```
%cd /content/drive/MyDrive/ImageBindFinetuning_new
```

```
/content/drive/MyDrive/ImageBindFinetuning_new
```

```
!pip install -r requirements.txt --quiet
```

```
  Preparing metadata (setup.py) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 50.2/50.2 kB 4.4 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 42.2/42.2 kB 3.3 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.1/7.1 MB 15.1 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 151.6/151.6 kB 13.4 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 890.1/890.1 MB 1.7 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.3/24.3 MB 42.7 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.2/4.2 MB 62.0 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 510.0/510.0 kB 19.9 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.6/13.6 MB 78.5 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.8/1.8 MB 62.4 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 317.1/317.1 MB 4.8 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.0/21.0 MB 84.3 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 849.3/849.3 kB 53.2 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 557.1/557.1 MB 3.4 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 54.4/54.4 kB 3.9 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 43.2/43.2 kB 3.4 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.6/11.6 MB 54.8 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 66.4/66.4 kB 5.9 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 80.8/80.8 kB 7.4 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 55.5/55.5 kB 4.5 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 64.3/64.3 kB 5.6 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.2/1.2 MB 59.1 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.1/3.1 MB 87.7 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 59.1 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 868.8/868.8 kB 53.7 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.1/5.1 MB 87.4 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.5/1.5 MB 12.0 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.8/62.8 kB 5.9 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 129.9/129.9 kB 11.4 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 230.0/230.0 kB 19.8 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 33.5/33.5 MB 17.6 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 268.9/268.9 kB 20.9 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 802.3/802.3 kB 47.6 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.4/58.4 kB 5.0 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.3/58.3 kB 5.6 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 139.2/139.2 kB 13.3 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.4/12.4 MB 24.3 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 82.7/82.7 kB 7.2 MB/s eta 0:00:00
  Building wheel for pytorchvideo (setup.py) ... done
  Building wheel for fvcore (setup.py) ... done
  Building wheel for iopath (setup.py) ... done
  Building wheel for mayavi (pyproject.toml) ... done
ERROR: pip's dependency resolver does not currently take into account all the packages that are installe
torchtext 0.18.0 requires torch>=2.3.0, but you have torch 1.13.0 which is incompatible.
```

```
import os
current_directory = os.getcwd()
```

```
import torch
import logging
import data
from models import imagebind_model
from models.imagebind_model import ModalityType

logging.basicConfig(level=logging.INFO, force=True)

# device = "cuda:0" if torch.cuda.is_available() else "cpu"
device = "cpu"
```

```
/usr/local/lib/python3.10/dist-packages/torchvision/transforms/_functional_video.py:6: UserWarning: The
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torchvision/transforms/_transforms_video.py:22: UserWarning: The
  warnings.warn(
```

```
model = imagebind_model.imagebind_huge(pretrained=True)
```

```
model.eval()
model.to(device)
```

```
ImageBindModel(
  (modality_preprocessors): ModuleDict(
    (vision): RGBDTPreprocessor(
      (cls_token): tensor((1, 1, 1280), requires_grad=True)

      (rgbt_stem): PatchEmbedGeneric(
        (proj): Sequential(
          (0): PadIm2Video()
          (1): Conv3d(3, 1280, kernel_size=(2, 14, 14), stride=(2, 14, 14), bias=False)
        )
      )
      (pos_embedding_helper): SpatioTemporalPosEmbeddingHelper(
        (pos_embed): tensor((1, 257, 1280), requires_grad=True)

      )
    )
    (text): TextPreprocessor(
      (pos_embed): tensor((1, 77, 1024), requires_grad=True)
      (mask): tensor((77, 77), requires_grad=False)

      (token_embedding): Embedding(49408, 1024)
    )
    (audio): AudioPreprocessor(
      (cls_token): tensor((1, 1, 768), requires_grad=True)

      (rgbt_stem): PatchEmbedGeneric(
        (proj): Conv2d(1, 768, kernel_size=(16, 16), stride=(10, 10), bias=False)
        (norm_layer): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
      )
      (pos_embedding_helper): SpatioTemporalPosEmbeddingHelper(
        (pos_embed): tensor((1, 229, 768), requires_grad=True)

      )
    )
    (depth): RGBDTPreprocessor(
      (cls_token): tensor((1, 1, 384), requires_grad=True)

      (depth_stem): PatchEmbedGeneric(
        (proj): Conv2d(1, 384, kernel_size=(16, 16), stride=(16, 16), bias=False)
        (norm_layer): LayerNorm((384,), eps=1e-05, elementwise_affine=True)
      )
      (pos_embedding_helper): SpatioTemporalPosEmbeddingHelper(
        (pos_embed): tensor((1, 197, 384), requires_grad=True)

      )
    )
    (thermal): ThermalPreprocessor(
      (cls_token): tensor((1, 1, 768), requires_grad=True)

      (rgbt_stem): PatchEmbedGeneric(
        (proj): Conv2d(1, 768, kernel_size=(16, 16), stride=(16, 16), bias=False)
        (norm_layer): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
      )
      (pos_embedding_helper): SpatioTemporalPosEmbeddingHelper(
        (pos_embed): tensor((1, 197, 768), requires_grad=True)

      )
    )
```

```
train_path = current_directory+"/new_data/"
train_path_img = train_path + "images/"
train_path_audio = train_path + "audio/"
```

```
labels = {}
for lb_idx,label in enumerate(os.listdir(train_path_img)):
  labels[label] = lb_idx
```

```
image_paths = []
audio_paths = []
label_list = []
for label in os.listdir(train_path_img):
  train_path_img_cat = train_path_img + label
  train_path_audio_cat = train_path_audio + label
  for img_file_name in os.listdir(train_path_img_cat):
    img_file_path = train_path_img_cat + "/" + img_file_name
    audio_file_name = img_file_name.split(".")[0] + ".wav"
    audio_file_path = train_path_audio_cat + "/" + audio_file_name
    image_paths.append(img_file_path)
    audio_paths.append(audio_file_path)
    label_list.append(labels[label])
```

Available: 42.36 compute units

Usage rate: approximately 4.82 per hour

You have 1 active session.

Manage sessions

Want more memory and disk space?

Upgrade to Colab Pro

Python 3 Google Compute Engine backend (GPU)

Showing resources from 10:13 PM to 11:21 PM

System RAM
8.3 / 12.7 GB

GPU RAM
0.0 / 16.0 GB

Disk
39.1 / 78.2 GB

```
image_paths.sort()
audio_paths.sort()
label_list.sort()
```

```
# image_paths = image_paths[:10]
# audio_paths = audio_paths[:10]
# label_list = label_list[:10]
```

```
# inputs = {
#     ModalityType.VISION: data.load_and_transform_vision_data(image_paths, device, to_tensor=True),
#     ModalityType.AUDIO: data.load_and_transform_audio_data(audio_paths, device),
# }
```

```
# torch.save(inputs, 'inputs.pth')
```

```
loaded_inputs = torch.load('inputs.pth', map_location=torch.device('cpu'))
```

```
# import h5py

# with h5py.File('vision_embeddings.h5', 'w') as h5f:
#   num_embeddings = loaded_inputs['vision'].shape[0]
#   embedding_size = 1024
#   dataset = h5f.create_dataset('vision_embeddings', (num_embeddings, embedding_size), dtype='f')
#   for i in range(num_embeddings):
#         # Calculate the embedding for the current input
#         embd = model({'vision': torch.unsqueeze(loaded_inputs['vision'][i], dim=0)})

#         # Write the embedding to the dataset
#         dataset[i] = embd['vision'].detach().cpu().numpy()

#         # Free the memory used by the current embedding
#         del embd
#         # torch.cuda.empty_cache()  # Clear cached memory if using GPU
```

```
import numpy as np
file_path = 'vision_embeddings.h5'

# Open the HDF5 file
with h5py.File(file_path, 'r') as h5f:
    # Access the dataset
    dataset = h5f['vision_embeddings']

    # Load the data into a NumPy array
    vision_embeddings = np.array(dataset)

print("Embeddings loaded from 'embeddings.h5'")
print(vision_embeddings.shape)  # Print the shape of the loaded embeddings
```

```
Embeddings loaded from 'embeddings.h5'
(1617, 1024)
```

```
import h5py
with h5py.File('audio_embeddings.h5', 'w') as h5f:
  num_embeddings = loaded_inputs['audio'].shape[0]
  embedding_size = 1024
  dataset = h5f.create_dataset('audio_embeddings', (num_embeddings, embedding_size), dtype='f')
  for i in range(num_embeddings):
        # Calculate the embedding for the current input
        embd = model({'audio': torch.unsqueeze(loaded_inputs['audio'][i], dim=0)})

        # Write the embedding to the dataset
        dataset[i] = embd['audio'].detach().cpu().numpy()

        # Free the memory used by the current embedding
        del embd
        # torch.cuda.empty_cache()  # Clear cached memory if using GPU
```

```
import numpy as np
file_path = 'audio_embeddings.h5'

# Open the HDF5 file
with h5py.File(file_path, 'r') as h5f:
    # Access the dataset
    dataset = h5f['audio_embeddings']

    # Load the data into a NumPy array
    audio_embeddings = np.array(dataset)

print("Embeddings loaded from 'embeddings.h5'")
print(audio_embeddings.shape)  # Print the shape of the loaded embeddings
```

```
Embeddings loaded from 'embeddings.h5'
(1617, 1024)
```

```
import numpy as np
```

```
vision_embeddings_np = embeddings[ModalityType.VISION].cpu().numpy()
audio_embeddings_np = embeddings[ModalityType.AUDIO].cpu().numpy()
```

```
np.save('vision_embeddings.npy', vision_embeddings_np)
np.save('audio_embeddings.npy', audio_embeddings_np)
```

```
vision_embeddings_np_loaded = np.load('vision_embeddings.npy')
audio_embeddings_np_loaded = np.load('audio_embeddings.npy')
```

```
vision_embeddings = torch.tensor(vision_embeddings_np_loaded)
audio_embeddings = torch.tensor(audio_embeddings_np_loaded)
```

```
vision_embeddings.shape
```

```
(1617, 1024)
```

```
audio_embeddings.shape
```

```
(1617, 1024)
```

```
X = []
y = []
for i in range(vision_embeddings.shape[0]):
  #concatenated_embedding = np.concatenate((vision_embeddings[i], audio_embeddings[i]))
  concatenated_embedding = vision_embeddings[i] + audio_embeddings[i]
  # np.save(f'concatenated_embedding_{i}.npy', concatenated_embedding)
  X.append(concatenated_embedding)
  y.append(label_list[i])
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=43)
```

```
from sklearn.svm import SVC
svm_classifier = SVC(kernel='linear')  #'rbf
svm_classifier.fit(X_train, y_train)
```

```
▾ SVC
SVC(kernel='linear')
```

```
from sklearn.metrics import precision_recall_fscore_support, accuracy_score

# Assuming y_pred and y_test are already defined
y_pred = svm_classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)

# Calculate precision, recall, and F1 score
precision, recall, f1, _ = precision_recall_fscore_support(y_test, y_pred, average='weighted')

print(f"Accuracy: {accuracy * 100:.2f}%")
print(f"Precision: {precision * 100:.2f}%")
print(f"Recall: {recall * 100:.2f}%")
print(f"F1 Score: {f1 * 100:.2f}%")
```

```
Accuracy: 74.19%
Precision: 72.91%
```

```
Recall: 74.19%
F1 Score: 73.07%
```