```
#GPU


import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.preprocessing.image import ImageDat
from sklearn.model_selection import train_test_split




# Set up parameters
AUTO = tf.data.AUTOTUNE
IMAGE_SIZE = [224, 224]  # adjust size as needed
batch_size = 32
gcs_pattern_train = 'gs://grad_proj/train/*/*.jpg'
gcs_pattern_test = 'gs://grad_proj/test/*/*.jpg'
num_images_per_class = 100
num_images_per_class = 100


# Define data loading functions
def parse_image(filename, label):
    img = tf.io.read_file(filename)
    img = tf.image.decode_jpeg(img, channels=3)
    img = tf.image.resize(img, IMAGE_SIZE)
    img = tf.cast(img, tf.float32) / 255.0
    return img, label

# Define data loading functions
def load_dataset(gcs_pattern, num_images):
    filenames = tf.io.gfile.glob(gcs_pattern)
    random.shuffle(filenames)  # Shuffle the filenames fo
    filenames = filenames[:num_images]  # Select a subset
    labels = [1 if 'dog' in filename else 0 for filename
    dataset = tf.data.Dataset.from_tensor_slices((filenam
    dataset = dataset.map(parse_image, num_parallel_calls
    return dataset



import random
# Load datasets
train_dataset = load_dataset(gcs_pattern_train, num_imag
test_dataset = load_dataset(gcs_pattern_test, num_images

# Assuming your train dataset is named 'train_dataset'
train_dataset_length = tf.data.experimental.cardinality(

print(f"Train dataset length: {train_dataset_length}")
```

```
Train dataset length: 100
```

Resources ✕

You are not subscribed. Learn more.

You currently have zero compute units available. Resources offered free of charge are not guaranteed. Purchase more units here.

Manage sessions

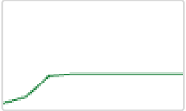
Want more memory and disk space? ✕

Upgrade to Colab Pro

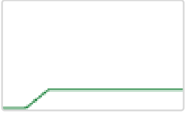
Python 3 Google Compute Engine backend (GPU)

Showing resources from 03:17 to 03:34

System RAM
4.2 / 12.7 GB



GPU RAM
2.8 / 15.0 GB



Disk
26.9 / 78.2 GB

```
# Define and compile the model
model = models.Sequential()
model.add(layers.Conv2D(32, kernel_size=(3, 3), padding=
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D(pool_size=(2, 2), strides=
model.add(layers.Conv2D(64, kernel_size=(3, 3), padding=
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D(pool_size=(2, 2), strides=
model.add(layers.Conv2D(128, kernel_size=(3, 3), padding
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D(pool_size=(2, 2), strides=
model.add(layers.Flatten())
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dropout(0.1))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dropout(0.1))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary_crossentrop
```

```
import time

start_time = time.time()

history = model.fit(
    train_dataset.shuffle(1000).batch(batch_size).prefet
    epochs=10,
    validation_data=test_dataset.batch(batch_size).prefe
)

end_time = time.time()
elapsed_time = end_time - start_time

print(f"Training took {elapsed_time} seconds.")
```

```
Epoch 1/10
4/4 [==============================] - 29s 3s/step -
Epoch 2/10
4/4 [==============================] - 12s 2s/step -
Epoch 3/10
4/4 [==============================] - 16s 3s/step -
Epoch 4/10
4/4 [==============================] - 11s 2s/step -
Epoch 5/10
4/4 [==============================] - 16s 3s/step -
Epoch 6/10
4/4 [==============================] - 16s 3s/step -
Epoch 7/10
4/4 [==============================] - 10s 2s/step -
Epoch 8/10
4/4 [==============================] - 15s 3s/step -
Epoch 9/10
4/4 [==============================] - 15s 3s/step -
Epoch 10/10
4/4 [==============================] - 10s 2s/step -
Training took 161.66323280334473 seconds.
```

Change runtime type