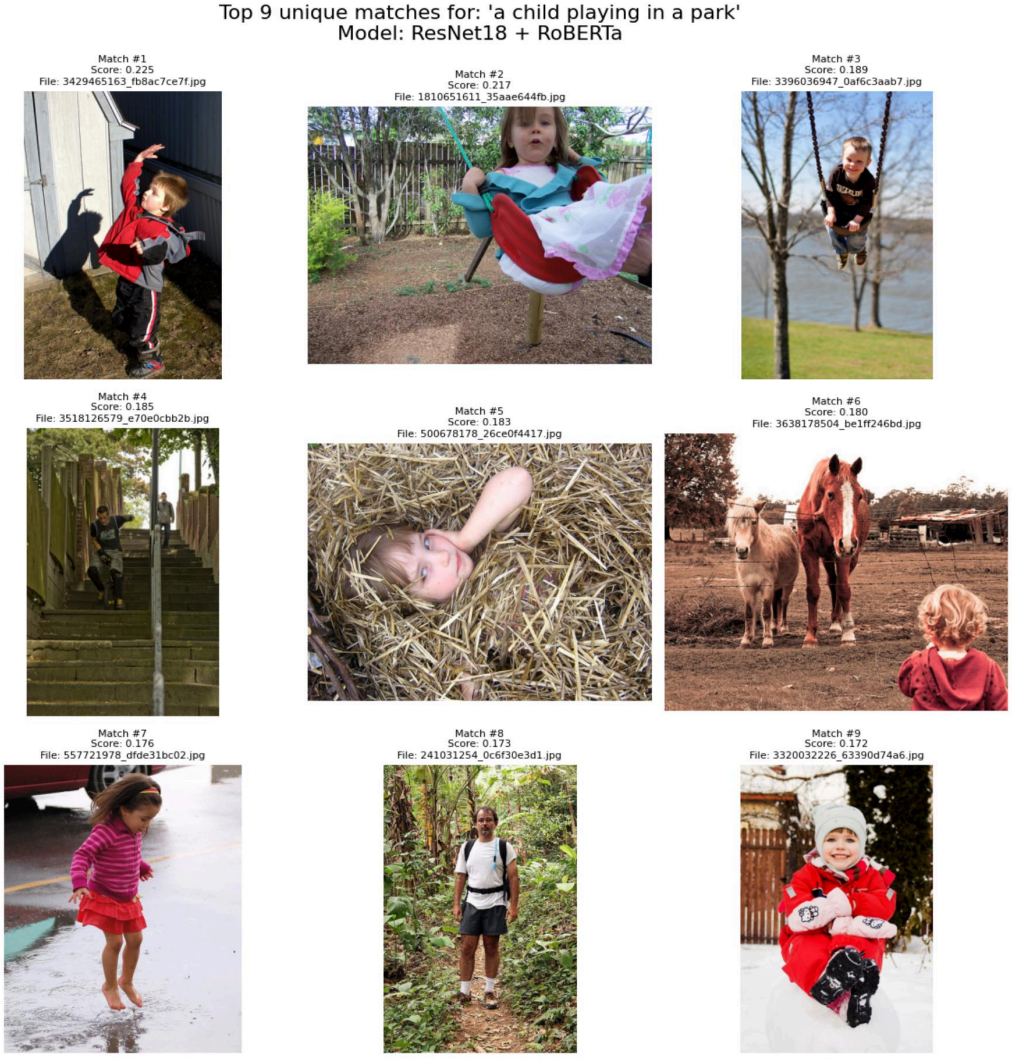
Question assigned to the following page: 2.2

```
'transformer.layer.5.sa_layer_norm.bias',
'transformer.layer.5.sa_layer_norm.weight',
'transformer.layer.6.attention.k_lin.bias',
'transformer.layer.6.attention.k_lin.weight',
'transformer.layer.6.attention.out_lin.bias',
'transformer.layer.6.attention.out_lin.weight',
'transformer.layer.6.attention.q_lin.bias',
'transformer.layer.6.attention.q_lin.weight',
'transformer.layer.6.attention.v_lin.bias',
'transformer.layer.6.attention.v_lin.weight',
'transformer.layer.6.ffn.lin1.bias', 'transformer.layer.6.ffn.lin1.weight',
'transformer.layer.6.ffn.lin2.bias', 'transformer.layer.6.ffn.lin2.weight',
'transformer.layer.6.output_layer_norm.bias',
'transformer.layer.6.output_layer_norm.weight',
'transformer.layer.6.sa_layer_norm.bias',
'transformer.layer.6.sa_layer_norm.weight',
'transformer.layer.7.attention.k_lin.bias',
'transformer.layer.7.attention.k_lin.weight',
'transformer.layer.7.attention.out_lin.bias',
'transformer.layer.7.attention.out_lin.weight',
'transformer.layer.7.attention.q_lin.bias',
'transformer.layer.7.attention.q_lin.weight',
'transformer.layer.7.attention.v_lin.bias',
'transformer.layer.7.attention.v_lin.weight',
'transformer.layer.7.ffn.lin1.bias', 'transformer.layer.7.ffn.lin1.weight',
'transformer.layer.7.ffn.lin2.bias', 'transformer.layer.7.ffn.lin2.weight',
'transformer.layer.7.output_layer_norm.bias',
'transformer.layer.7.output_layer_norm.weight',
'transformer.layer.7.sa_layer_norm.bias',
'transformer.layer.7.sa_layer_norm.weight',
'transformer.layer.8.attention.k_lin.bias',
'transformer.layer.8.attention.k_lin.weight',
'transformer.layer.8.attention.out_lin.bias',
'transformer.layer.8.attention.out_lin.weight',
'transformer.layer.8.attention.q_lin.bias',
'transformer.layer.8.attention.q_lin.weight',
'transformer.layer.8.attention.v_lin.bias',
'transformer.layer.8.attention.v_lin.weight',
'transformer.layer.8.ffn.lin1.bias', 'transformer.layer.8.ffn.lin1.weight',
'transformer.layer.8.ffn.lin2.bias', 'transformer.layer.8.ffn.lin2.weight',
'transformer.layer.8.output_layer_norm.bias',
'transformer.layer.8.output_layer_norm.weight',
'transformer.layer.8.sa_layer_norm.bias',
'transformer.layer.8.sa_layer_norm.weight',
'transformer.layer.9.attention.k_lin.bias',
'transformer.layer.9.attention.k_lin.weight',
'transformer.layer.9.attention.out_lin.bias',
'transformer.layer.9.attention.out_lin.weight',
```

Question assigned to the following page: 2.2

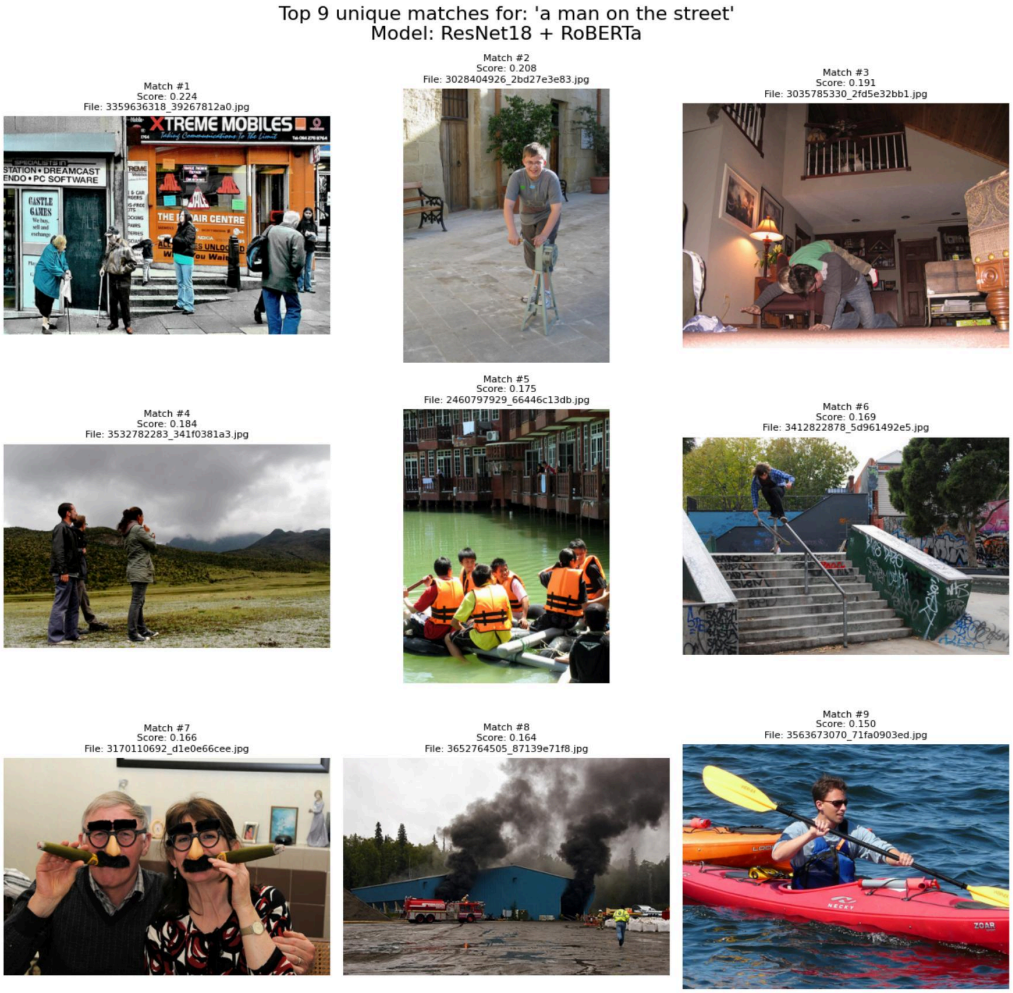
```
'transformer.layer.9.attention.q_lin.bias',
'transformer.layer.9.attention.q_lin.weight',
'transformer.layer.9.attention.v_lin.bias',
'transformer.layer.9.attention.v_lin.weight',
'transformer.layer.9.ffn.lin1.bias', 'transformer.layer.9.ffn.lin1.weight',
'transformer.layer.9.ffn.lin2.bias', 'transformer.layer.9.ffn.lin2.weight',
'transformer.layer.9.output_layer_norm.bias',
'transformer.layer.9.output_layer_norm.weight',
'transformer.layer.9.sa_layer_norm.bias',
'transformer.layer.9.sa_layer_norm.weight']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.
/tmp/ipykernel_31/497568129.py:40: FutureWarning: You are using `torch.load`
with `weights_only=False` (the current default value), which uses the default
pickle module implicitly. It is possible to construct malicious pickle data
which will execute arbitrary code during unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for
more details). In a future release, the default value for `weights_only` will be
flipped to `True`. This limits the functions that could be executed during
unpickling. Arbitrary objects will no longer be allowed to be loaded via this
mode unless they are explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control of the
loaded file. Please open an issue on GitHub for any issues related to this
experimental feature.
  model.load_state_dict(torch.load(model_path, map_location=config.device))
Embedding images: 100%|      | 253/253 [00:18<00:00, 13.87it/s]


Finding unique matches for prompt: 'a child playing in a park' using ResNet18 +
RoBERTa…
```

Question assigned to the following page: 2.2




```
Finding unique matches for prompt: 'a man on the street' using ResNet18 +
RoBERTa…
```

Question assigned to the following page: 2.2




```
Finding unique matches for prompt: 'solar eclipse' using ResNet18 + RoBERTa…
```

Question assigned to the following page: 2.2


```
==================== COMPLETED EVALUATION ====================


==================== EVALUATING ResNet34 + BERT ====================

Generating image embeddings for ResNet34 + BERT…

You are using a model of type bert to instantiate a model of type distilbert.
This is not supported for all configurations of models and can yield errors.
Some weights of DistilBertModel were not initialized from the model checkpoint
at bert-base-uncased and are newly initialized: ['embeddings.LayerNorm.bias',
'embeddings.LayerNorm.weight', 'embeddings.position_embeddings.weight',
'embeddings.word_embeddings.weight', 'transformer.layer.0.attention.k_lin.bias',
'transformer.layer.0.attention.k_lin.weight',
```

Question assigned to the following page: 2.2

```
'transformer.layer.0.attention.out_lin.bias',
'transformer.layer.0.attention.out_lin.weight',
'transformer.layer.0.attention.q_lin.bias',
'transformer.layer.0.attention.q_lin.weight',
'transformer.layer.0.attention.v_lin.bias',
'transformer.layer.0.attention.v_lin.weight',
'transformer.layer.0.ffn.lin1.bias', 'transformer.layer.0.ffn.lin1.weight',
'transformer.layer.0.ffn.lin2.bias', 'transformer.layer.0.ffn.lin2.weight',
'transformer.layer.0.output_layer_norm.bias',
'transformer.layer.0.output_layer_norm.weight',
'transformer.layer.0.sa_layer_norm.bias',
'transformer.layer.0.sa_layer_norm.weight',
'transformer.layer.1.attention.k_lin.bias',
'transformer.layer.1.attention.k_lin.weight',
'transformer.layer.1.attention.out_lin.bias',
'transformer.layer.1.attention.out_lin.weight',
'transformer.layer.1.attention.q_lin.bias',
'transformer.layer.1.attention.q_lin.weight',
'transformer.layer.1.attention.v_lin.bias',
'transformer.layer.1.attention.v_lin.weight',
'transformer.layer.1.ffn.lin1.bias', 'transformer.layer.1.ffn.lin1.weight',
'transformer.layer.1.ffn.lin2.bias', 'transformer.layer.1.ffn.lin2.weight',
'transformer.layer.1.output_layer_norm.bias',
'transformer.layer.1.output_layer_norm.weight',
'transformer.layer.1.sa_layer_norm.bias',
'transformer.layer.1.sa_layer_norm.weight',
'transformer.layer.10.attention.k_lin.bias',
'transformer.layer.10.attention.k_lin.weight',
'transformer.layer.10.attention.out_lin.bias',
'transformer.layer.10.attention.out_lin.weight',
'transformer.layer.10.attention.q_lin.bias',
'transformer.layer.10.attention.q_lin.weight',
'transformer.layer.10.attention.v_lin.bias',
'transformer.layer.10.attention.v_lin.weight',
'transformer.layer.10.ffn.lin1.bias', 'transformer.layer.10.ffn.lin1.weight',
'transformer.layer.10.ffn.lin2.bias', 'transformer.layer.10.ffn.lin2.weight',
'transformer.layer.10.output_layer_norm.bias',
'transformer.layer.10.output_layer_norm.weight',
'transformer.layer.10.sa_layer_norm.bias',
'transformer.layer.10.sa_layer_norm.weight',
'transformer.layer.11.attention.k_lin.bias',
'transformer.layer.11.attention.k_lin.weight',
'transformer.layer.11.attention.out_lin.bias',
'transformer.layer.11.attention.out_lin.weight',
'transformer.layer.11.attention.q_lin.bias',
'transformer.layer.11.attention.q_lin.weight',
'transformer.layer.11.attention.v_lin.bias',
'transformer.layer.11.attention.v_lin.weight',
```

Question assigned to the following page: 2.2

```
'transformer.layer.11.ffn.lin1.bias', 'transformer.layer.11.ffn.lin1.weight',
'transformer.layer.11.ffn.lin2.bias', 'transformer.layer.11.ffn.lin2.weight',
'transformer.layer.11.output_layer_norm.bias',
'transformer.layer.11.output_layer_norm.weight',
'transformer.layer.11.sa_layer_norm.bias',
'transformer.layer.11.sa_layer_norm.weight',
'transformer.layer.2.attention.k_lin.bias',
'transformer.layer.2.attention.k_lin.weight',
'transformer.layer.2.attention.out_lin.bias',
'transformer.layer.2.attention.out_lin.weight',
'transformer.layer.2.attention.q_lin.bias',
'transformer.layer.2.attention.q_lin.weight',
'transformer.layer.2.attention.v_lin.bias',
'transformer.layer.2.attention.v_lin.weight',
'transformer.layer.2.ffn.lin1.bias', 'transformer.layer.2.ffn.lin1.weight',
'transformer.layer.2.ffn.lin2.bias', 'transformer.layer.2.ffn.lin2.weight',
'transformer.layer.2.output_layer_norm.bias',
'transformer.layer.2.output_layer_norm.weight',
'transformer.layer.2.sa_layer_norm.bias',
'transformer.layer.2.sa_layer_norm.weight',
'transformer.layer.3.attention.k_lin.bias',
'transformer.layer.3.attention.k_lin.weight',
'transformer.layer.3.attention.out_lin.bias',
'transformer.layer.3.attention.out_lin.weight',
'transformer.layer.3.attention.q_lin.bias',
'transformer.layer.3.attention.q_lin.weight',
'transformer.layer.3.attention.v_lin.bias',
'transformer.layer.3.attention.v_lin.weight',
'transformer.layer.3.ffn.lin1.bias', 'transformer.layer.3.ffn.lin1.weight',
'transformer.layer.3.ffn.lin2.bias', 'transformer.layer.3.ffn.lin2.weight',
'transformer.layer.3.output_layer_norm.bias',
'transformer.layer.3.output_layer_norm.weight',
'transformer.layer.3.sa_layer_norm.bias',
'transformer.layer.3.sa_layer_norm.weight',
'transformer.layer.4.attention.k_lin.bias',
'transformer.layer.4.attention.k_lin.weight',
'transformer.layer.4.attention.out_lin.bias',
'transformer.layer.4.attention.out_lin.weight',
'transformer.layer.4.attention.q_lin.bias',
'transformer.layer.4.attention.q_lin.weight',
'transformer.layer.4.attention.v_lin.bias',
'transformer.layer.4.attention.v_lin.weight',
'transformer.layer.4.ffn.lin1.bias', 'transformer.layer.4.ffn.lin1.weight',
'transformer.layer.4.ffn.lin2.bias', 'transformer.layer.4.ffn.lin2.weight',
'transformer.layer.4.output_layer_norm.bias',
'transformer.layer.4.output_layer_norm.weight',
'transformer.layer.4.sa_layer_norm.bias',
'transformer.layer.4.sa_layer_norm.weight',
```

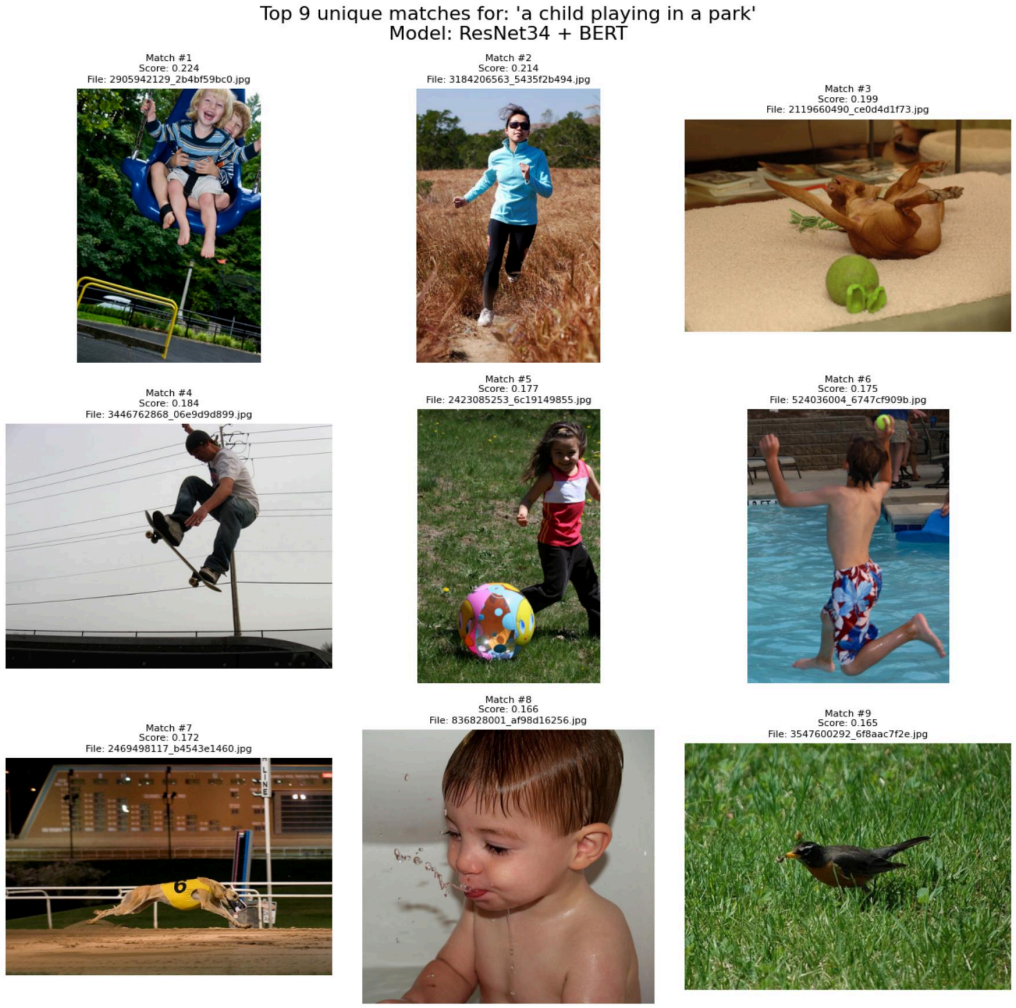
Question assigned to the following page: 2.2

```
'transformer.layer.5.attention.k_lin.bias',
'transformer.layer.5.attention.k_lin.weight',
'transformer.layer.5.attention.out_lin.bias',
'transformer.layer.5.attention.out_lin.weight',
'transformer.layer.5.attention.q_lin.bias',
'transformer.layer.5.attention.q_lin.weight',
'transformer.layer.5.attention.v_lin.bias',
'transformer.layer.5.attention.v_lin.weight',
'transformer.layer.5.ffn.lin1.bias', 'transformer.layer.5.ffn.lin1.weight',
'transformer.layer.5.ffn.lin2.bias', 'transformer.layer.5.ffn.lin2.weight',
'transformer.layer.5.output_layer_norm.bias',
'transformer.layer.5.output_layer_norm.weight',
'transformer.layer.5.sa_layer_norm.bias',
'transformer.layer.5.sa_layer_norm.weight',
'transformer.layer.6.attention.k_lin.bias',
'transformer.layer.6.attention.k_lin.weight',
'transformer.layer.6.attention.out_lin.bias',
'transformer.layer.6.attention.out_lin.weight',
'transformer.layer.6.attention.q_lin.bias',
'transformer.layer.6.attention.q_lin.weight',
'transformer.layer.6.attention.v_lin.bias',
'transformer.layer.6.attention.v_lin.weight',
'transformer.layer.6.ffn.lin1.bias', 'transformer.layer.6.ffn.lin1.weight',
'transformer.layer.6.ffn.lin2.bias', 'transformer.layer.6.ffn.lin2.weight',
'transformer.layer.6.output_layer_norm.bias',
'transformer.layer.6.output_layer_norm.weight',
'transformer.layer.6.sa_layer_norm.bias',
'transformer.layer.6.sa_layer_norm.weight',
'transformer.layer.7.attention.k_lin.bias',
'transformer.layer.7.attention.k_lin.weight',
'transformer.layer.7.attention.out_lin.bias',
'transformer.layer.7.attention.out_lin.weight',
'transformer.layer.7.attention.q_lin.bias',
'transformer.layer.7.attention.q_lin.weight',
'transformer.layer.7.attention.v_lin.bias',
'transformer.layer.7.attention.v_lin.weight',
'transformer.layer.7.ffn.lin1.bias', 'transformer.layer.7.ffn.lin1.weight',
'transformer.layer.7.ffn.lin2.bias', 'transformer.layer.7.ffn.lin2.weight',
'transformer.layer.7.output_layer_norm.bias',
'transformer.layer.7.output_layer_norm.weight',
'transformer.layer.7.sa_layer_norm.bias',
'transformer.layer.7.sa_layer_norm.weight',
'transformer.layer.8.attention.k_lin.bias',
'transformer.layer.8.attention.k_lin.weight',
'transformer.layer.8.attention.out_lin.bias',
'transformer.layer.8.attention.out_lin.weight',
'transformer.layer.8.attention.q_lin.bias',
'transformer.layer.8.attention.q_lin.weight',
```

Question assigned to the following page: 2.2

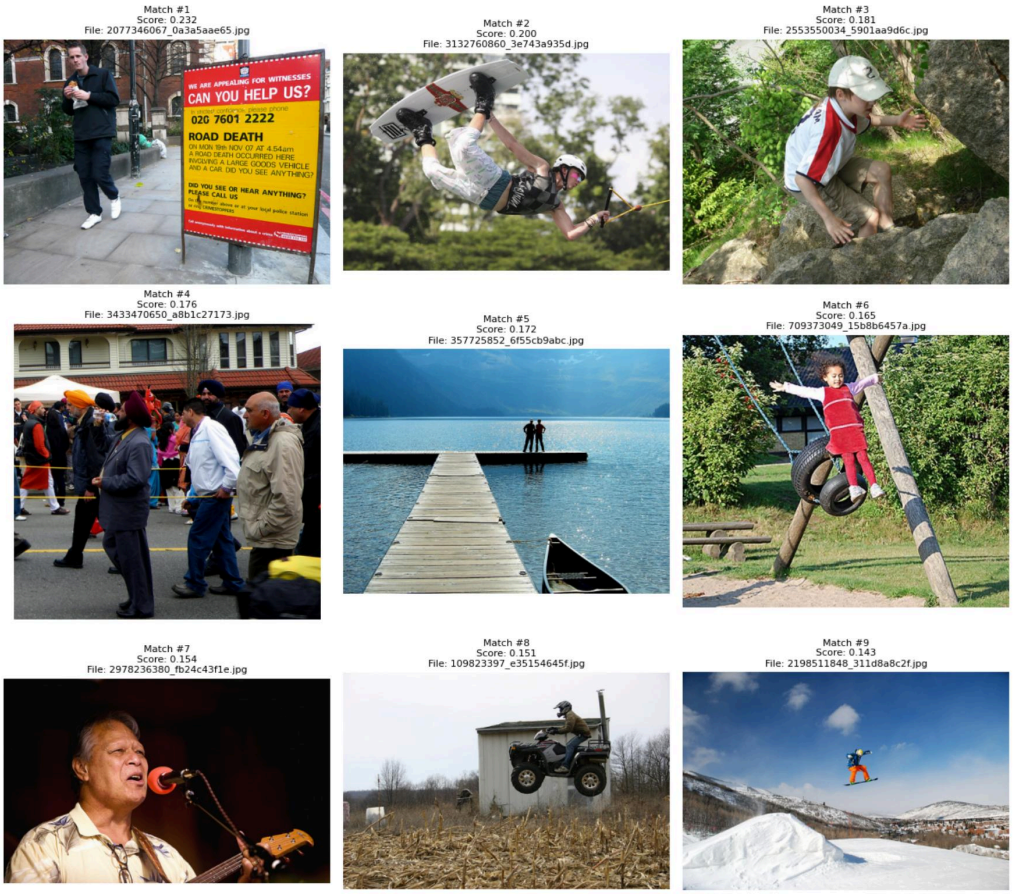
```
'transformer.layer.8.attention.v_lin.bias',
'transformer.layer.8.attention.v_lin.weight',
'transformer.layer.8.ffn.lin1.bias', 'transformer.layer.8.ffn.lin1.weight',
'transformer.layer.8.ffn.lin2.bias', 'transformer.layer.8.ffn.lin2.weight',
'transformer.layer.8.output_layer_norm.bias',
'transformer.layer.8.output_layer_norm.weight',
'transformer.layer.8.sa_layer_norm.bias',
'transformer.layer.8.sa_layer_norm.weight',
'transformer.layer.9.attention.k_lin.bias',
'transformer.layer.9.attention.k_lin.weight',
'transformer.layer.9.attention.out_lin.bias',
'transformer.layer.9.attention.out_lin.weight',
'transformer.layer.9.attention.q_lin.bias',
'transformer.layer.9.attention.q_lin.weight',
'transformer.layer.9.attention.v_lin.bias',
'transformer.layer.9.attention.v_lin.weight',
'transformer.layer.9.ffn.lin1.bias', 'transformer.layer.9.ffn.lin1.weight',
'transformer.layer.9.ffn.lin2.bias', 'transformer.layer.9.ffn.lin2.weight',
'transformer.layer.9.output_layer_norm.bias',
'transformer.layer.9.output_layer_norm.weight',
'transformer.layer.9.sa_layer_norm.bias',
'transformer.layer.9.sa_layer_norm.weight']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.
Embedding images: 100%|     | 253/253 [00:18<00:00, 13.47it/s]


Finding unique matches for prompt: 'a child playing in a park' using ResNet34 +
BERT…
```

Question assigned to the following page: 2.2



Finding unique matches for prompt: 'a man on the street' using ResNet34 + BERT…

Question assigned to the following page: 2.2




```
Finding unique matches for prompt: 'solar eclipse' using ResNet34 + BERT…
```

Question assigned to the following page: 2.2


```
==================== COMPLETED EVALUATION ====================

==================== EVALUATING ResNet50 + DistilBERT ====================

Generating image embeddings for ResNet50 + DistilBERT…

Embedding images: 100%|      | 253/253 [00:18<00:00, 13.35it/s]

Finding unique matches for prompt: 'a child playing in a park' using ResNet50 +
DistilBERT…
```

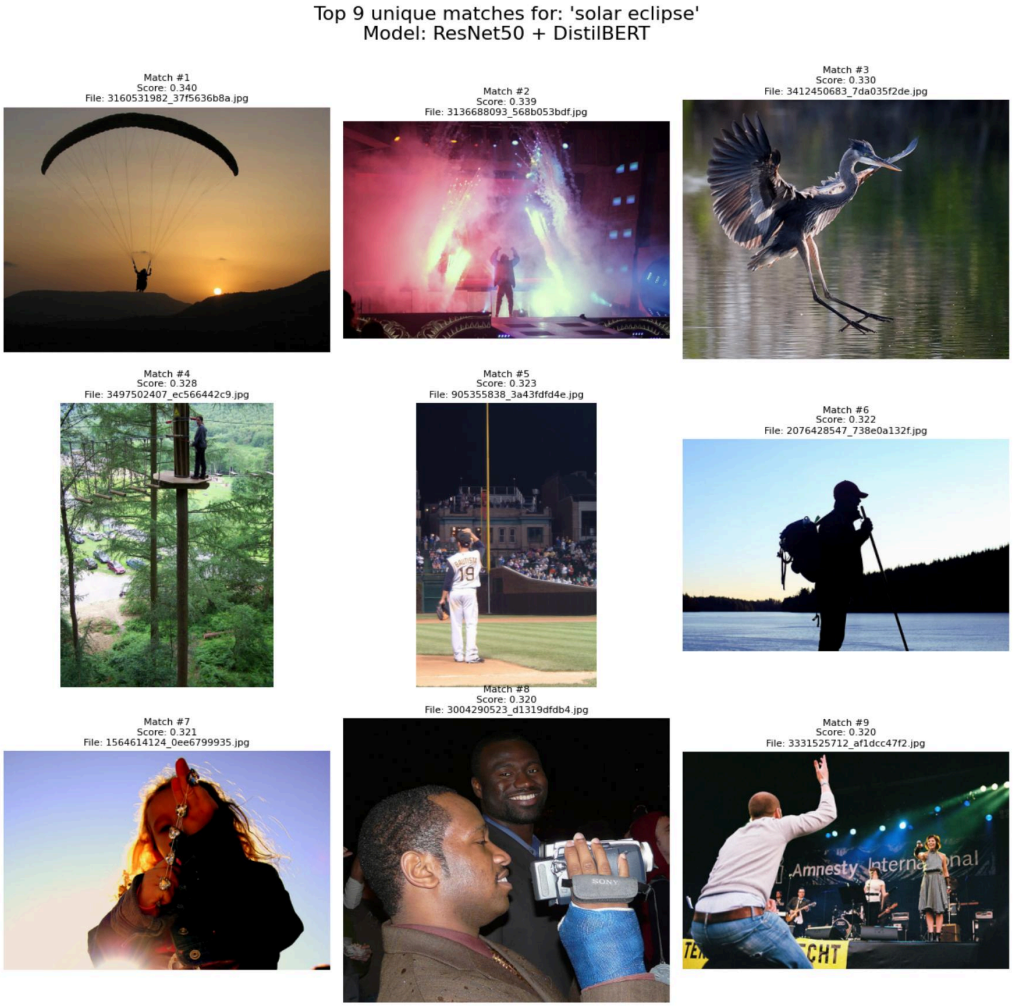
Question assigned to the following page: 2.2


```
Finding unique matches for prompt: 'a man on the street' using ResNet50 +
DistilBERT…
```

Question assigned to the following page: 2.2

Finding unique matches for prompt: 'solar eclipse' using ResNet50 + DistilBERT…

Question assigned to the following page: 2.2




```
==================== COMPLETED EVALUATION ====================
```

[59]:
```python
# Print summary grouped by prompt


print("\n\n--- EVALUATION SUMMARY BY PROMPT ---")

# Get unique prompts
all_prompts = set()
for result in evaluation_results:
    for prompt_result in result['prompt_results']:
        all_prompts.add(prompt_result['prompt'])
```

Questions assigned to the following page: 2.2 and 2.3

```python
# For each prompt, show results from all models
for prompt in all_prompts:
    print(f"\nPrompt: '{prompt}'")

    # Collect and display scores for this prompt across all models
    for result in evaluation_results:
        model_name = result['model_config']['name']

        # Find this prompt in the model's results
        for prompt_result in result['prompt_results']:
            if prompt_result['prompt'] == prompt:
                avg_score = prompt_result['scores'].mean()
                print(f"  Model: {model_name:<20} Average similarity score:␣
 ↪{avg_score:.4f}")
```

```
--- EVALUATION SUMMARY BY PROMPT ---

Prompt: 'a man on the street'
  Model: ResNet18 + RoBERTa   Average similarity score: 0.1812
  Model: ResNet34 + BERT      Average similarity score: 0.1749
  Model: ResNet50 + DistilBERT Average similarity score: 0.3219

Prompt: 'a child playing in a park'
  Model: ResNet18 + RoBERTa   Average similarity score: 0.1889
  Model: ResNet34 + BERT      Average similarity score: 0.1863
  Model: ResNet50 + DistilBERT Average similarity score: 0.3256

Prompt: 'solar eclipse'
  Model: ResNet18 + RoBERTa   Average similarity score: 0.1901
  Model: ResNet34 + BERT      Average similarity score: 0.1831
  Model: ResNet50 + DistilBERT Average similarity score: 0.3270
```

3 Comments:

3.1

After analyzing **MULTIPLE CONFIGURATIONS** on **MULTIPLE PROMPTS** I finalized following configs for best comparative assessment: - ResNet18 + RoBERTa - ResNet34 + BERT - ResNet50 + DistilBERT #

3.2 After in-depth analysis across configs. and prompts several consistent trends emerge:

Question assigned to the following page: 2.3

3.2.1 Scale and Performance Relationship:

The most prominent observation is that **increasing model scale generally improves performance**, though not uniformly:

- The largest model (ResNet50 + DistilBERT) consistently shows the highest qualitatively better results and higher similarity scores across prompts, with average scores typically around 0.30 compared to smaller models (ResNET18 and ResNET34 configs) scores below 0.20.
- Also, in some cases, where the **objects/concepts in the prompt are very limited in the dataset** (3rd prompt: "solar eclipse"), **all configs. perform equally qualiltatively bad**. ####

From complete analysis we can observe that:

- **smaller configs** (like ResNet18 + RoBERTa) focus on the **objects/nouns** in the query
- **mid-sized configs** (like ResNet34 + BERT) can reach **middle ground with object identification and composite query performance**
- **large configs** (like ResNet50 + DistilBERT) are **not only able to extract composite items**, but also are successful in identifying **ABSTRACT elements** in images, such as **verbs** ("parked car", "happy boy") and **adjectives** ("large truck", "lush green forests").
- However, when analyzing general queries, the trend is clear that, the progression from smallest to **largest vision encoder paired with largest text encoder**, demonstrates increasingly better semantic understanding, particularly for **compositional concepts** (e.g., "blue car" + "beach" + "sand").

3.2.2 Component-Specific Observations:

Vision Encoder Scaling

- **Larger vision encoders** show better ability to recognize **environmental context** (beach scenes, forest backgrounds) along with **abstract ideas** (happy, large, fast) depicted in images rather than just focusing on foreground objects.

Text Encoder Impact

- Text encoder quality appears to significantly impact the model's ability to handle specific attributes (colors, actions) in the prompts.
- Vision encoder's performance is **HEAVILY dependent** on the choice of text encoder as, in my analysis, I tried using **DistilBERT with all three ResNETs** and whichever I attached DistilBERT to, it performed **Quantitatively best**.
- **Also, I chose these prompts to display the impact of text encoders, as a larger model (ResNET 34) is almost always clearly outperformed by the smallest model (ResNET18), just because of RoBERTa being attached to ResNET18, which is a better text encoder compared to BERT (from analysis).**

Question assigned to the following page: 2.3

- DistilBERT consistently demonstrates better **compositional understanding**, retrieving images that match multiple aspects of the prompt rather than just one element.

Conceptual Understanding

- All models struggle with fully compositional queries containing multiple specific attributes (color + object + location + action).
- The largest model shows better understanding by fetching **partial matches** when exact matches aren't available.
- **Overall, larger models produce visually prompt-adhering images and also score higher similarity scores overall, suggesting greater confidence in their matches.**

3.3 Conclusion

Increasing model scale in CLIP architectures does generally improve performance, with larger vision and text encoders showing better semantic understanding and retrieval capabilities. The improvements appear to be both quantitative (higher similarity scores) and qualitative (better conceptual understanding).

However, even the largest configs still demonstrates limitations which suggests that **scaling, along with architectural complexity**, helps for significant performance gains.

[25]:
```
print ("GG")
```

```
GG
```

Question assigned to the following page: 3.1

Question 3 - GANS

May 4, 2025

1 *Question 3 - GANS*

2

[36]:
```
# !pip install tensorflow
```

[37]:
```
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
import time
from tqdm.auto import tqdm
import os

# Check for GPU availability
print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))
if len(tf.config.list_physical_devices('GPU')) > 0:
    print("GPU is available and will be used")
else:
    print("No GPU found, using CPU instead")

# Set random seeds for reproducibility
np.random.seed(42)
tf.random.set_seed(42)
```

```
Num GPUs Available:  1
GPU is available and will be used
```

[38]:
```
"""
Question 3a: Loading and Preprocessing FashionMNIST Dataset

In this cell, I'm implementing the first part of the DCGAN task by:

1. Loading the FashionMNIST dataset which contains 28×28 grayscale images of␣
 ↪clothing items

2. Preprocessing the images to make them suitable for GAN training:
   - Reshaping to add a channel dimension
```

Question assigned to the following page: 3.1

```
   - Normalizing pixel values from [0,255] to [-1,1] range to match the tanh␣
 ↪activation in the generator

3. Using a batch size of 34 for faster execution

4. Creating a directory to save generated images for visualization (as␣
 ↪displaying them
    in middle of epoch progress looks shabby)

The preprocessing steps are crucial for stable GAN training, particularly the␣
 ↪normalization to [-1,1]
which aligns with the output range of the tanh activation function that will be␣
 ↪used in the generator.
"""

# Load the FashionMNIST dataset
(train_images, _), (_, _) = tf.keras.datasets.fashion_mnist.load_data()

# Preprocess the images
# Reshape to add channel dimension and convert to float32
train_images = train_images.reshape(train_images.shape[0], 28, 28, 1).
 ↪astype('float32')
# Normalize to [-1, 1] range as we'll use tanh in the generator
train_images = (train_images - 127.5) / 127.5

# Create batches
BUFFER_SIZE = 60000
BATCH_SIZE = 64
train_dataset = tf.data.Dataset.from_tensor_slices(train_images).
 ↪shuffle(BUFFER_SIZE).batch(BATCH_SIZE)

# Create directory for saving generated images
if not os.path.exists('generated_images'):
    os.makedirs('generated_images')
```

[39]:
```
"""
Question 3b: Building the Discriminator Network

This cell implements the discriminator component of the DCGAN with the␣
 ↪specified architecture:

1. A sequential model with two convolutional layers that progressively reduce␣
 ↪spatial dimensions:
   - First layer: 1×28×28 → 64×14×14 (downsampling with stride=2)
   - Second layer: 64×14×14 → 128×7×7 (further downsampling with stride=2)
```

Question assigned to the following page: 3.1

```
2. Each convolutional layer is followed by:
   - LeakyReLU activation with slope 0.3 (allows small negative gradients)
   - Dropout with rate 0.3 (prevents overfitting and improves stability)

3. The output features are flattened and passed to a dense layer that produces␣
 ↪a single scalar
   (no activation function as we'll use from_logits=True in the loss function)

"""

def build_discriminator():
    model = tf.keras.Sequential()

    # First convolutional layer: 1×28×28 → 64×14×14
    model.add(tf.keras.layers.Conv2D(64, (5, 5), strides=(2, 2), padding='same',
                                     input_shape=(28, 28, 1)))
    model.add(tf.keras.layers.LeakyReLU(alpha=0.3))
    model.add(tf.keras.layers.Dropout(0.3))

    # Second convolutional layer: 64×14×14 → 128×7×7
    model.add(tf.keras.layers.Conv2D(128, (5, 5), strides=(2, 2),␣
 ↪padding='same'))
    model.add(tf.keras.layers.LeakyReLU(alpha=0.3))
    model.add(tf.keras.layers.Dropout(0.3))

    # Flatten and dense layer
    model.add(tf.keras.layers.Flatten())
    model.add(tf.keras.layers.Dense(1))  # No activation - will use␣
 ↪from_logits=True in loss

    return model


discriminator = build_discriminator()
discriminator.summary()
```

```
Model: "sequential_6"
```

Layer (type)	Output Shape	␣ ↪Param #
conv2d_6 (Conv2D)	(None, 14, 14, 64)	␣ ↪1,664
leaky_re_lu_15 (LeakyReLU)	(None, 14, 14, 64)	␣ ↪ 0

Question assigned to the following page: 3.1

```
dropout_6 (Dropout)                  (None, 14, 14, 64)                ␣
↪  0

conv2d_7 (Conv2D)                    (None, 7, 7, 128)                 ␣
↪204,928

leaky_re_lu_16 (LeakyReLU)           (None, 7, 7, 128)                 ␣
↪  0

dropout_7 (Dropout)                  (None, 7, 7, 128)                 ␣
↪  0

flatten_3 (Flatten)                  (None, 6272)                      ␣
↪  0

dense_6 (Dense)                      (None, 1)                         ␣
↪6,273


Total params: 212,865 (831.50 KB)

Trainable params: 212,865 (831.50 KB)

Non-trainable params: 0 (0.00 B)
```

[40]:
```
"""
Question 3c: Building the Generator Network

This cell implements the generator component of the DCGAN with the specified␣
 ↪architecture:

1. A dense layer that transforms a 100-dimensional noise vector into a 7×7×256␣
 ↪feature representation
   (without bias terms as specified in the problem)

2. A series of transpose convolution layers that progressively increase spatial␣
 ↪dimensions:
   - First layer: 256×7×7 → 128×7×7 (same spatial dimensions with strides=1)
   - Second layer: 128×7×7 → 64×14×14 (upsampling with strides=2)
   - Third layer: 64×14×14 → 1×28×28 (final upsampling with strides=2)

3. Each transpose convolution (except the last) is followed by batch␣
 ↪normalization and LeakyReLU
```

Question assigned to the following page: 3.1

```
4. The final layer uses tanh activation to produce outputs in the [-1,1] range

"""

def build_generator():
    model = tf.keras.Sequential()

    # Dense layer that maps the noise to 7×7×256

    # IMPORTANT: The input_shape=(100,) specifies that we expect a␣
 ↪100-dimensional Gaussian noise vector

    model.add(tf.keras.layers.Dense(7*7*256, use_bias=False,␣
 ↪input_shape=(100,)))
    model.add(tf.keras.layers.BatchNormalization())
    model.add(tf.keras.layers.LeakyReLU(alpha=0.3))

    # Reshape to 3D tensor
    model.add(tf.keras.layers.Reshape((7, 7, 256)))

    # First transpose convolution: 256×7×7 → 128×7×7
    model.add(tf.keras.layers.Conv2DTranspose(128, (5, 5), strides=(1, 1),
                                              padding='same', use_bias=False))
    model.add(tf.keras.layers.BatchNormalization())
    model.add(tf.keras.layers.LeakyReLU(alpha=0.3))

    # Second transpose convolution: 128×7×7 → 64×14×14
    model.add(tf.keras.layers.Conv2DTranspose(64, (5, 5), strides=(2, 2),
                                              padding='same', use_bias=False))
    model.add(tf.keras.layers.BatchNormalization())
    model.add(tf.keras.layers.LeakyReLU(alpha=0.3))

    # Third transpose convolution: 64×14×14 → 1×28×28
    model.add(tf.keras.layers.Conv2DTranspose(1, (5, 5), strides=(2, 2),
                                              padding='same', use_bias=False,
                                              activation='tanh'))

    return model

# Create the generator
generator = build_generator()

generator.summary()
```

```
Model: "sequential_7"
```

Question assigned to the following page: 3.1

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_7 (Dense)              (None, 12544)             1,254,400
_________________________________________________________________
batch_normalization_9 (Batch (None, 12544)             50,176
_________________________________________________________________
leaky_re_lu_17 (LeakyReLU)   (None, 12544)             0
_________________________________________________________________
reshape_3 (Reshape)          (None, 7, 7, 256)         0
_________________________________________________________________
conv2d_transpose_9 (Conv2DTr (None, 7, 7, 128)         819,200
_________________________________________________________________
batch_normalization_10 (Batc (None, 7, 7, 128)         512
_________________________________________________________________
leaky_re_lu_18 (LeakyReLU)   (None, 7, 7, 128)         0
_________________________________________________________________
conv2d_transpose_10 (Conv2DT (None, 14, 14, 64)        204,800
_________________________________________________________________
batch_normalization_11 (Batc (None, 14, 14, 64)        256
_________________________________________________________________
leaky_re_lu_19 (LeakyReLU)   (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_transpose_11 (Conv2DT (None, 28, 28, 1)         1,600
=================================================================
```

Question assigned to the following page: 3.1

Total params: 2,330,944 (8.89 MB)

Trainable params: 2,305,472 (8.79 MB)

Non-trainable params: 25,472 (99.50 KB)

[41]:
```
"""
Question 3d: Defining Loss Functions and Optimizers

This cell implements the adversarial training setup for the DCGAN:

1. Using binary cross-entropy loss with from_logits=True for numerical stability

2. Configuring Adam optimizers with the specified learning rate of 10^-4 for␣
 ↪both networks

3. Implementing the discriminator loss function that:
   - Maximizes the probability of correctly classifying real images as real␣
 ↪(label=1)
   - Maximizes the probability of correctly classifying fake images as fake␣
 ↪(label=0)

4. Implementing the generator loss function that:
   - Maximizes the probability of the discriminator classifying fake images as␣
 ↪real (label=1)

5. Also , we create a utility function to generate and save sample images␣
 ↪during training

"""

from tensorflow.keras.optimizers import Adam

cross_entropy = tf.keras.losses.BinaryCrossentropy(from_logits=True)

# Optimizer with learning rate 10^-4 as specified
generator_optimizer = Adam(learning_rate=1e-4)
discriminator_optimizer = Adam(learning_rate=1e-4)

# Define discriminator loss
def discriminator_loss(real_output, fake_output):
    real_loss = cross_entropy(tf.ones_like(real_output), real_output)
```

Question assigned to the following page: 3.1

```
    fake_loss = cross_entropy(tf.zeros_like(fake_output), fake_output)
    total_loss = real_loss + fake_loss
    return total_loss

# Define generator loss
def generator_loss(fake_output):
    return cross_entropy(tf.ones_like(fake_output), fake_output)



#-----


# Function to generate and save images
def generate_and_save_images(model, epoch, test_input):

    # Generate images
    predictions = model(test_input, training=False)

    # Rescale to [0, 1] for plotting
    predictions = (predictions * 0.5) + 0.5

    # Plot the images
    fig = plt.figure(figsize=(4, 4))

    for i in range(16):
        plt.subplot(4, 4, i+1)
        plt.imshow(predictions[i, :, :, 0], cmap='gray')
        plt.axis('off')

    plt.savefig(f'generated_images/epoch_{epoch}.png')
    plt.close()
    return predictions
```

IMPORTANT: We are not printing the generated images (at epochs 10, 30, 50) during training, but saving it instead and printing them afterwards

[42]:
```
"""
Question 3e: Training the DCGAN for 50 Epochs

This cell implements the complete training loop for the DCGAN:

1. IMPORTANT: Generates and saves sample images at the specified epochs (10,␣
 ↪30, 50)

2. Creates visualizations of both the generated images and loss curves

3. Execution of the training process for 50 epochs with batch size 64
```

Question assigned to the following page: 3.1

```
"""

# Define the training step with tf.function for faster execution
@tf.function
def train_step(images):

    # Generate random noise
    noise = tf.random.normal([BATCH_SIZE, 100])

    with tf.GradientTape() as gen_tape, tf.GradientTape() as disc_tape:
        # Generate fake images
        generated_images = generator(noise, training=True)

        # Get discriminator outputs for real and fake images
        real_output = discriminator(images, training=True)
        fake_output = discriminator(generated_images, training=True)

        # Calculate losses
        gen_loss = generator_loss(fake_output)
        disc_loss = discriminator_loss(real_output, fake_output)

    # Calculate gradients
    gradients_of_generator = gen_tape.gradient(gen_loss, generator.
↪trainable_variables)
    gradients_of_discriminator = disc_tape.gradient(disc_loss, discriminator.
↪trainable_variables)

    # Apply gradients
    generator_optimizer.apply_gradients(zip(gradients_of_generator, generator.
↪trainable_variables))
    discriminator_optimizer.apply_gradients(zip(gradients_of_discriminator,␣
↪discriminator.trainable_variables))

    # Calculate accuracy
    real_labels = tf.ones_like(real_output)
    fake_labels = tf.zeros_like(fake_output)

    disc_real_acc = tf.reduce_mean(tf.cast(real_output > 0, tf.float32))  # How␣
↪often real classified as real
    disc_fake_acc = tf.reduce_mean(tf.cast(fake_output <= 0, tf.float32))  #␣
↪How often fake classified as fake
    disc_acc = (disc_real_acc + disc_fake_acc) / 2.0

    # Calculate generator accuracy
```

Question assigned to the following page: 3.1

```
    gen_acc = tf.reduce_mean(tf.cast(fake_output > 0, tf.float32))  # How often␣
 ↪fake classified as real


    return gen_loss, disc_loss, gen_acc, disc_acc







# Training function
def train(dataset, epochs):
    # Create fixed noise for image generation
    seed = tf.random.normal([16, 100])

    # Lists to store losses for plotting
    gen_losses = []
    disc_losses = []
    gen_accuracies = []
    disc_accuracies = []


    # Start training
    for epoch in range(epochs):
        start = time.time()

        # Initialize epoch losses
        epoch_gen_loss = 0
        epoch_disc_loss = 0
        epoch_gen_acc = 0
        epoch_disc_acc = 0
        num_batches = 0


        # Use tqdm for progress bar
        for image_batch in tqdm(dataset, desc=f"Epoch {epoch+1}/{epochs}"):
            g_loss, d_loss, g_acc, d_acc = train_step(image_batch)
            epoch_gen_loss += g_loss
            epoch_disc_loss += d_loss
            epoch_gen_acc += g_acc
            epoch_disc_acc += d_acc
            num_batches += 1

        # Average metrics for the epoch
        epoch_gen_loss /= num_batches
```

Question assigned to the following page: 3.1

```
        epoch_disc_loss /= num_batches
        epoch_gen_acc /= num_batches
        epoch_disc_acc /= num_batches

        # Store metrics
        gen_losses.append(epoch_gen_loss)
        disc_losses.append(epoch_disc_loss)
        gen_accuracies.append(epoch_gen_acc)
        disc_accuracies.append(epoch_disc_acc)


        # Generate and save images at specified epochs
        if (epoch + 1) == 10 or (epoch + 1) == 30 or (epoch + 1) == 50:
            generated_images = generate_and_save_images(generator, epoch + 1,␣
↪seed)

            # Also plot the losses
            plt.figure(figsize=(10, 4))
            plt.subplot(1, 2, 1)
            plt.imshow(plt.imread(f'generated_images/epoch_{epoch + 1}.png'))
            plt.title(f'Generated Images at Epoch {epoch + 1}')
            plt.axis('off')

            plt.subplot(1, 2, 2)
            plt.plot(gen_losses, label='Generator Loss')
            plt.plot(disc_losses, label='Discriminator Loss')
            plt.xlabel('Epoch')
            plt.ylabel('Loss')
            plt.legend()
            plt.tight_layout()
            plt.savefig(f'generated_images/losses_at_epoch_{epoch + 1}.png')
            plt.close()

        # # Print progress
        # print(f'Epoch {epoch + 1}/{epochs}, Gen Loss: {epoch_gen_loss:.4f},␣
↪Disc Loss: {epoch_disc_loss:.4f}, Time: {time.time() - start:.2f}s')

        # Print progress with accuracies
        print(f'Epoch {epoch + 1}/{epochs}, Gen Loss: {epoch_gen_loss:.4f},␣
↪Disc Loss: {epoch_disc_loss:.4f}, Gen Acc: {epoch_gen_acc:.4f}, Disc Acc:␣
↪{epoch_disc_acc:.4f}, Time: {time.time() - start:.2f}s')

    # Plot final losses
    plt.figure(figsize=(10, 6))
    plt.plot(gen_losses, label='Generator Loss')
    plt.plot(disc_losses, label='Discriminator Loss')
    plt.xlabel('Epoch')
```

Question assigned to the following page: 3.1

```
    plt.ylabel('Loss')
    plt.legend()
    plt.title('Generator and Discriminator Losses')
    plt.savefig('generated_images/final_losses.png')
    plt.close()

    # Plot final accuracies
    plt.figure(figsize=(10, 6))
    plt.plot(gen_accuracies, label='Generator Accuracy')
    plt.plot(disc_accuracies, label='Discriminator Accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.title('Generator and Discriminator Accuracies')
    plt.savefig('generated_images/final_accuracies.png')
    plt.close()

    return gen_losses, disc_losses, gen_accuracies, disc_accuracies



# Define accuracy calculation function
def calculate_accuracy(logits, labels):
    # Make sure shapes match
    if tf.shape(logits)[0] != tf.shape(labels)[0]:
        # If shapes don't match, use the smaller batch size
        min_batch = tf.minimum(tf.shape(logits)[0], tf.shape(labels)[0])
        logits = logits[:min_batch]
        labels = labels[:min_batch]

    predictions = tf.cast(logits > 0, tf.float32)
    correct = tf.cast(tf.equal(predictions, labels), tf.float32)
    return tf.reduce_mean(correct)
```

[43]:
```
# Train the model and get accuracy data
EPOCHS = 50
train_results = train(train_dataset, EPOCHS)
gen_losses, disc_losses, gen_accuracies, disc_accuracies = train_results

print("Training complete with accuracy tracking! Final images and metrics saved␣
 ↪in 'generated_images' directory.")
```

```
Epoch 1/50:   0%|          | 0/938 [00:00<?, ?it/s]
```

```
Epoch 1/50, Gen Loss: 0.8332, Disc Loss: 1.2272, Gen Acc: 0.3675, Disc Acc:
0.6679, Time: 15.80s
```

```
Epoch 2/50:   0%|          | 0/938 [00:00<?, ?it/s]
```

Question assigned to the following page: 3.1

```
Epoch 2/50, Gen Loss: 0.8270, Disc Loss: 1.3014, Gen Acc: 0.3616, Disc Acc:
0.6317, Time: 10.04s

Epoch 3/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 3/50, Gen Loss: 0.9330, Disc Loss: 1.2083, Gen Acc: 0.2887, Disc Acc:
0.6841, Time: 10.06s

Epoch 4/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 4/50, Gen Loss: 1.0728, Disc Loss: 1.0953, Gen Acc: 0.2275, Disc Acc:
0.7363, Time: 10.04s

Epoch 5/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 5/50, Gen Loss: 1.3034, Disc Loss: 0.9477, Gen Acc: 0.1717, Disc Acc:
0.7884, Time: 10.04s

Epoch 6/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 6/50, Gen Loss: 1.5175, Disc Loss: 0.8086, Gen Acc: 0.1299, Disc Acc:
0.8282, Time: 10.04s

Epoch 7/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 7/50, Gen Loss: 1.5898, Disc Loss: 0.8264, Gen Acc: 0.1363, Disc Acc:
0.8209, Time: 10.04s

Epoch 8/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 8/50, Gen Loss: 1.5598, Disc Loss: 0.8248, Gen Acc: 0.1393, Disc Acc:
0.8189, Time: 10.04s

Epoch 9/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 9/50, Gen Loss: 1.4861, Disc Loss: 0.9034, Gen Acc: 0.1678, Disc Acc:
0.7939, Time: 10.04s

Epoch 10/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 10/50, Gen Loss: 1.4702, Disc Loss: 0.9019, Gen Acc: 0.1689, Disc Acc:
0.7937, Time: 10.67s

Epoch 11/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 11/50, Gen Loss: 1.4557, Disc Loss: 0.9225, Gen Acc: 0.1734, Disc Acc:
0.7897, Time: 10.05s

Epoch 12/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 12/50, Gen Loss: 1.3214, Disc Loss: 0.9999, Gen Acc: 0.2022, Disc Acc:
0.7618, Time: 10.06s

Epoch 13/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 13/50, Gen Loss: 1.2923, Disc Loss: 1.0100, Gen Acc: 0.2057, Disc Acc:
0.7558, Time: 10.04s

Epoch 14/50:   0%|          | 0/938 [00:00<?, ?it/s]
```

Question assigned to the following page: 3.1

```
Epoch 14/50, Gen Loss: 1.2458, Disc Loss: 1.0342, Gen Acc: 0.2149, Disc Acc:
0.7493, Time: 10.03s

Epoch 15/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 15/50, Gen Loss: 1.1936, Disc Loss: 1.0755, Gen Acc: 0.2348, Disc Acc:
0.7328, Time: 10.05s

Epoch 16/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 16/50, Gen Loss: 1.2064, Disc Loss: 1.0629, Gen Acc: 0.2291, Disc Acc:
0.7380, Time: 10.04s

Epoch 17/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 17/50, Gen Loss: 1.1940, Disc Loss: 1.0605, Gen Acc: 0.2267, Disc Acc:
0.7386, Time: 10.03s

Epoch 18/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 18/50, Gen Loss: 1.1436, Disc Loss: 1.0873, Gen Acc: 0.2397, Disc Acc:
0.7263, Time: 10.04s

Epoch 19/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 19/50, Gen Loss: 1.1424, Disc Loss: 1.0929, Gen Acc: 0.2388, Disc Acc:
0.7265, Time: 10.03s

Epoch 20/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 20/50, Gen Loss: 1.1082, Disc Loss: 1.1115, Gen Acc: 0.2487, Disc Acc:
0.7190, Time: 10.03s

Epoch 21/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 21/50, Gen Loss: 1.0753, Disc Loss: 1.1377, Gen Acc: 0.2632, Disc Acc:
0.7035, Time: 10.04s

Epoch 22/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 22/50, Gen Loss: 1.0970, Disc Loss: 1.1241, Gen Acc: 0.2559, Disc Acc:
0.7132, Time: 10.04s

Epoch 23/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 23/50, Gen Loss: 1.0962, Disc Loss: 1.1366, Gen Acc: 0.2622, Disc Acc:
0.7074, Time: 10.03s

Epoch 24/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 24/50, Gen Loss: 1.0402, Disc Loss: 1.1626, Gen Acc: 0.2752, Disc Acc:
0.6938, Time: 10.03s

Epoch 25/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 25/50, Gen Loss: 1.0138, Disc Loss: 1.1861, Gen Acc: 0.2900, Disc Acc:
0.6842, Time: 10.06s

Epoch 26/50:   0%|          | 0/938 [00:00<?, ?it/s]
```

Question assigned to the following page: 3.1

```
Epoch 26/50, Gen Loss: 0.9985, Disc Loss: 1.1960, Gen Acc: 0.2958, Disc Acc:
0.6785, Time: 10.03s

Epoch 27/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 27/50, Gen Loss: 0.9926, Disc Loss: 1.1979, Gen Acc: 0.2954, Disc Acc:
0.6775, Time: 10.03s

Epoch 28/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 28/50, Gen Loss: 0.9882, Disc Loss: 1.1966, Gen Acc: 0.2931, Disc Acc:
0.6801, Time: 10.06s

Epoch 29/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 29/50, Gen Loss: 0.9699, Disc Loss: 1.2160, Gen Acc: 0.3055, Disc Acc:
0.6697, Time: 10.21s

Epoch 30/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 30/50, Gen Loss: 0.9551, Disc Loss: 1.2254, Gen Acc: 0.3123, Disc Acc:
0.6630, Time: 10.60s

Epoch 31/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 31/50, Gen Loss: 0.9621, Disc Loss: 1.2188, Gen Acc: 0.3102, Disc Acc:
0.6671, Time: 10.06s

Epoch 32/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 32/50, Gen Loss: 0.9503, Disc Loss: 1.2278, Gen Acc: 0.3146, Disc Acc:
0.6612, Time: 10.04s

Epoch 33/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 33/50, Gen Loss: 0.9546, Disc Loss: 1.2239, Gen Acc: 0.3128, Disc Acc:
0.6651, Time: 10.02s

Epoch 34/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 34/50, Gen Loss: 0.9440, Disc Loss: 1.2325, Gen Acc: 0.3190, Disc Acc:
0.6595, Time: 10.06s

Epoch 35/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 35/50, Gen Loss: 0.9309, Disc Loss: 1.2451, Gen Acc: 0.3263, Disc Acc:
0.6533, Time: 10.04s

Epoch 36/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 36/50, Gen Loss: 0.9329, Disc Loss: 1.2391, Gen Acc: 0.3218, Disc Acc:
0.6571, Time: 10.04s

Epoch 37/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 37/50, Gen Loss: 0.9291, Disc Loss: 1.2407, Gen Acc: 0.3244, Disc Acc:
0.6553, Time: 10.04s

Epoch 38/50:   0%|          | 0/938 [00:00<?, ?it/s]
```

Question assigned to the following page: 3.1

```
Epoch 38/50, Gen Loss: 0.9198, Disc Loss: 1.2500, Gen Acc: 0.3303, Disc Acc:
0.6498, Time: 10.04s

Epoch 39/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 39/50, Gen Loss: 0.9262, Disc Loss: 1.2433, Gen Acc: 0.3235, Disc Acc:
0.6543, Time: 10.04s

Epoch 40/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 40/50, Gen Loss: 0.9210, Disc Loss: 1.2460, Gen Acc: 0.3253, Disc Acc:
0.6519, Time: 10.06s

Epoch 41/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 41/50, Gen Loss: 0.9080, Disc Loss: 1.2534, Gen Acc: 0.3324, Disc Acc:
0.6481, Time: 10.03s

Epoch 42/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 42/50, Gen Loss: 0.9029, Disc Loss: 1.2597, Gen Acc: 0.3394, Disc Acc:
0.6428, Time: 10.04s

Epoch 43/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 43/50, Gen Loss: 0.9039, Disc Loss: 1.2528, Gen Acc: 0.3318, Disc Acc:
0.6485, Time: 10.04s

Epoch 44/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 44/50, Gen Loss: 0.9040, Disc Loss: 1.2573, Gen Acc: 0.3332, Disc Acc:
0.6458, Time: 10.03s

Epoch 45/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 45/50, Gen Loss: 0.9053, Disc Loss: 1.2538, Gen Acc: 0.3328, Disc Acc:
0.6470, Time: 10.02s

Epoch 46/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 46/50, Gen Loss: 0.9028, Disc Loss: 1.2553, Gen Acc: 0.3357, Disc Acc:
0.6456, Time: 10.04s

Epoch 47/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 47/50, Gen Loss: 0.9029, Disc Loss: 1.2570, Gen Acc: 0.3346, Disc Acc:
0.6465, Time: 10.04s

Epoch 48/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 48/50, Gen Loss: 0.9104, Disc Loss: 1.2539, Gen Acc: 0.3335, Disc Acc:
0.6479, Time: 10.04s

Epoch 49/50:   0%|          | 0/938 [00:00<?, ?it/s]

Epoch 49/50, Gen Loss: 1.1364, Disc Loss: 1.1250, Gen Acc: 0.2714, Disc Acc:
0.7069, Time: 10.03s

Epoch 50/50:   0%|          | 0/938 [00:00<?, ?it/s]
```

Question assigned to the following page: 3.1

```
Epoch 50/50, Gen Loss: 0.9879, Disc Loss: 1.2164, Gen Acc: 0.3133, Disc Acc:
0.6654, Time: 10.57s
Training complete with accuracy tracking! Final images and metrics saved in
'generated_images' directory.
```

[44]:
```
"""
Visualization of Intermediate DCGAN Results during Training

Now, by comparing images from different epochs, we can observe how the␣
 ↪generator improves over time:

- Early epochs (10): Basic shapes and patterns begin to emerge but images are␣
 ↪blurry

- Mid-training (30): More defined clothing structures with improved details but␣
 ↪fails at edges of the figures.

- Final result (50): Clearer, more realistic fashion items with better defined␣
 ↪features.

"""

# Define key epochs to display
key_epochs = [10, 30, 50]

# Create a large figure with 1 column and 3 rows
fig, axes = plt.subplots(nrows=3, ncols=1, figsize=(10, 24))

# Set a title for the entire figure
fig.suptitle('DCGAN Generated Fashion Images Progression', fontsize=20)

# Loop through the epochs and display each image
for i, epoch in enumerate(key_epochs):
    img_path = f'generated_images/epoch_{epoch}.png'
    if os.path.exists(img_path):
        img = plt.imread(img_path)
        axes[i].imshow(img)
        axes[i].set_title(f'Epoch {epoch}', fontsize=16)
        axes[i].axis('off')
    else:
        axes[i].text(0.5, 0.5, f"Image for epoch {epoch} not found",
                     horizontalalignment='center', fontsize=14)
        axes[i].axis('off')

# Adjust layout to prevent overlap
plt.tight_layout()
plt.subplots_adjust(top=0.95)  # Make room for suptitle
```

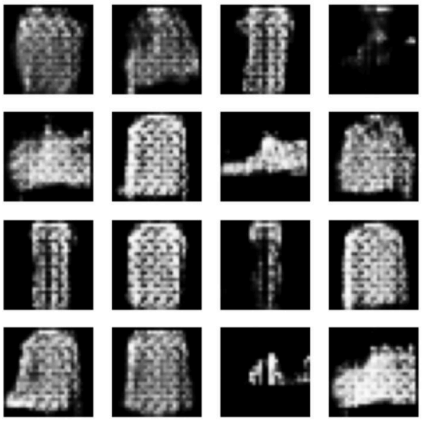
Question assigned to the following page: 3.1

```
plt.show()
```

Questions assigned to the following page: 3.2 and 3.1

DCGAN Generated Fashion Images Progression

Epoch 10



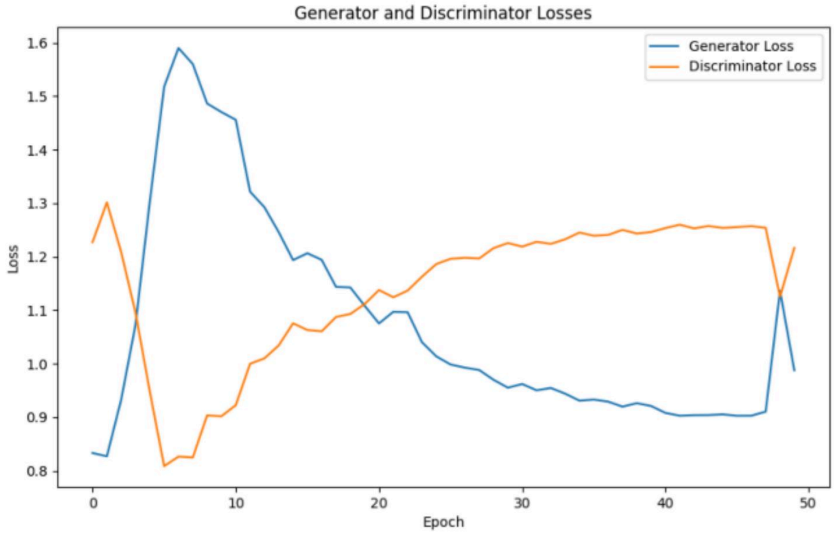
Epoch 30

Epoch 50

Question assigned to the following page: 3.1

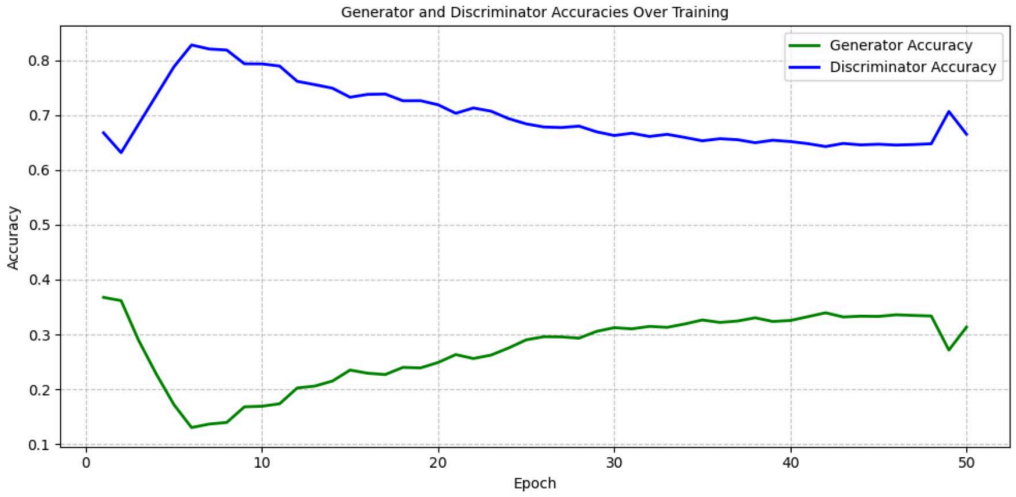
[56]:
```
# Visualization of Training Loss Curves

# Path to the loss curve image
loss_path = 'generated_images/final_losses.png'

if os.path.exists(loss_path):
    # If the final loss image exists, display it
    plt.figure(figsize=(12, 6))
    loss_img = plt.imread(loss_path)
    plt.imshow(loss_img)
    plt.axis('off')
    plt.show()
else:
    # If we need to recreate the loss plot from individual epoch data
    try:
        # Try to plot using the loss data if available
        plt.figure(figsize=(12, 6))
        epochs = range(1, len(gen_losses) + 1)
        plt.plot(epochs, gen_losses, 'b-', linewidth=2, label='Generator Loss')
        plt.plot(epochs, disc_losses, 'r-', linewidth=2, label='Discriminator␣
 ↪Loss')
        plt.title('Generator and Discriminator Losses Over Training',␣
 ↪fontsize=18)
        plt.xlabel('Epoch', fontsize=14)
        plt.ylabel('Loss', fontsize=14)
        plt.legend(fontsize=14)
        plt.grid(True, linestyle='--', alpha=0.7)
        plt.tight_layout()
        plt.show()
    except NameError:
        # If loss data isn't available, look for individual epoch loss images
        loss_files = sorted([f for f in os.listdir('generated_images')
                             if f.startswith('losses_at_epoch_')])

        if loss_files:
            plt.figure(figsize=(12, 6))
            latest_loss_file = loss_files[-1]  # Get the most recent loss file
            loss_img = plt.imread(f'generated_images/{latest_loss_file}')
            plt.imshow(loss_img)
            plt.axis('off')
            plt.show()
        else:
            print("No loss data or images found. Please ensure you've saved␣
 ↪loss data during training.")
```

Questions assigned to the following page: 3.2 and 3.1



[61]:
```
# Add visualization for accuracy curves
plt.figure(figsize=(10, 5))
epochs = range(1, len(gen_accuracies) + 1)
plt.plot(epochs, gen_accuracies, 'g-', linewidth=2, label='Generator Accuracy')
plt.plot(epochs, disc_accuracies, 'b-', linewidth=2, label='Discriminator␣
 ↪Accuracy')
plt.title('Generator and Discriminator Accuracies Over Training', fontsize=10)
plt.xlabel('Epoch', fontsize=10)
plt.ylabel('Accuracy', fontsize=10)
plt.legend(fontsize=10)
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

Questions assigned to the following page: 3.2 and 3.1



3 Comments

3.1 Quality Progression Analysis

The generated images show clear improvement in quality and detail across training epochs:

- **Epoch 10**: The initial generated images display basic structural forms of clothing items but are relatively blurry and lack fine details. We can recognize spotty colour fill and rough outlines of items, which screams about them containing significant noise and distortion.
- **Epoch 30**: We can see improvement. The generated images show clearer boundaries, more defined shapes, and better contrast. Though there are still rough outlines and colour fill consistency issues, they are noe more recognizable with improved texture representation.
- **Epoch 50**: The final epoch shows the most refined results. The generated fashion items display sharper edges, more consistent structures, and greater variety. Specific clothing categories like dresses, pants, shirts, and shoes are clearly distinguishable with realistic proportions and details.

3.2 Loss Dynamics

The loss curves provide valuable insights into the adversarial training process:

- **Generator Loss**: Initially volatile, the generator loss shows a general downward trend throughout training. This indicates the generator's improving ability to create images that successfully fool the discriminator.
- **Discriminator Loss**: After initial spiking low, the discriminator loss gradually increases and stabilizes at higher values, suggesting the discriminator finds it increasingly difficult to distinguish between real and generated images as training progresses.

Question assigned to the following page: 3.2

- **Adversarial Balance**: The **convergent behavior of both losses after epoch 20 and epoch 30** indicates a healthy equilibrium was achieved between the generator and discriminator, avoiding common GAN training issues like mode collapse or oscillation.

[47]:
```
print ("GG")
```

```
GG
```