Tanawin-st123975-ViT-v100

November 17, 2023

[1]:
```
!nvidia-smi
```

```
Fri Nov 17 07:29:33 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 525.105.17   Driver Version: 525.105.17   CUDA Version: 12.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2…  Off  | 00000000:00:04.0 Off |                    0 |
| N/A   34C    P0    24W / 300W |      0MiB / 16384MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

[2]:
```
# from google.colab import files
# files.upload()
```

[3]:
```
# import os

# # Set Kaggle API credentials as environment variables
# os.environ['KAGGLE_USERNAME'] = 'kmutnb'
# os.environ['KAGGLE_KEY'] = '2e8f3eec48ac90959791a330f9109431'

# import kaggle

# # Authenticate with Kaggle API
# kaggle.api.authenticate()
```

[4]:
```
# !kaggle datasets download -d gpiosenka/sports-classification
```

[5]:
```
# !unzip /content/sports-classification.zip -d data
```

[6]:
```
!pip install vit-pytorch linformer
```

```
Collecting vit-pytorch
  Downloading vit_pytorch-1.6.4-py3-none-any.whl (100 kB)
                           100.3/100.3

kB 2.5 MB/s eta 0:00:00
Collecting linformer
  Downloading linformer-0.2.1-py3-none-any.whl (6.1 kB)
Collecting einops>=0.7.0 (from vit-pytorch)
  Downloading einops-0.7.0-py3-none-any.whl (44 kB)
                           44.6/44.6 kB
6.5 MB/s eta 0:00:00
Requirement already satisfied: torch>=1.10 in
/usr/local/lib/python3.10/dist-packages (from vit-pytorch) (2.1.0+cu118)
Requirement already satisfied: torchvision in /usr/local/lib/python3.10/dist-
packages (from vit-pytorch) (0.16.0+cu118)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-
packages (from torch>=1.10->vit-pytorch) (3.13.1)
Requirement already satisfied: typing-extensions in
/usr/local/lib/python3.10/dist-packages (from torch>=1.10->vit-pytorch) (4.5.0)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages
(from torch>=1.10->vit-pytorch) (1.12)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-
packages (from torch>=1.10->vit-pytorch) (3.2.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages
(from torch>=1.10->vit-pytorch) (3.1.2)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages
(from torch>=1.10->vit-pytorch) (2023.6.0)
Requirement already satisfied: triton==2.1.0 in /usr/local/lib/python3.10/dist-
packages (from torch>=1.10->vit-pytorch) (2.1.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages
(from torchvision->vit-pytorch) (1.23.5)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (from torchvision->vit-pytorch) (2.31.0)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in
/usr/local/lib/python3.10/dist-packages (from torchvision->vit-pytorch) (9.4.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from jinja2->torch>=1.10->vit-pytorch)
(2.1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->torchvision->vit-
pytorch) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests->torchvision->vit-pytorch) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in
```

```
/usr/local/lib/python3.10/dist-packages (from requests->torchvision->vit-
pytorch) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests->torchvision->vit-
pytorch) (2023.7.22)
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-
packages (from sympy->torch>=1.10->vit-pytorch) (1.3.0)
Installing collected packages: einops, linformer, vit-pytorch
Successfully installed einops-0.7.0 linformer-0.2.1 vit-pytorch-1.6.4
```

[7]:
```
import glob
from itertools import chain
import os
import random
import zipfile

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from linformer import Linformer
from PIL import Image
from sklearn.model_selection import train_test_split
from torch.optim.lr_scheduler import StepLR
from torch.utils.data import DataLoader, Dataset
from torchvision import datasets, transforms
from tqdm.notebook import tqdm

from vit_pytorch.efficient import ViT

import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader
from torchvision.datasets import ImageFolder
from torchvision.transforms import transforms
```

[8]:
```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
device
```

[8]:
```
device(type='cuda')
```

[9]:
```
transform = transforms.Compose([
    transforms.RandomResizedCrop(224),
```

```
    transforms.RandomHorizontalFlip(),
    transforms.RandomVerticalFlip(),
    transforms.ColorJitter(brightness=0.4, contrast=0.4, saturation=0.4, hue=0.
 ↪1),
    transforms.RandomRotation(30),
    transforms.RandomAffine(degrees=0, translate=(0.1, 0.1), scale=(0.9, 1.1),␣
 ↪shear=10),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])
```

[10]:
```
train_data = "/content/data/train"
val_data = "/content/data/valid"
```

[11]:
```
train_dataset = ImageFolder(train_data, transform=transform)
valid_dataset = ImageFolder(val_data, transform=transform)
```

[12]:
```
batch_size = 8
# Define the data loaders
train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True,␣
 ↪num_workers=4)
valid_loader = DataLoader(valid_dataset, batch_size=batch_size, shuffle=False,␣
 ↪num_workers=4)
```

[13]:
```
from vit_pytorch import ViT

v = ViT(
    image_size = 256,
    patch_size = 32,
    num_classes = 100,
    dim = 1024,
    depth = 6,
    heads = 16,
    mlp_dim = 2048,
    dropout = 0.1,
    emb_dropout = 0.1
)
```

[14]:
```
from torchvision.models import vit_b_16 as ViT, ViT_B_16_Weights
model = ViT(weights=ViT_B_16_Weights.DEFAULT)

# from torchvision.models import vit_l_32 as ViT, ViT_L_32_Weights
# model = ViT(weights=ViT_L_32_Weights.DEFAULT)

model = model.to(device)
```

```
Downloading: "https://download.pytorch.org/models/vit_b_16-c867db91.pth" to
```

```
/root/.cache/torch/hub/checkpoints/vit_b_16-c867db91.pth
100%|      | 330M/330M [00:05<00:00, 67.6MB/s]
```

[15]:
```
model._dropout = nn.Dropout(p=0.5)
```

[16]:
```
# Define the loss function and optimizer
criterion = nn.CrossEntropyLoss()
# optimizer = optim.Adam(model.parameters(), lr=0.001)
# optimizer = optim.Adam(model.parameters(), lr=0.000001)
optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)
```

[17]:
```
# Define the learning rate scheduler
scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=5, gamma=0.1)
```

[18]:
```
model_name = "ViT-"+str(batch_size)+"-15-nov-23"
# Training loop
num_epochs = 10
train_loss_list = []
valid_loss_list = []
train_accuracy_list = []
valid_accuracy_list = []


for epoch in range(num_epochs):
    model.train()
    running_loss = 0.0
    correct = 0
    total = 0

    for images, labels in train_loader:
        images = images.to(device)
        labels = labels.to(device)

        optimizer.zero_grad()

        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
        _, predicted = torch.max(outputs, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

    train_loss = running_loss / len(train_loader)
    train_accuracy = correct / total
```

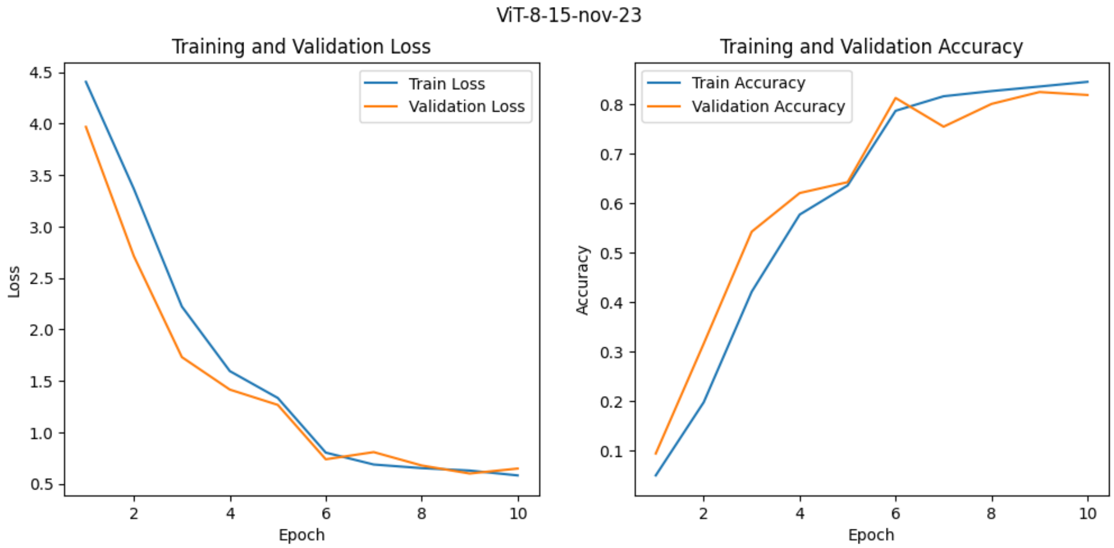
```python
    # Validation loop
    model.eval()
    running_loss = 0.0
    correct = 0
    total = 0

    with torch.no_grad():
        for images, labels in valid_loader:
            images = images.to(device)
            labels = labels.to(device)

            outputs = model(images)
            loss = criterion(outputs, labels)

            running_loss += loss.item()
            _, predicted = torch.max(outputs, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()

    valid_loss = running_loss / len(valid_loader)
    valid_accuracy = correct / total

    # Store loss and accuracy values
    train_loss_list.append(train_loss)
    valid_loss_list.append(valid_loss)
    train_accuracy_list.append(train_accuracy)
    valid_accuracy_list.append(valid_accuracy)

    # Print the training/validation statistics
    print(f"Epoch: {epoch+1}/{num_epochs} | "
          f"Train Loss: {train_loss:.4f} | Train Acc: {train_accuracy:.4f} | "
          f"Valid Loss: {valid_loss:.4f} | Valid Acc: {valid_accuracy:.4f}")


    # Update the learning rate
    scheduler.step()

# Save the trained model
torch.save(model.state_dict(), model_name+".pth")
```

```
Epoch: 1/10 | Train Loss: 4.4063 | Train Acc: 0.0498 | Valid Loss: 3.9686 |
Valid Acc: 0.0940
Epoch: 2/10 | Train Loss: 3.3651 | Train Acc: 0.1976 | Valid Loss: 2.7117 |
Valid Acc: 0.3160
Epoch: 3/10 | Train Loss: 2.2224 | Train Acc: 0.4208 | Valid Loss: 1.7309 |
Valid Acc: 0.5420
```

```
Epoch: 4/10 | Train Loss: 1.5948 | Train Acc: 0.5765 | Valid Loss: 1.4149 |
Valid Acc: 0.6200
Epoch: 5/10 | Train Loss: 1.3335 | Train Acc: 0.6354 | Valid Loss: 1.2661 |
Valid Acc: 0.6420
Epoch: 6/10 | Train Loss: 0.8024 | Train Acc: 0.7861 | Valid Loss: 0.7375 |
Valid Acc: 0.8120
Epoch: 7/10 | Train Loss: 0.6867 | Train Acc: 0.8154 | Valid Loss: 0.8072 |
Valid Acc: 0.7540
Epoch: 8/10 | Train Loss: 0.6515 | Train Acc: 0.8259 | Valid Loss: 0.6773 |
Valid Acc: 0.8000
Epoch: 9/10 | Train Loss: 0.6279 | Train Acc: 0.8351 | Valid Loss: 0.5994 |
Valid Acc: 0.8240
Epoch: 10/10 | Train Loss: 0.5808 | Train Acc: 0.8446 | Valid Loss: 0.6475 |
Valid Acc: 0.8180
```

[19]:

```
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
# Plot training and validation loss
plt.plot(range(1, num_epochs+1), train_loss_list, label='Train Loss')
plt.plot(range(1, num_epochs+1), valid_loss_list, label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training and Validation Loss')
plt.legend()

plt.subplot(1, 2, 2)
# Plot training and validation accuracy
plt.plot(range(1, num_epochs+1), train_accuracy_list, label='Train Accuracy')
plt.plot(range(1, num_epochs+1), valid_accuracy_list, label='Validation␣
 ↪Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend()

plt.suptitle(model_name)
plt.show()
```



[20]:

```
# Initialize lists to store true labels and predicted labels
true_labels = []
predicted_labels = []


# Validation loop
model.eval()
with torch.no_grad():
    for images, labels in valid_loader:
        images = images.to(device)
        labels = labels.to(device)

        outputs = model(images)
        loss = criterion(outputs, labels)

        running_loss += loss.item()
        _, predicted = torch.max(outputs, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

        # Append true labels and predicted labels
        true_labels.extend(labels.cpu().numpy())
        predicted_labels.extend(predicted.cpu().numpy())

valid_loss = running_loss / len(valid_loader)
valid_accuracy = correct / total
```

```
print(f"Epoch: {epoch+1}/{num_epochs} | "
      f"Train Loss: {train_loss:.4f} | Train Acc: {train_accuracy:.4f} | "
      f"Valid Loss: {valid_loss:.4f} | Valid Acc: {valid_accuracy:.4f}")
```

```
Epoch: 10/10 | Train Loss: 0.5808 | Train Acc: 0.8446 | Valid Loss: 1.3321 |
Valid Acc: 0.8170
```

[21]:

```
from sklearn.metrics import classification_report

# Generate classification report
classification_rep = classification_report(true_labels, predicted_labels)
print("Classification Report:")
print(classification_rep)
```

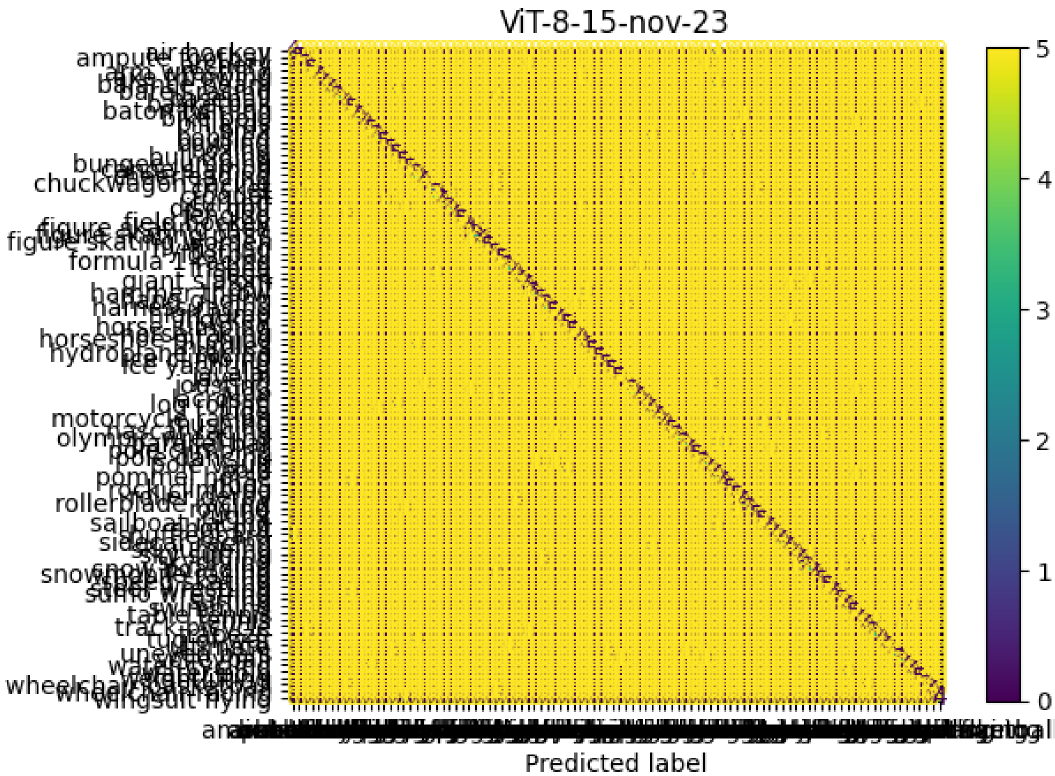
```
Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.80      0.80         5
           1       0.83      1.00      0.91         5
           2       0.75      0.60      0.67         5
           3       1.00      1.00      1.00         5
           4       1.00      0.80      0.89         5
           5       1.00      0.80      0.89         5
           6       1.00      0.80      0.89         5
           7       1.00      0.60      0.75         5
           8       0.67      0.80      0.73         5
           9       0.71      1.00      0.83         5
          10       1.00      0.80      0.89         5
          11       1.00      0.80      0.89         5
          12       0.83      1.00      0.91         5
          13       0.75      0.60      0.67         5
          14       0.83      1.00      0.91         5
          15       0.83      1.00      0.91         5
          16       0.71      1.00      0.83         5
          17       0.62      1.00      0.77         5
          18       1.00      1.00      1.00         5
          19       0.60      0.60      0.60         5
          20       0.80      0.80      0.80         5
          21       1.00      0.40      0.57         5
          22       0.60      0.60      0.60         5
          23       1.00      0.80      0.89         5
          24       0.75      0.60      0.67         5
          25       1.00      0.80      0.89         5
          26       1.00      1.00      1.00         5
          27       0.57      0.80      0.67         5
          28       0.75      0.60      0.67         5
          29       0.80      0.80      0.80         5
          30       1.00      1.00      1.00         5
```

31	0.42	1.00	0.59	5
32	1.00	1.00	1.00	5
33	0.75	0.60	0.67	5
34	0.80	0.80	0.80	5
35	1.00	0.80	0.89	5
36	0.57	0.80	0.67	5
37	0.60	0.60	0.60	5
38	0.83	1.00	0.91	5
39	1.00	0.60	0.75	5
40	0.71	1.00	0.83	5
41	1.00	0.80	0.89	5
42	0.83	1.00	0.91	5
43	0.71	1.00	0.83	5
44	0.67	0.80	0.73	5
45	0.80	0.80	0.80	5
46	1.00	1.00	1.00	5
47	1.00	1.00	1.00	5
48	0.83	1.00	0.91	5
49	1.00	1.00	1.00	5
50	0.83	1.00	0.91	5
51	0.50	0.40	0.44	5
52	1.00	0.60	0.75	5
53	0.80	0.80	0.80	5
54	1.00	0.80	0.89	5
55	0.67	0.80	0.73	5
56	1.00	0.80	0.89	5
57	1.00	0.80	0.89	5
58	1.00	1.00	1.00	5
59	1.00	0.60	0.75	5
60	0.75	0.60	0.67	5
61	1.00	0.80	0.89	5
62	0.83	1.00	0.91	5
63	0.60	0.60	0.60	5
64	0.80	0.80	0.80	5
65	0.50	0.80	0.62	5
66	0.80	0.80	0.80	5
67	1.00	1.00	1.00	5
68	0.71	1.00	0.83	5
69	0.80	0.80	0.80	5
70	1.00	0.60	0.75	5
71	0.83	1.00	0.91	5
72	0.83	1.00	0.91	5
73	0.80	0.80	0.80	5
74	1.00	0.80	0.89	5
75	0.80	0.80	0.80	5
76	0.67	0.80	0.73	5
77	0.80	0.80	0.80	5
78	1.00	0.80	0.89	5

```
          79       0.60      0.60      0.60         5
          80       1.00      0.80      0.89         5
          81       0.71      1.00      0.83         5
          82       1.00      1.00      1.00         5
          83       1.00      0.80      0.89         5
          84       0.83      1.00      0.91         5
          85       0.80      0.80      0.80         5
          86       0.62      1.00      0.77         5
          87       1.00      0.80      0.89         5
          88       0.67      0.80      0.73         5
          89       0.75      0.60      0.67         5
          90       1.00      0.80      0.89         5
          91       1.00      0.80      0.89         5
          92       1.00      0.40      0.57         5
          93       0.75      0.60      0.67         5
          94       1.00      0.80      0.89         5
          95       0.80      0.80      0.80         5
          96       0.83      1.00      0.91         5
          97       1.00      0.60      0.75         5
          98       1.00      0.80      0.89         5
          99       1.00      0.80      0.89         5

    accuracy                           0.82       500
   macro avg       0.84      0.82      0.82       500
weighted avg       0.84      0.82      0.82       500
```

[22]:
```python
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 5))

# Generate confusion matrix
cm = confusion_matrix(true_labels, predicted_labels)
cm_display = ConfusionMatrixDisplay(confusion_matrix=cm,
 ↪display_labels=valid_dataset.classes)
cm_display.plot()

plt.title(model_name)



plt.show()
```

```
<Figure size 800x500 with 0 Axes>
```


ViT-8-15-nov-23
True label
Predicted label
5
4
3
2
1
0