Loading and Wrangling Data

Loading in data using the functions that we made:

In [1]:
```
# importing the images
import data_loading as dt
import os
import gdown
import logging
from zipfile import ZipFile
import os
import numpy as np
from multiprocessing import Pool
from pathlib import Path
import concurrent.futures
import cv2

idArrays, imageArrays = [], []
for num in range(13):
    print(num)
    path = f"./batch{num}.zip"

    # We import images into np.arrays
    newPath = path.replace(".zip", f"/part_{num}")
    ids, images = dt.importImages(newPath)

    # Adding to arrays
    idArrays.append(ids)
    imageArrays.append(images)


totalIds = np.concatenate(idArrays) if len(idArrays) > 1 else idArrays[0]
totalImages = (
    np.concatenate(imageArrays) if len(imageArrays) > 1 else imageArrays[0]
    )

ids = totalIds
images= totalImages
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
```

Getting annotations and getting them into the correct order:

In [2]:
```
import pandas as pd
annotations = pd.read_csv("data/annotations.csv")
annotations.head()
```

Out[2]:

	position	image
0	standing	1
1	standing	2
2	standing	3
3	standing	4
4	standing	5

In [3]:
```
position_maps= {"standing": 0,
                "takedown1": 1,
                "takedown2": 2,
                "open_guard1": 3,
                "open_guard2": 4,
                "half_guard1": 5,
                "half_guard2": 6,
                "closed_guard1": 7,
                "closed_guard2": 8,
                "5050_guard": 9,
                "mount1": 10,
                "mount2": 11,
                "back1": 12,
                "back2": 13,
                "turtle1": 14,
                "turtle2": 15,
                "side_control1" : 16,
                "side_control2" : 17}


labels = []
for id in ids:
    labels.append(position_maps[annotations[annotations['image'] == id]['position']]
```

In [4]:
```
labels[0:20]
```

Out[4]:
```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

Converting images to tensors:

In [5]:
```
import torch
import numpy as np

# Converting into torch tensors
for i, img in enumerate(images):
    images[i] = torch.from_numpy(np.array(img))
```

Convolutional Neural Net: Transfer Learning Squeeze Net

Creating Neural Net

In [6]:
```
# Data argumentation
from torchvision import transforms
data_transforms = transforms.Compose([
        transforms.GaussianBlur(kernel_size=(3,3), sigma=(0.1, 5)),
        transforms.RandomHorizontalFlip(),
        transforms.RandomRotation(degrees=(0, 180))
])
```

In [7]:
```
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision.models as models

# Define the modified SqueezeNet model
class MySqueezeNet(nn.Module):
    def __init__(self, num_classes):
        super(MySqueezeNet, self).__init__()
        self.squeezenet = models.squeezenet1_1(pretrained=False)
        self.num_classes = num_classes
        self.squeezenet.num_classes = self.num_classes

        self.classifier = nn.Sequential(
            nn.Conv2d(1000, 512, 1, 1),
            nn.ReLU(),
            nn.Conv2d(512, 256, 1, 1),
            nn.ReLU(),
            # nn.Flatten(),
            # nn.Linear(256, self.num_classes)
            nn.Conv2d(256, self.num_classes, 1, 1)
        )

    def forward(self, x):
        x = self.squeezenet.features(x)
        x = self.squeezenet.classifier(x)
        x = self.classifier(x)
        return x.view(x.size(0), self.num_classes)

# Create an instance of the modified SqueezeNet model
squeeze_model = MySqueezeNet(num_classes=18)
state_dict = torch.load('weights/squeezenet1_1-f364aa15.pth')

# Copy the weights from the pre-trained model to the modified model
pretrained_dict = {k: v for k, v in state_dict.items() if k.startswith('features')
model_dict = squeeze_model.squeezenet.state_dict()
model_dict.update(pretrained_dict)
squeeze_model.squeezenet.load_state_dict(model_dict)
```

```
# Freeze the parameters of the pre-trained layers
for param in squeeze_model.squeezenet.parameters():
    param.requires_grad = False

# for param in squeeze_model.squeezenet.classifier[1].parameters():
#     param.requires_grad = True

# Define the optimizer for the new fully connected layer
lrate = 0.001
# optimizer = optim.Adam(squeeze_model.squeezenet.classifier[1].parameters(), lr=lr
optimizer = optim.Adam(squeeze_model.classifier.parameters(), lr=lrate)
```

```
C:\Users\leemingi\AppData\Local\anaconda3\lib\site-packages\torchvision\models\_util
s.py:208: UserWarning: The parameter 'pretrained' is deprecated since 0.13 and will
be removed in 0.15, please use 'weights' instead.
  warnings.warn(
C:\Users\leemingi\AppData\Local\anaconda3\lib\site-packages\torchvision\models\_util
s.py:223: UserWarning: Arguments other than a weight enum or `None` for 'weights' ar
e deprecated since 0.13 and will be removed in 0.15. The current behavior is equival
ent to passing `weights=None`.
  warnings.warn(msg)
```

Training Neural Net

Train-test split:

In [8]:
```
from sklearn.model_selection import train_test_split
train_X, test_X, train_y, test_y = train_test_split(
    images, labels, test_size=0.25, random_state=42)

train_X = torch.from_numpy(train_X)
train_X = torch.movedim(train_X, source=3, destination=1)

test_X = torch.from_numpy(test_X)
test_X = torch.movedim(test_X, source=3, destination=1)
```

In [9]:
```
from torch.utils.data import DataLoader, TensorDataset

# Hyperparameters
epochs = 300
lrate = 0.001
bsize = 32

# For reproducibility
torch.manual_seed(3)

# Cost Function
cost_fn = nn.CrossEntropyLoss()

# Initialize the model
net = squeeze_model

# Optimizer (Stochastic Gradient Descent)
optimizer = torch.optim.SGD(net.parameters(), lr=lrate)
```

```
# Make DataLoader
y_tensor = torch.Tensor(train_y)
train_loader = DataLoader(TensorDataset(train_X.type(torch.FloatTensor),
                                        y_tensor.type(torch.LongTensor)), batch_siz

# Training Loop
track_cost = np.zeros(epochs)

for epoch in range(epochs):
    cur_cost = 0.0

    for i, (inputs, labels) in enumerate(train_loader):
        # Transform the input data using our data augmentation strategies
        inputs = data_transforms(inputs)

        # Forward, backward, and optimize
        optimizer.zero_grad()
        outputs = net(inputs)
        cost = cost_fn(outputs, labels)  # CrossEntropyLoss already applies Softmax
        cost.backward()
        optimizer.step()

        cur_cost += cost.item()

    # Store the accumulated cost at each epoch
    track_cost[epoch] = cur_cost
    print(epoch/epochs, f"Epoch: {epoch} Cost: {cur_cost}")
```

```
0.0 Epoch: 0 Cost: 6760.960058093071
0.003333333333333335 Epoch: 1 Cost: 6007.229216098785
0.006666666666666667 Epoch: 2 Cost: 5807.298224806786
0.01 Epoch: 3 Cost: 5668.483103394508
0.013333333333333334 Epoch: 4 Cost: 5587.605333685875
0.016666666666666666 Epoch: 5 Cost: 5501.246820569038
0.02 Epoch: 6 Cost: 5449.751280546188
0.023333333333333334 Epoch: 7 Cost: 5391.028585195541
0.026666666666666667 Epoch: 8 Cost: 5329.137958526611
0.03 Epoch: 9 Cost: 5308.0824893713
0.03333333333333333 Epoch: 10 Cost: 5267.300131082535
0.036666666666666667 Epoch: 11 Cost: 5237.440890312195
0.04 Epoch: 12 Cost: 5214.242956519127
0.043333333333333335 Epoch: 13 Cost: 5166.403807759285
0.046666666666666667 Epoch: 14 Cost: 5139.665705442429
0.05 Epoch: 15 Cost: 5122.61977481842
0.05333333333333334 Epoch: 16 Cost: 5092.248592495918
0.056666666666666664 Epoch: 17 Cost: 5080.598364472389
0.06 Epoch: 18 Cost: 5059.067266941071
0.06333333333333334 Epoch: 19 Cost: 5036.939256072044
0.06666666666666667 Epoch: 20 Cost: 5020.250499010086
0.07 Epoch: 21 Cost: 5005.404866933823
0.07333333333333333 Epoch: 22 Cost: 4984.4669489860535
0.07666666666666666 Epoch: 23 Cost: 4963.361346721649
0.08 Epoch: 24 Cost: 4953.986559808254
0.08333333333333333 Epoch: 25 Cost: 4956.790878295898
```

0.08666666666666667 Epoch: 26 Cost: 4932.539506316185
0.09 Epoch: 27 Cost: 4905.356189489365
0.09333333333333334 Epoch: 28 Cost: 4899.21087372303
0.09666666666666666 Epoch: 29 Cost: 4872.4415756464
0.1 Epoch: 30 Cost: 4880.150162935257
0.10333333333333333 Epoch: 31 Cost: 4873.227182388306
0.10666666666666667 Epoch: 32 Cost: 4857.471678972244
0.11 Epoch: 33 Cost: 4842.562870264053
0.11333333333333333 Epoch: 34 Cost: 4840.5817476511
0.11666666666666667 Epoch: 35 Cost: 4820.109882116318
0.12 Epoch: 36 Cost: 4816.756395697594
0.12333333333333334 Epoch: 37 Cost: 4804.662709534168
0.12666666666666668 Epoch: 38 Cost: 4804.751940011978
0.13 Epoch: 39 Cost: 4796.047972977161
0.13333333333333333 Epoch: 40 Cost: 4776.395036458969
0.13666666666666666 Epoch: 41 Cost: 4773.523706197739
0.14 Epoch: 42 Cost: 4765.487584590912
0.14333333333333334 Epoch: 43 Cost: 4764.097178936005
0.14666666666666667 Epoch: 44 Cost: 4745.448634088039
0.15 Epoch: 45 Cost: 4746.729011356831
0.15333333333333332 Epoch: 46 Cost: 4733.955567002296
0.15666666666666668 Epoch: 47 Cost: 4740.845437169075
0.16 Epoch: 48 Cost: 4719.8640204668045
0.16333333333333333 Epoch: 49 Cost: 4710.324948370457
0.16666666666666666 Epoch: 50 Cost: 4703.027209043503
0.17 Epoch: 51 Cost: 4703.040988683701
0.17333333333333334 Epoch: 52 Cost: 4677.482635974884
0.17666666666666667 Epoch: 53 Cost: 4686.813736796379
0.18 Epoch: 54 Cost: 4696.36791819334
0.18333333333333332 Epoch: 55 Cost: 4686.0873918533325
0.18666666666666668 Epoch: 56 Cost: 4676.495685994625
0.19 Epoch: 57 Cost: 4675.961597502232
0.19333333333333333 Epoch: 58 Cost: 4663.100611746311
0.19666666666666666 Epoch: 59 Cost: 4644.954853355885
0.2 Epoch: 60 Cost: 4655.947023510933
0.20333333333333334 Epoch: 61 Cost: 4655.033487677574
0.20666666666666667 Epoch: 62 Cost: 4642.162995934486
0.21 Epoch: 63 Cost: 4648.3931975364685
0.21333333333333335 Epoch: 64 Cost: 4632.579499602318
0.21666666666666667 Epoch: 65 Cost: 4619.756344139576
0.22 Epoch: 66 Cost: 4621.46522963047
0.22333333333333333 Epoch: 67 Cost: 4611.323665201664
0.22666666666666666 Epoch: 68 Cost: 4610.6816465854645
0.23 Epoch: 69 Cost: 4612.815335035324
0.23333333333333334 Epoch: 70 Cost: 4617.906051814556
0.23666666666666666 Epoch: 71 Cost: 4598.375298798084
0.24 Epoch: 72 Cost: 4588.5441491007805
0.24333333333333335 Epoch: 73 Cost: 4589.42587274313
0.24666666666666667 Epoch: 74 Cost: 4596.684934794903
0.25 Epoch: 75 Cost: 4584.21719878912
0.25333333333333335 Epoch: 76 Cost: 4563.913330316544
0.25666666666666665 Epoch: 77 Cost: 4578.659813582897
0.26 Epoch: 78 Cost: 4576.322642087936
0.2633333333333333 Epoch: 79 Cost: 4571.197899878025
0.26666666666666666 Epoch: 80 Cost: 4555.696313202381
0.27 Epoch: 81 Cost: 4572.870288729668

0.2733333333333333 Epoch: 82 Cost: 4563.714366257191
0.27666666666666667 Epoch: 83 Cost: 4544.639844298363
0.28 Epoch: 84 Cost: 4536.490013837814
0.2833333333333333 Epoch: 85 Cost: 4546.432686805725
0.2866666666666667 Epoch: 86 Cost: 4540.182847201824
0.29 Epoch: 87 Cost: 4556.842894732952
0.29333333333333333 Epoch: 88 Cost: 4532.905566751957
0.2966666666666667 Epoch: 89 Cost: 4543.284554243088
0.3 Epoch: 90 Cost: 4520.050776481628
0.30333333333333334 Epoch: 91 Cost: 4512.914502620697
0.30666666666666664 Epoch: 92 Cost: 4518.432791411877
0.31 Epoch: 93 Cost: 4581.608615636826
0.31333333333333335 Epoch: 94 Cost: 4531.430371820927
0.31666666666666665 Epoch: 95 Cost: 4522.372952282429
0.32 Epoch: 96 Cost: 4531.944829583168
0.3233333333333333 Epoch: 97 Cost: 4506.632364273071
0.32666666666666666 Epoch: 98 Cost: 4520.897686898708
0.33 Epoch: 99 Cost: 4507.1246255636215
0.3333333333333333 Epoch: 100 Cost: 4514.722877383232
0.33666666666666667 Epoch: 101 Cost: 4531.415671348572
0.34 Epoch: 102 Cost: 4485.5562286973
0.3433333333333333 Epoch: 103 Cost: 4494.979943156242
0.3466666666666667 Epoch: 104 Cost: 4481.279260277748
0.35 Epoch: 105 Cost: 4485.517537415028
0.35333333333333333 Epoch: 106 Cost: 4483.471640050411
0.3566666666666667 Epoch: 107 Cost: 4480.042143642902
0.36 Epoch: 108 Cost: 4485.347831249237
0.36333333333333334 Epoch: 109 Cost: 4472.680375516415
0.36666666666666664 Epoch: 110 Cost: 4470.656073331833
0.37 Epoch: 111 Cost: 4462.095431804657
0.37333333333333335 Epoch: 112 Cost: 4455.990680754185
0.37666666666666665 Epoch: 113 Cost: 4473.183609545231
0.38 Epoch: 114 Cost: 4457.995449304581
0.38333333333333336 Epoch: 115 Cost: 4454.501186788082
0.38666666666666666 Epoch: 116 Cost: 4443.386462569237
0.39 Epoch: 117 Cost: 4452.36483502388
0.3933333333333333 Epoch: 118 Cost: 4445.780612707138
0.39666666666666667 Epoch: 119 Cost: 4431.02290314436
0.4 Epoch: 120 Cost: 4454.666296303272
0.4033333333333333 Epoch: 121 Cost: 4437.458249330521
0.4066666666666667 Epoch: 122 Cost: 4441.376124501228
0.41 Epoch: 123 Cost: 4427.531896412373
0.41333333333333333 Epoch: 124 Cost: 4425.470556855202
0.4166666666666667 Epoch: 125 Cost: 4418.321358323097
0.42 Epoch: 126 Cost: 4427.561573624611
0.42333333333333334 Epoch: 127 Cost: 4429.291622817516
0.4266666666666667 Epoch: 128 Cost: 4431.407096028328
0.43 Epoch: 129 Cost: 4417.361450254917
0.43333333333333335 Epoch: 130 Cost: 4423.145028650761
0.43666666666666665 Epoch: 131 Cost: 4413.909478068352
0.44 Epoch: 132 Cost: 4425.240535378456
0.44333333333333336 Epoch: 133 Cost: 4396.275132715702
0.44666666666666666 Epoch: 134 Cost: 4393.413942396641
0.45 Epoch: 135 Cost: 4401.965200662613
0.4533333333333333 Epoch: 136 Cost: 4408.591434657574
0.45666666666666667 Epoch: 137 Cost: 4398.992884039879

0.46 Epoch: 138 Cost: 4400.908433139324
0.4633333333333333 Epoch: 139 Cost: 4396.412758111954
0.4666666666666667 Epoch: 140 Cost: 4396.462906122208
0.47 Epoch: 141 Cost: 4398.340850353241
0.47333333333333333 Epoch: 142 Cost: 4402.927444159985
0.4766666666666667 Epoch: 143 Cost: 4383.279222071171
0.48 Epoch: 144 Cost: 4380.556644201279
0.48333333333333334 Epoch: 145 Cost: 4392.264291405678
0.4866666666666667 Epoch: 146 Cost: 4380.921797573566
0.49 Epoch: 147 Cost: 4388.380382597446
0.49333333333333335 Epoch: 148 Cost: 4379.422616481781
0.49666666666666665 Epoch: 149 Cost: 4376.2942034602165
0.5 Epoch: 150 Cost: 4385.229397416115
0.5033333333333333 Epoch: 151 Cost: 4369.268422305584
0.5066666666666667 Epoch: 152 Cost: 4367.854983091354
0.51 Epoch: 153 Cost: 4374.9071007966995
0.5133333333333333 Epoch: 154 Cost: 4366.1664301157
0.5166666666666667 Epoch: 155 Cost: 4358.589497447014
0.52 Epoch: 156 Cost: 4362.663179636002
0.5233333333333333 Epoch: 157 Cost: 4367.745414674282
0.5266666666666666 Epoch: 158 Cost: 4357.311285972595
0.53 Epoch: 159 Cost: 4359.56849950552
0.5333333333333333 Epoch: 160 Cost: 4363.0372032523155
0.5366666666666666 Epoch: 161 Cost: 4353.119123876095
0.54 Epoch: 162 Cost: 4349.318535447121
0.5433333333333333 Epoch: 163 Cost: 4338.320441186428
0.5466666666666666 Epoch: 164 Cost: 4350.312439322472
0.55 Epoch: 165 Cost: 4351.465180397034
0.5533333333333333 Epoch: 166 Cost: 4359.37362742424
0.5566666666666666 Epoch: 167 Cost: 4348.535645127296
0.56 Epoch: 168 Cost: 4326.962784171104
0.5633333333333334 Epoch: 169 Cost: 4333.09072881937
0.5666666666666667 Epoch: 170 Cost: 4326.407300889492
0.57 Epoch: 171 Cost: 4325.000238001347
0.5733333333333334 Epoch: 172 Cost: 4338.798462986946
0.5766666666666667 Epoch: 173 Cost: 4326.737965166569
0.58 Epoch: 174 Cost: 4320.55959713459
0.5833333333333334 Epoch: 175 Cost: 4328.439543902874
0.5866666666666667 Epoch: 176 Cost: 4328.65830552578
0.59 Epoch: 177 Cost: 4324.766900777817
0.5933333333333334 Epoch: 178 Cost: 4305.130733549595
0.5966666666666667 Epoch: 179 Cost: 4315.684460401535
0.6 Epoch: 180 Cost: 4313.92194455862
0.6033333333333334 Epoch: 181 Cost: 4323.322410047054
0.6066666666666667 Epoch: 182 Cost: 4321.081378042698
0.61 Epoch: 183 Cost: 4308.1324117183685
0.6133333333333333 Epoch: 184 Cost: 4300.665404200554
0.6166666666666667 Epoch: 185 Cost: 4324.278484463692
0.62 Epoch: 186 Cost: 4325.11242890358
0.6233333333333333 Epoch: 187 Cost: 4299.570528626442
0.6266666666666667 Epoch: 188 Cost: 4314.236149728298
0.63 Epoch: 189 Cost: 4315.437850415707
0.6333333333333333 Epoch: 190 Cost: 4297.931919932365
0.6366666666666667 Epoch: 191 Cost: 4307.5025190114975
0.64 Epoch: 192 Cost: 4303.247478783131
0.6433333333333333 Epoch: 193 Cost: 4312.528435230255

0.6466666666666666 Epoch: 194 Cost: 4306.679012179375
0.65 Epoch: 195 Cost: 4289.258022367954
0.6533333333333333 Epoch: 196 Cost: 4289.288729965687
0.6566666666666666 Epoch: 197 Cost: 4294.316139340401
0.66 Epoch: 198 Cost: 4275.991395652294
0.6633333333333333 Epoch: 199 Cost: 4271.968594551086
0.6666666666666666 Epoch: 200 Cost: 4295.949177145958
0.67 Epoch: 201 Cost: 4283.072637677193
0.6733333333333333 Epoch: 202 Cost: 4284.425626695156
0.6766666666666666 Epoch: 203 Cost: 4280.667166233063
0.68 Epoch: 204 Cost: 4292.806351482868
0.6833333333333333 Epoch: 205 Cost: 4284.340915679932
0.6866666666666666 Epoch: 206 Cost: 4279.706401705742
0.69 Epoch: 207 Cost: 4291.858324408531
0.6933333333333334 Epoch: 208 Cost: 4266.517838537693
0.6966666666666667 Epoch: 209 Cost: 4281.866287827492
0.7 Epoch: 210 Cost: 4268.1794110536575
0.7033333333333334 Epoch: 211 Cost: 4260.45360738039
0.7066666666666667 Epoch: 212 Cost: 4278.446915626526
0.71 Epoch: 213 Cost: 4253.094880700111
0.7133333333333334 Epoch: 214 Cost: 4279.995619833469
0.7166666666666667 Epoch: 215 Cost: 4275.1316928863525
0.72 Epoch: 216 Cost: 4258.059379756451
0.7233333333333334 Epoch: 217 Cost: 4259.9076162576675
0.7266666666666667 Epoch: 218 Cost: 4268.476198554039
0.73 Epoch: 219 Cost: 4259.552305340767
0.7333333333333333 Epoch: 220 Cost: 4266.709404826164
0.7366666666666667 Epoch: 221 Cost: 4258.504854798317
0.74 Epoch: 222 Cost: 4247.20710259676
0.7433333333333333 Epoch: 223 Cost: 4261.492211937904
0.7466666666666667 Epoch: 224 Cost: 4255.860190927982
0.75 Epoch: 225 Cost: 4247.540048778057
0.7533333333333333 Epoch: 226 Cost: 4234.171633422375
0.7566666666666667 Epoch: 227 Cost: 4243.31098228693
0.76 Epoch: 228 Cost: 4238.262918293476
0.7633333333333333 Epoch: 229 Cost: 4241.333632409573
0.7666666666666667 Epoch: 230 Cost: 4266.541413068771
0.77 Epoch: 231 Cost: 4255.291230380535
0.7733333333333333 Epoch: 232 Cost: 4236.878985226154
0.7766666666666666 Epoch: 233 Cost: 4232.023225605488
0.78 Epoch: 234 Cost: 4247.176582336426
0.7833333333333333 Epoch: 235 Cost: 4262.300283432007
0.7866666666666666 Epoch: 236 Cost: 4236.578273594379
0.79 Epoch: 237 Cost: 4242.137474477291
0.7933333333333333 Epoch: 238 Cost: 4236.8689635396
0.7966666666666666 Epoch: 239 Cost: 4235.743195474148
0.8 Epoch: 240 Cost: 4246.170690536499
0.8033333333333333 Epoch: 241 Cost: 4225.9157836437225
0.8066666666666666 Epoch: 242 Cost: 4221.681380093098
0.81 Epoch: 243 Cost: 4217.655107438564
0.8133333333333334 Epoch: 244 Cost: 4235.941815435886
0.8166666666666667 Epoch: 245 Cost: 4228.704253137112
0.82 Epoch: 246 Cost: 4242.628641366959
0.8233333333333334 Epoch: 247 Cost: 4224.662956953049
0.8266666666666667 Epoch: 248 Cost: 4220.155010938644
0.83 Epoch: 249 Cost: 4221.051313996315

```
0.8333333333333334 Epoch: 250 Cost: 4230.9307206869125
0.8366666666666667 Epoch: 251 Cost: 4223.962675631046
0.84 Epoch: 252 Cost: 4225.9840705394745
0.8433333333333334 Epoch: 253 Cost: 4221.553915381432
0.8466666666666667 Epoch: 254 Cost: 4208.454810500145
0.85 Epoch: 255 Cost: 4218.052424967289
0.8533333333333334 Epoch: 256 Cost: 4224.584865272045
0.8566666666666667 Epoch: 257 Cost: 4229.240589559078
0.86 Epoch: 258 Cost: 4227.8152956962585
0.8633333333333333 Epoch: 259 Cost: 4197.608137011528
0.8666666666666667 Epoch: 260 Cost: 4208.539855718613
0.87 Epoch: 261 Cost: 4216.829761385918
0.8733333333333333 Epoch: 262 Cost: 4209.917200446129
0.8766666666666667 Epoch: 263 Cost: 4206.696895778179
0.88 Epoch: 264 Cost: 4208.27414560318
0.8833333333333333 Epoch: 265 Cost: 4199.312081754208
0.8866666666666667 Epoch: 266 Cost: 4213.488370239735
0.89 Epoch: 267 Cost: 4215.162463247776
0.8933333333333333 Epoch: 268 Cost: 4211.049794316292
0.8966666666666666 Epoch: 269 Cost: 4196.696676433086
0.9 Epoch: 270 Cost: 4204.797405600548
0.9033333333333333 Epoch: 271 Cost: 4202.436672329903
0.9066666666666666 Epoch: 272 Cost: 4216.190566956997
0.91 Epoch: 273 Cost: 4197.460306465626
0.9133333333333333 Epoch: 274 Cost: 4195.178662955761
0.9166666666666666 Epoch: 275 Cost: 4189.534616410732
0.92 Epoch: 276 Cost: 4187.0838151574135
0.9233333333333333 Epoch: 277 Cost: 4186.310322105885
0.9266666666666666 Epoch: 278 Cost: 4197.535088479519
0.93 Epoch: 279 Cost: 4201.6911262869835
0.9333333333333333 Epoch: 280 Cost: 4202.07620882988
0.9366666666666666 Epoch: 281 Cost: 4190.554895162582
0.94 Epoch: 282 Cost: 4186.589818716049
0.9433333333333334 Epoch: 283 Cost: 4188.494856953621
0.9466666666666667 Epoch: 284 Cost: 4180.952504038811
0.95 Epoch: 285 Cost: 4165.343299984932
0.9533333333333334 Epoch: 286 Cost: 4181.247908234596
0.9566666666666667 Epoch: 287 Cost: 4193.262175142765
0.96 Epoch: 288 Cost: 4186.209253668785
0.9633333333333334 Epoch: 289 Cost: 4170.092851996422
0.9666666666666667 Epoch: 290 Cost: 4185.920852184296
0.97 Epoch: 291 Cost: 4182.840587258339
0.9733333333333334 Epoch: 292 Cost: 4181.743186891079
0.9766666666666667 Epoch: 293 Cost: 4178.177795052528
0.98 Epoch: 294 Cost: 4186.912286937237
0.9833333333333333 Epoch: 295 Cost: 4163.575363576412
0.9866666666666667 Epoch: 296 Cost: 4181.658987343311
0.99 Epoch: 297 Cost: 4177.236304223537
0.9933333333333333 Epoch: 298 Cost: 4183.549402654171
0.9966666666666667 Epoch: 299 Cost: 4188.634226322174
```

Calculating training accuracy:

In [10]:
```
filename = "transfer"
f = open(filename + '.txt', "a")
```

```
## Initialize objects for counting correct/total
correct = 0
total = 0

# Specify no changes to the gradient in the subsequent steps (since we're not using
with torch.no_grad():
    for data in train_loader:
        # Current batch of data
        images, labels = data

        # pass each batch into the network
        outputs = net(images)

        # the class with the maximum score is what we choose as prediction
        _, predicted = torch.max(outputs.data, 1)

        # add size of the current batch
        total += labels.size(0)

        # add the number of correct predictions in the current batch
        correct += (predicted == labels).sum().item()

## Calculate and print the proportion correct
print(f"Training Accuracy is {correct/total}")
f.write(f"Training Accuracy is {correct/total}")
```

```
Training Accuracy is 0.3726790009865978
```

Calculating testing accuracy:

In [11]:

```
## Combine X and y tensors into a TensorDataset and DataLoader
test_loader = DataLoader(TensorDataset(test_X.type(torch.FloatTensor),
                         torch.Tensor(test_y).type(torch.LongTensor)), batch_size=bs

## Initialize objects for counting correct/total
correct = 0
total = 0

# Specify no changes to the gradient in the subsequent steps (since we're not using
with torch.no_grad():
    for data in test_loader:
        # Current batch of data
        images, labels = data

        # pass each batch into the network
        outputs = net(images)

        # the class with the maximum score is what we choose as prediction
        _, predicted = torch.max(outputs.data, 1)

        # add size of the current batch
        total += labels.size(0)

        # add the number of correct predictions in the current batch
        correct += (predicted == labels).sum().item()
```

```
## Calculate and print the proportion correct
print(f"Test Accuracy is {correct/total}")
f.write(f"\nTest Accuracy is {correct/total}")

f.close()
```

```
Test Accuracy is 0.36907216494845363
```

In [12]:

```
# verifying the convergence of cost
import matplotlib.pyplot as plt
plt.plot(np.linspace(0, epochs, epochs), track_cost)
plt.show()


plt.plot(np.linspace(0, epochs, epochs), track_cost)
plt.savefig('transfer_plot.png')
```