

# SL Project - Feature extraction

December 14, 2022

```
[1]: import torchvision
import torch
import torch.nn as nn
from torchsummary import summary
```

```
[2]: from torchvision import transforms, datasets
import matplotlib.pyplot as plt
from tqdm import tqdm
```

```
[3]: import pandas as pd
```

```
[4]: import os
import pickle
import seaborn as sns
import matplotlib.pyplot as plt
```

```
[5]: sns.set_style("darkgrid")
```

```
[6]: import cv2

image = cv2.imread("../data/asl_alphabet_train/A/A1.jpg")
print(image.shape)  # image dimensions

(200, 200, 3)
```

```
[7]: transform = transforms.Compose([
    transforms.Resize(256),
    transforms.RandomCrop(224),
    transforms.ToTensor()
])
```

```
[8]: PATH = "../data/asl_alphabet_train/"
```

```
[9]: dataset = datasets.ImageFolder(PATH, transform=transform)
```

```
[10]: n = len(dataset)
```

```
[11]: print(n)
```

87000

```
[12]: torch.manual_seed(1)
      indices = torch.randperm(n)
```

```
[13]: test_proportion = 0.2 # 20 percent of data used for testing
      test_size = int(n * test_proportion)
```

```
[14]: train_dataset = torch.utils.data.Subset(dataset, indices[test_size:])
      test_dataset = torch.utils.data.Subset(dataset, indices[:test_size])
```

```
[15]: len(train_dataset)
```

[15]: 69600

```
[16]: len(test_dataset)
```

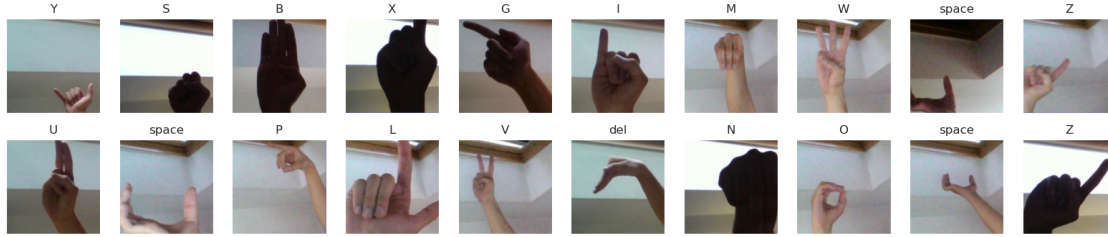
[16]: 17400

```
[17]: train_dataloader = torch.utils.data.DataLoader(dataset=train_dataset,
                                                    batch_size=32,
                                                    shuffle=True,
                                                    num_workers=4)

      test_dataloader = torch.utils.data.DataLoader(dataset=test_dataset,
                                                    batch_size=32,
                                                    shuffle=False,
                                                    num_workers=4)
```

```
[18]: classes = dataset.classes
```

```
[19]: cols = 10
      rows = 2
      fig, ax = plt.subplots(rows, cols, figsize=(20, 4))
      i = 0
      for img, label in train_dataloader:
          plt.subplot(rows, cols, i + 1)
          plt.imshow(img[0].permute(1, 2, 0))
          plt.xticks(())
          plt.yticks(())
          plt.title(classes[label[0]])
          i += 1
      if i == 20:
          break
```



```
[20]: model = torch.hub.load(repo_or_dir='pytorch/vision:v0.10.0', model='googlenet',
                             weights='GoogLeNet_Weights.IMAGENET1K_V1')
```

Using cache found in /home/rao.ans/.cache/torch/hub/pytorch\_vision\_v0.10.0

```
[21]: # freeze all but last few layers
i = 0
for param in model.parameters():
    if i == 162:
        break
    param.requires_grad = False
    i += 1
```

```
[22]: model.fc = torch.nn.Linear(model.fc.in_features, len(classes))
```

```
[23]: criterion = torch.nn.CrossEntropyLoss()
```

```
[24]: optimizer = torch.optim.Adam(model.parameters(), lr=3e-4, weight_decay=0.001)
```

```
[25]: DEVICE = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
```

```
[26]: if torch.cuda.is_available():
    model.cuda()
```

```
[27]: summary(model, (3, 224, 224))
```

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 112, 112]	9,408
BatchNorm2d-2	[-1, 64, 112, 112]	128
BasicConv2d-3	[-1, 64, 112, 112]	0
MaxPool2d-4	[-1, 64, 56, 56]	0
Conv2d-5	[-1, 64, 56, 56]	4,096
BatchNorm2d-6	[-1, 64, 56, 56]	128
BasicConv2d-7	[-1, 64, 56, 56]	0
Conv2d-8	[-1, 192, 56, 56]	110,592
BatchNorm2d-9	[-1, 192, 56, 56]	384
BasicConv2d-10	[-1, 192, 56, 56]	0

MaxPool2d-11	[-1, 192, 28, 28]	0
Conv2d-12	[-1, 64, 28, 28]	12,288
BatchNorm2d-13	[-1, 64, 28, 28]	128
BasicConv2d-14	[-1, 64, 28, 28]	0
Conv2d-15	[-1, 96, 28, 28]	18,432
BatchNorm2d-16	[-1, 96, 28, 28]	192
BasicConv2d-17	[-1, 96, 28, 28]	0
Conv2d-18	[-1, 128, 28, 28]	110,592
BatchNorm2d-19	[-1, 128, 28, 28]	256
BasicConv2d-20	[-1, 128, 28, 28]	0
Conv2d-21	[-1, 16, 28, 28]	3,072
BatchNorm2d-22	[-1, 16, 28, 28]	32
BasicConv2d-23	[-1, 16, 28, 28]	0
Conv2d-24	[-1, 32, 28, 28]	4,608
BatchNorm2d-25	[-1, 32, 28, 28]	64
BasicConv2d-26	[-1, 32, 28, 28]	0
MaxPool2d-27	[-1, 192, 28, 28]	0
Conv2d-28	[-1, 32, 28, 28]	6,144
BatchNorm2d-29	[-1, 32, 28, 28]	64
BasicConv2d-30	[-1, 32, 28, 28]	0
Inception-31	[-1, 256, 28, 28]	0
Conv2d-32	[-1, 128, 28, 28]	32,768
BatchNorm2d-33	[-1, 128, 28, 28]	256
BasicConv2d-34	[-1, 128, 28, 28]	0
Conv2d-35	[-1, 128, 28, 28]	32,768
BatchNorm2d-36	[-1, 128, 28, 28]	256
BasicConv2d-37	[-1, 128, 28, 28]	0
Conv2d-38	[-1, 192, 28, 28]	221,184
BatchNorm2d-39	[-1, 192, 28, 28]	384
BasicConv2d-40	[-1, 192, 28, 28]	0
Conv2d-41	[-1, 32, 28, 28]	8,192
BatchNorm2d-42	[-1, 32, 28, 28]	64
BasicConv2d-43	[-1, 32, 28, 28]	0
Conv2d-44	[-1, 96, 28, 28]	27,648
BatchNorm2d-45	[-1, 96, 28, 28]	192
BasicConv2d-46	[-1, 96, 28, 28]	0
MaxPool2d-47	[-1, 256, 28, 28]	0
Conv2d-48	[-1, 64, 28, 28]	16,384
BatchNorm2d-49	[-1, 64, 28, 28]	128
BasicConv2d-50	[-1, 64, 28, 28]	0
Inception-51	[-1, 480, 28, 28]	0
MaxPool2d-52	[-1, 480, 14, 14]	0
Conv2d-53	[-1, 192, 14, 14]	92,160
BatchNorm2d-54	[-1, 192, 14, 14]	384
BasicConv2d-55	[-1, 192, 14, 14]	0
Conv2d-56	[-1, 96, 14, 14]	46,080
BatchNorm2d-57	[-1, 96, 14, 14]	192
BasicConv2d-58	[-1, 96, 14, 14]	0

Conv2d-59	[-1, 208, 14, 14]	179,712
BatchNorm2d-60	[-1, 208, 14, 14]	416
BasicConv2d-61	[-1, 208, 14, 14]	0
Conv2d-62	[-1, 16, 14, 14]	7,680
BatchNorm2d-63	[-1, 16, 14, 14]	32
BasicConv2d-64	[-1, 16, 14, 14]	0
Conv2d-65	[-1, 48, 14, 14]	6,912
BatchNorm2d-66	[-1, 48, 14, 14]	96
BasicConv2d-67	[-1, 48, 14, 14]	0
MaxPool2d-68	[-1, 480, 14, 14]	0
Conv2d-69	[-1, 64, 14, 14]	30,720
BatchNorm2d-70	[-1, 64, 14, 14]	128
BasicConv2d-71	[-1, 64, 14, 14]	0
Inception-72	[-1, 512, 14, 14]	0
Conv2d-73	[-1, 160, 14, 14]	81,920
BatchNorm2d-74	[-1, 160, 14, 14]	320
BasicConv2d-75	[-1, 160, 14, 14]	0
Conv2d-76	[-1, 112, 14, 14]	57,344
BatchNorm2d-77	[-1, 112, 14, 14]	224
BasicConv2d-78	[-1, 112, 14, 14]	0
Conv2d-79	[-1, 224, 14, 14]	225,792
BatchNorm2d-80	[-1, 224, 14, 14]	448
BasicConv2d-81	[-1, 224, 14, 14]	0
Conv2d-82	[-1, 24, 14, 14]	12,288
BatchNorm2d-83	[-1, 24, 14, 14]	48
BasicConv2d-84	[-1, 24, 14, 14]	0
Conv2d-85	[-1, 64, 14, 14]	13,824
BatchNorm2d-86	[-1, 64, 14, 14]	128
BasicConv2d-87	[-1, 64, 14, 14]	0
MaxPool2d-88	[-1, 512, 14, 14]	0
Conv2d-89	[-1, 64, 14, 14]	32,768
BatchNorm2d-90	[-1, 64, 14, 14]	128
BasicConv2d-91	[-1, 64, 14, 14]	0
Inception-92	[-1, 512, 14, 14]	0
Conv2d-93	[-1, 128, 14, 14]	65,536
BatchNorm2d-94	[-1, 128, 14, 14]	256
BasicConv2d-95	[-1, 128, 14, 14]	0
Conv2d-96	[-1, 128, 14, 14]	65,536
BatchNorm2d-97	[-1, 128, 14, 14]	256
BasicConv2d-98	[-1, 128, 14, 14]	0
Conv2d-99	[-1, 256, 14, 14]	294,912
BatchNorm2d-100	[-1, 256, 14, 14]	512
BasicConv2d-101	[-1, 256, 14, 14]	0
Conv2d-102	[-1, 24, 14, 14]	12,288
BatchNorm2d-103	[-1, 24, 14, 14]	48
BasicConv2d-104	[-1, 24, 14, 14]	0
Conv2d-105	[-1, 64, 14, 14]	13,824
BatchNorm2d-106	[-1, 64, 14, 14]	128

BasicConv2d-107	[-1, 64, 14, 14]	0
MaxPool2d-108	[-1, 512, 14, 14]	0
Conv2d-109	[-1, 64, 14, 14]	32,768
BatchNorm2d-110	[-1, 64, 14, 14]	128
BasicConv2d-111	[-1, 64, 14, 14]	0
Inception-112	[-1, 512, 14, 14]	0
Conv2d-113	[-1, 112, 14, 14]	57,344
BatchNorm2d-114	[-1, 112, 14, 14]	224
BasicConv2d-115	[-1, 112, 14, 14]	0
Conv2d-116	[-1, 144, 14, 14]	73,728
BatchNorm2d-117	[-1, 144, 14, 14]	288
BasicConv2d-118	[-1, 144, 14, 14]	0
Conv2d-119	[-1, 288, 14, 14]	373,248
BatchNorm2d-120	[-1, 288, 14, 14]	576
BasicConv2d-121	[-1, 288, 14, 14]	0
Conv2d-122	[-1, 32, 14, 14]	16,384
BatchNorm2d-123	[-1, 32, 14, 14]	64
BasicConv2d-124	[-1, 32, 14, 14]	0
Conv2d-125	[-1, 64, 14, 14]	18,432
BatchNorm2d-126	[-1, 64, 14, 14]	128
BasicConv2d-127	[-1, 64, 14, 14]	0
MaxPool2d-128	[-1, 512, 14, 14]	0
Conv2d-129	[-1, 64, 14, 14]	32,768
BatchNorm2d-130	[-1, 64, 14, 14]	128
BasicConv2d-131	[-1, 64, 14, 14]	0
Inception-132	[-1, 528, 14, 14]	0
Conv2d-133	[-1, 256, 14, 14]	135,168
BatchNorm2d-134	[-1, 256, 14, 14]	512
BasicConv2d-135	[-1, 256, 14, 14]	0
Conv2d-136	[-1, 160, 14, 14]	84,480
BatchNorm2d-137	[-1, 160, 14, 14]	320
BasicConv2d-138	[-1, 160, 14, 14]	0
Conv2d-139	[-1, 320, 14, 14]	460,800
BatchNorm2d-140	[-1, 320, 14, 14]	640
BasicConv2d-141	[-1, 320, 14, 14]	0
Conv2d-142	[-1, 32, 14, 14]	16,896
BatchNorm2d-143	[-1, 32, 14, 14]	64
BasicConv2d-144	[-1, 32, 14, 14]	0
Conv2d-145	[-1, 128, 14, 14]	36,864
BatchNorm2d-146	[-1, 128, 14, 14]	256
BasicConv2d-147	[-1, 128, 14, 14]	0
MaxPool2d-148	[-1, 528, 14, 14]	0
Conv2d-149	[-1, 128, 14, 14]	67,584
BatchNorm2d-150	[-1, 128, 14, 14]	256
BasicConv2d-151	[-1, 128, 14, 14]	0
Inception-152	[-1, 832, 14, 14]	0
MaxPool2d-153	[-1, 832, 7, 7]	0
Conv2d-154	[-1, 256, 7, 7]	212,992

BatchNorm2d-155	[-1, 256, 7, 7]	512
BasicConv2d-156	[-1, 256, 7, 7]	0
Conv2d-157	[-1, 160, 7, 7]	133,120
BatchNorm2d-158	[-1, 160, 7, 7]	320
BasicConv2d-159	[-1, 160, 7, 7]	0
Conv2d-160	[-1, 320, 7, 7]	460,800
BatchNorm2d-161	[-1, 320, 7, 7]	640
BasicConv2d-162	[-1, 320, 7, 7]	0
Conv2d-163	[-1, 32, 7, 7]	26,624
BatchNorm2d-164	[-1, 32, 7, 7]	64
BasicConv2d-165	[-1, 32, 7, 7]	0
Conv2d-166	[-1, 128, 7, 7]	36,864
BatchNorm2d-167	[-1, 128, 7, 7]	256
BasicConv2d-168	[-1, 128, 7, 7]	0
MaxPool2d-169	[-1, 832, 7, 7]	0
Conv2d-170	[-1, 128, 7, 7]	106,496
BatchNorm2d-171	[-1, 128, 7, 7]	256
BasicConv2d-172	[-1, 128, 7, 7]	0
Inception-173	[-1, 832, 7, 7]	0
Conv2d-174	[-1, 384, 7, 7]	319,488
BatchNorm2d-175	[-1, 384, 7, 7]	768
BasicConv2d-176	[-1, 384, 7, 7]	0
Conv2d-177	[-1, 192, 7, 7]	159,744
BatchNorm2d-178	[-1, 192, 7, 7]	384
BasicConv2d-179	[-1, 192, 7, 7]	0
Conv2d-180	[-1, 384, 7, 7]	663,552
BatchNorm2d-181	[-1, 384, 7, 7]	768
BasicConv2d-182	[-1, 384, 7, 7]	0
Conv2d-183	[-1, 48, 7, 7]	39,936
BatchNorm2d-184	[-1, 48, 7, 7]	96
BasicConv2d-185	[-1, 48, 7, 7]	0
Conv2d-186	[-1, 128, 7, 7]	55,296
BatchNorm2d-187	[-1, 128, 7, 7]	256
BasicConv2d-188	[-1, 128, 7, 7]	0
MaxPool2d-189	[-1, 832, 7, 7]	0
Conv2d-190	[-1, 128, 7, 7]	106,496
BatchNorm2d-191	[-1, 128, 7, 7]	256
BasicConv2d-192	[-1, 128, 7, 7]	0
Inception-193	[-1, 1024, 7, 7]	0
AdaptiveAvgPool2d-194	[-1, 1024, 1, 1]	0
Dropout-195	[-1, 1024]	0
Linear-196	[-1, 29]	29,725

=====

Total params: 5,629,629

Trainable params: 232,061

Non-trainable params: 5,397,568

-----

Input size (MB): 0.57

Forward/backward pass size (MB): 94.10  
Params size (MB): 21.48  
Estimated Total Size (MB): 116.15  
-----

```
[28]: def train_one_epoch(train_loader, model, device, optimizer, log_interval,
    ↪epoch):
    model.train()
    losses = []
    counter = []

    for i, (img, label) in enumerate(train_loader):
        img, label = img.to(device), label.to(device)
        optimizer.zero_grad()
        output = model(img)
        loss = criterion(output, label)
        loss.backward()
        optimizer.step()
        if (i+1) % log_interval == 0:
            losses.append(loss.item())

    return losses
```

```
[29]: def test_one_epoch(test_loader, model, device):
    model.eval()
    test_loss = 0
    num_correct = 0

    with torch.no_grad():
        for i, (img, label) in enumerate(test_loader):
            img, label = img.to(device), label.to(device)
            output = model(img)
            pred = torch.argmax(output, dim=1)
            num_correct += (pred == label).sum().item()
            test_loss += criterion(output, label).item()

    test_loss /= len(test_loader.dataset)
    return test_loss, num_correct
```

```
[30]: no_of_epochs = 30
```

```
[31]: lr = 0.01

log_interval = 100

train_losses = []
test_losses = []
```



```

test_correct = []
train_correct = []
for epoch in tqdm(range(no_of_epochs)):
    train_loss = train_one_epoch(train_dataloader, model, DEVICE, optimizer,
    log_interval, epoch)
    test_loss, num_correct = test_one_epoch(test_dataloader, model, DEVICE)
    _, num_correct_train = test_one_epoch(train_dataloader, model, DEVICE)

    # record results
    train_losses.extend(train_loss)
    test_losses.append(test_loss)
    test_correct.append(num_correct)
    train_correct.append(num_correct_train)

```

100%| | 30/30 [2:31:12<00:00, 302.40s/it]

```
[32]: SAVE_PATH = "../data/googlenet_asl_v1.pth"
```

```
[33]: torch.save(model, SAVE_PATH)
```

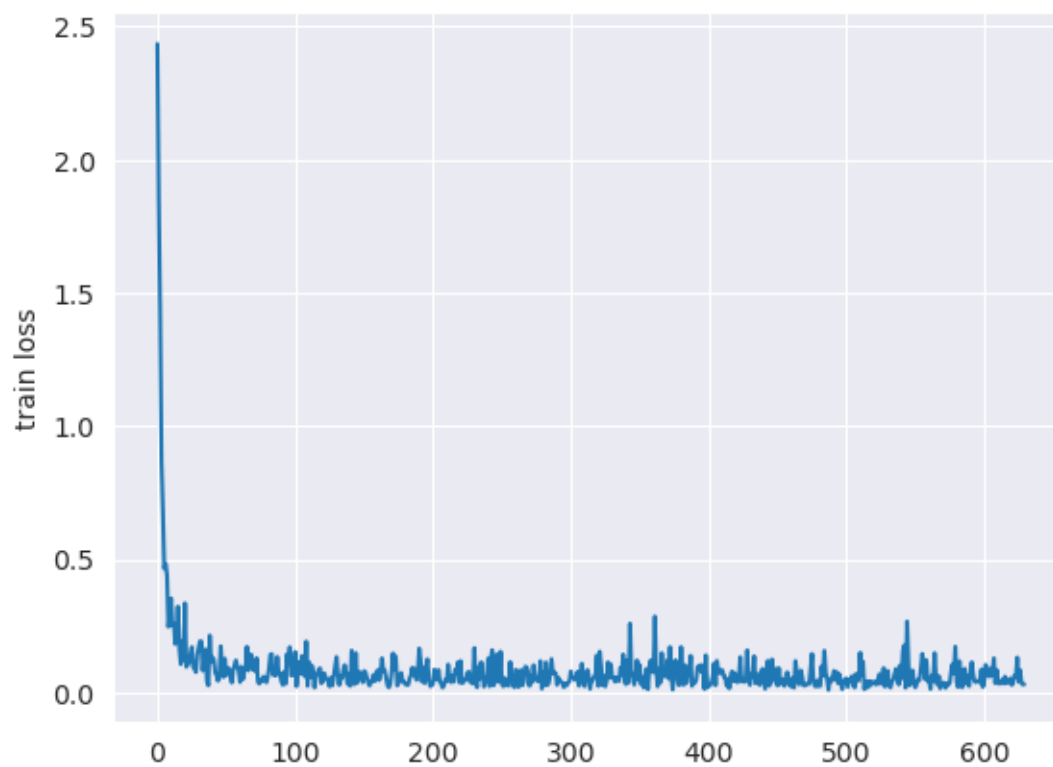
```
[34]: print(f"test accuracy: {test_correct[-1]/len(test_dataloader.dataset)}")
      print(f"train accuracy: {train_correct[-1]/len(train_dataloader.dataset)}")

```

test accuracy: 0.9947126436781609  
train accuracy: 0.9960919540229886

```
[35]: sns.lineplot(train_losses)
      plt.ylabel("train loss");

```



```
[36]: sns.lineplot(test_losses, color="orange")  
      plt.ylabel("test loss");
```

