SL Project - Feature Selection

December 14, 2022

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

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

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

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

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

[6]:
```python
import cv2

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

```
(200, 200, 3)
```

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

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

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

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

[11]:
```python
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]:

```python
train_losses = []
train_accuracies = []
val_losses = []
val_accuracies = []
```

[29]:

```python
step = 0
no_of_epochs = 30

for epoch in tqdm(range(no_of_epochs)):
    correct_train, total_train = 0, 0
    train_loss = 0
    model.train()
    for i, (images, labels) in enumerate(train_dataloader):
        step += 1
        images = images.to(device)
        labels = labels.to(device)
        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        total_train += labels.size(0)
        _, predicted = torch.max(outputs, dim=1)
        correct_train += (predicted == labels).sum().item()

        if step % 1000 == 0:
            print(f"epoch [{epoch + 1}]/[{no_of_epochs}]", end=" ")
            train_loss += loss.item()
            train_accuracy = (correct_train / total_train) * 100
            print(f"train accuracy: {train_accuracy}", end=" ")

            with torch.no_grad():
                correct_val, total_val = 0, 0
                val_loss = 0
                model.eval()
                for images, labels in test_dataloader:
                    images = images.to(device)
                    labels = labels.to(device)
                    outputs = model(images)

                    total_val += labels.size(0)
```

```
                _, predicted = torch.max(outputs, dim=1)
                correct_val += (predicted == labels).sum().item()

                val_loss += loss.item()

        val_accuracy = (correct_val / total_val) * 100
        print(f"val accuracy: {val_accuracy}")
        train_losses.append(train_loss / total_train)
        val_losses.append(val_loss / total_val)
        train_accuracies.append(train_accuracy)
        val_accuracies.append(val_accuracy)
```

```
  0%|          | 0/30 [00:00<?, ?it/s]

epoch [1]/[30] train accuracy: 79.10625 val accuracy: 96.21264367816092
epoch [1]/[30] train accuracy: 87.665625 val accuracy: 97.82758620689656

  3%|          | 1/30 [02:47<1:20:47, 167.14s/it]

epoch [2]/[30] train accuracy: 97.0909090909091 val accuracy: 98.28735632183908
epoch [2]/[30] train accuracy: 97.5513698630137 val accuracy: 97.73563218390805

  7%|          | 2/30 [05:40<1:19:35, 170.56s/it]

epoch [3]/[30] train accuracy: 97.79807692307692 val accuracy: 98.93103448275862
epoch [3]/[30] train accuracy: 98.23674242424244 val accuracy: 98.8103448275862

 10%|          | 3/30 [08:32<1:17:07, 171.40s/it]

epoch [4]/[30] train accuracy: 97.98684210526316 val accuracy: 99.13218390804597
epoch [4]/[30] train accuracy: 98.40254237288136 val accuracy: 98.94827586206897

 13%|          | 4/30 [11:24<1:14:27, 171.82s/it]

epoch [5]/[30] train accuracy: 98.10416666666667 val accuracy: 99.05172413793103
epoch [5]/[30] train accuracy: 98.60817307692308 val accuracy: 98.24712643678161

 17%|          | 5/30 [14:22<1:12:23, 173.74s/it]

epoch [6]/[30] train accuracy: 97.8 val accuracy: 99.01149425287356
epoch [6]/[30] train accuracy: 98.73611111111111 val accuracy: 98.86206896551725
epoch [6]/[30] train accuracy: 98.72205882352941 val accuracy: 98.84482758620689

 20%|          | 6/30 [17:46<1:13:37, 184.07s/it]

epoch [7]/[30] train accuracy: 98.16118421052632 val accuracy: 99.06896551724138
epoch [7]/[30] train accuracy: 98.49198717948718 val accuracy: 98.6551724137931

 23%|          | 7/30 [20:41<1:09:28, 181.22s/it]

epoch [8]/[30] train accuracy: 98.33870967741936 val accuracy: 99.10919540229885
epoch [8]/[30] train accuracy: 98.67605633802818 val accuracy: 99.01724137931035

 27%|          | 8/30 [23:36<1:05:44, 179.30s/it]
```

```
epoch [9]/[30] train accuracy: 98.33854166666667 val accuracy: 99.35632183908046
epoch [9]/[30] train accuracy: 98.697265625 val accuracy: 98.93103448275862

 30%|        | 9/30 [26:31<1:02:16, 177.94s/it]

epoch [10]/[30] train accuracy: 98.26470588235294 val accuracy:
99.21264367816092
epoch [10]/[30] train accuracy: 98.71271929824562 val accuracy:
99.04597701149426

 33%|        | 10/30 [29:26<59:00, 177.02s/it]

epoch [11]/[30] train accuracy: 97.98750000000001 val accuracy: 99.2183908045977
epoch [11]/[30] train accuracy: 98.7825 val accuracy: 98.74712643678161

 37%|        | 11/30 [31:02<48:09, 152.11s/it]

epoch [12]/[30] train accuracy: 98.125 val accuracy: 98.9080459770115
epoch [12]/[30] train accuracy: 98.96511627906976 val accuracy:
99.12068965517241
epoch [12]/[30] train accuracy: 98.9382530120482 val accuracy: 98.92528735632183

 40%|        | 12/30 [32:52<41:46, 139.25s/it]

epoch [13]/[30] train accuracy: 98.09375 val accuracy: 99.25287356321839
epoch [13]/[30] train accuracy: 98.60197368421053 val accuracy:
98.85057471264368

 43%|       | 13/30 [34:26<35:37, 125.73s/it]

epoch [14]/[30] train accuracy: 98.40086206896552 val accuracy:
99.27011494252874
epoch [14]/[30] train accuracy: 98.82246376811594 val accuracy:
98.49425287356321

 47%|       | 14/30 [36:01<31:01, 116.31s/it]

epoch [15]/[30] train accuracy: 98.21022727272727 val accuracy:
99.25862068965517
epoch [15]/[30] train accuracy: 98.77016129032258 val accuracy:
99.19540229885058

 50%|       | 15/30 [37:35<27:25, 109.69s/it]

epoch [16]/[30] train accuracy: 98.09166666666667 val accuracy:
99.24137931034483
epoch [16]/[30] train accuracy: 98.81818181818181 val accuracy:
99.12068965517241

 53%|       | 16/30 [39:09<24:29, 104.97s/it]

epoch [17]/[30] train accuracy: 98.015625 val accuracy: 99.20114942528735
epoch [17]/[30] train accuracy: 98.89583333333334 val accuracy:
99.13218390804597

 57%|       | 17/30 [40:43<22:01, 101.66s/it]
```

```
epoch [18]/[30] train accuracy: 96.875 val accuracy: 98.77011494252874
epoch [18]/[30] train accuracy: 98.89024390243902 val accuracy: 99.0
epoch [18]/[30] train accuracy: 98.9320987654321 val accuracy: 98.78735632183908

 60%|       | 18/30 [42:32<20:47, 103.92s/it]

epoch [19]/[30] train accuracy: 98.07720588235294 val accuracy:
99.36781609195403
epoch [19]/[30] train accuracy: 98.58783783783784 val accuracy: 99.0

 63%|       | 19/30 [44:07<18:31, 101.07s/it]

epoch [20]/[30] train accuracy: 98.12037037037037 val accuracy: 99.2816091954023
epoch [20]/[30] train accuracy: 98.71082089552239 val accuracy: 99.183908045977

 67%|       | 20/30 [45:41<16:30, 99.04s/it]

epoch [21]/[30] train accuracy: 98.30624999999999 val accuracy:
99.32183908045977
epoch [21]/[30] train accuracy: 98.82291666666667 val accuracy:
99.02873563218391

 70%|       | 21/30 [47:15<14:38, 97.62s/it]

epoch [22]/[30] train accuracy: 98.3076923076923 val accuracy: 99.40229885057471
epoch [22]/[30] train accuracy: 98.94339622641509 val accuracy: 98.6896551724138

 73%|       | 22/30 [48:50<12:53, 96.73s/it]

epoch [23]/[30] train accuracy: 98.02083333333333 val accuracy:
99.27586206896551
epoch [23]/[30] train accuracy: 99.05163043478261 val accuracy:
98.94827586206897
epoch [23]/[30] train accuracy: 99.10174418604652 val accuracy:
98.94827586206897

 77%|       | 23/30 [50:40<11:45, 100.82s/it]

epoch [24]/[30] train accuracy: 98.10576923076924 val accuracy: 99.3103448275862
epoch [24]/[30] train accuracy: 98.59177215189874 val accuracy: 99.2816091954023

 80%|       | 24/30 [52:15<09:54, 99.11s/it]

epoch [25]/[30] train accuracy: 98.34375 val accuracy: 99.36781609195403
epoch [25]/[30] train accuracy: 98.76041666666666 val accuracy:
99.03448275862068

 83%|       | 25/30 [53:50<08:09, 97.87s/it]

epoch [26]/[30] train accuracy: 98.205 val accuracy: 99.27586206896551
epoch [26]/[30] train accuracy: 98.79423076923078 val accuracy:
99.13793103448276

 87%|       | 26/30 [55:25<06:27, 96.89s/it]
```

```
epoch [27]/[30] train accuracy: 98.3263888888889 val accuracy: 99.316091954023
epoch [27]/[30] train accuracy: 98.88577586206897 val accuracy:
99.16091954022988
```

```
 90%|      | 27/30 [57:00<04:49, 96.34s/it]
```

```
epoch [28]/[30] train accuracy: 98.06818181818183 val accuracy:
99.45402298850576
epoch [28]/[30] train accuracy: 98.89950980392157 val accuracy:
99.24712643678161
```

```
 93%|     | 28/30 [58:35<03:11, 95.91s/it]
```

```
epoch [29]/[30] train accuracy: 97.5 val accuracy: 99.24137931034483
epoch [29]/[30] train accuracy: 99.02272727272728 val accuracy: 99.1264367816092
epoch [29]/[30] train accuracy: 99.07886904761905 val accuracy:
99.13218390804597
```

```
 97%|     | 29/30 [1:00:25<01:40, 100.22s/it]
```

```
epoch [30]/[30] train accuracy: 98.17567567567568 val accuracy: 99.3103448275862
epoch [30]/[30] train accuracy: 98.69805194805194 val accuracy:
99.10344827586208
```

```
100%|     | 30/30 [1:02:00<00:00, 124.02s/it]
```

[30]:
```
SAVE_PATH = "../../data/googlenet_asl_v1.pth"
```

[31]:
```
torch.save(model, SAVE_PATH)
```

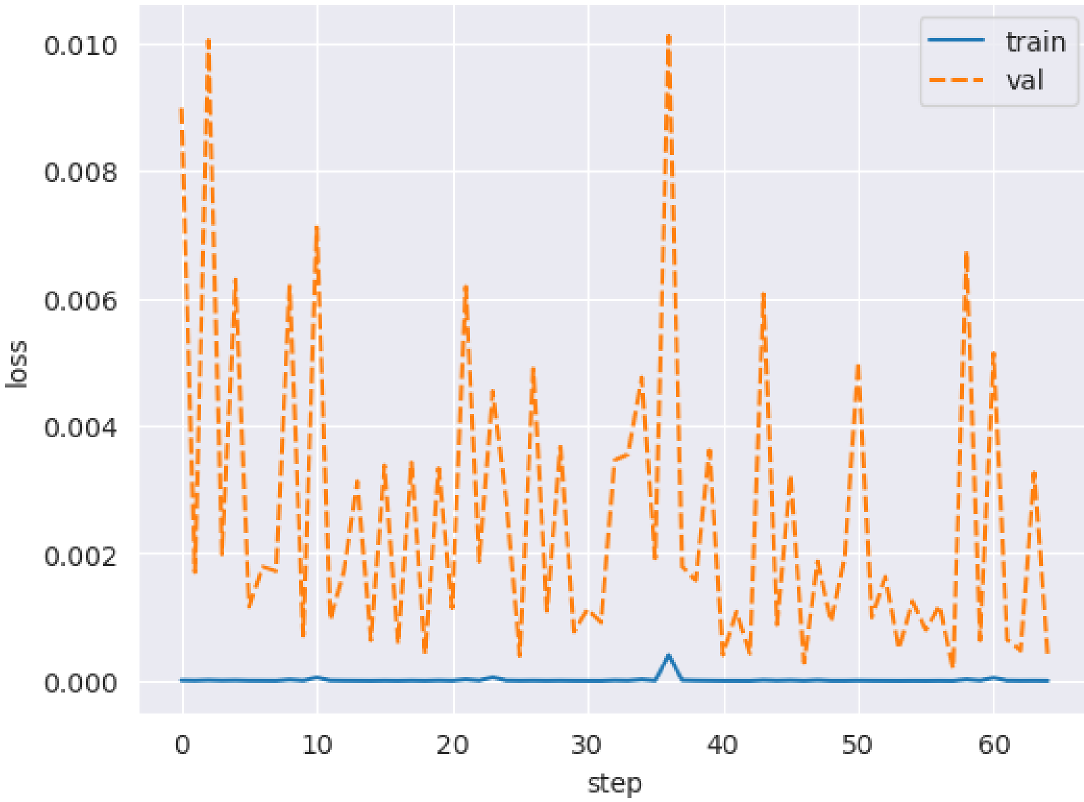
[32]:
```
with open('../../data/googlenet_asl_v1_train_losses.pkl', 'wb') as f:
    pickle.dump(train_losses, f)
```

[33]:
```
with open('../../data/googlenet_asl_v1_val_losses.pkl', 'wb') as f:
    pickle.dump(val_losses, f)
```

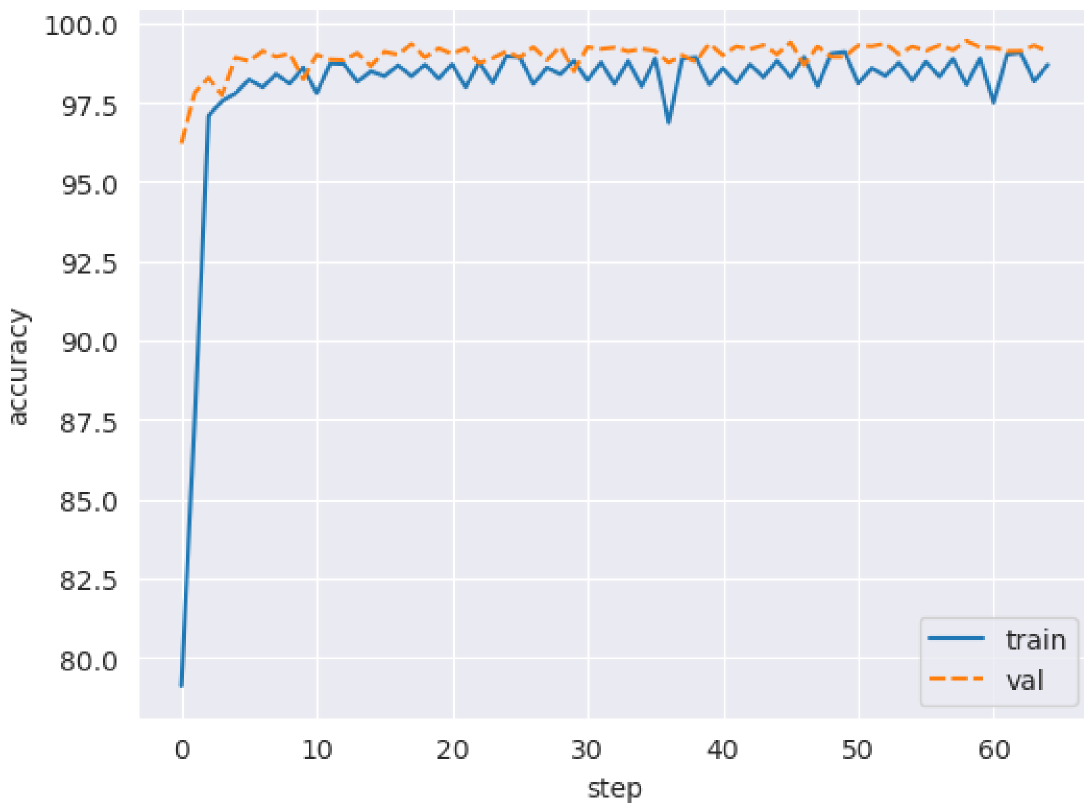
[34]:
```
with open('../../data/googlenet_asl_v1_train_accuracies.pkl', 'wb') as f:
    pickle.dump(train_accuracies, f)
```

[35]:
```
with open('../../data/googlenet_asl_v1_val_accuracies.pkl', 'wb') as f:
    pickle.dump(val_accuracies, f)
```

[36]:
```
loss = pd.DataFrame({'train': train_losses, 'val': val_losses})
sns.lineplot(loss)
plt.xlabel("step")
plt.ylabel("loss");
```



[37]:
```
accuracy = pd.DataFrame({'train': train_accuracies, 'val': val_accuracies})
sns.lineplot(accuracy)
plt.xlabel("step")
plt.ylabel("accuracy");
```


100.0
97.5
95.0
92.5
90.0
87.5
85.0
82.5
80.0
accuracy
0
10
20
30
40
50
60
step
train
val