# load packages

import numpy as np

import matplotlib.pyplot as plt

from datetime import datetime

from pathlib import Path

from tqdm import tqdm

from sklearn.metrics import classification\_report, confusion\_matrix, ConfusionMatrixDisplay

import torch

from torch.utils import data

import torch.nn as nn

device = torch.device("cuda" if torch.cuda.is\_available() else "cpu")

print("Using device:", device)

The dataset used is the FI-2010 dataset.

As in the original paper I used the first 7 days to train and to validate, and the rest 3 days to do the the testing.

# please change the data\_path to your local path and unzip the file

dec\_data = np.loadtxt('BenchmarkDatasets/NoAuction/1.NoAuction\_Zscore/NoAuction\_Zscore\_Training/Train\_Dst\_NoAuction\_ZScore\_CF\_7.txt')

dec\_train = dec\_data[:, :int(dec\_data.shape[1] \* 0.8)]

dec\_val = dec\_data[:, int(dec\_data.shape[1] \* 0.8):]

dec\_test1 = np.loadtxt('BenchmarkDatasets/NoAuction/1.NoAuction\_Zscore/NoAuction\_Zscore\_Testing/Test\_Dst\_NoAuction\_ZScore\_CF\_7.txt')

dec\_test2 = np.loadtxt('BenchmarkDatasets/NoAuction/1.NoAuction\_Zscore/NoAuction\_Zscore\_Testing/Test\_Dst\_NoAuction\_ZScore\_CF\_8.txt')

dec\_test3 = np.loadtxt('BenchmarkDatasets/NoAuction/1.NoAuction\_Zscore/NoAuction\_Zscore\_Testing/Test\_Dst\_NoAuction\_ZScore\_CF\_9.txt')

dec\_test = np.hstack((dec\_test1, dec\_test2, dec\_test3))

W = 40 #number of features

dim = 40 #number of LOB states

horizon = 2 #if h = 5 than k = 10, h = 2 then k=50

T = 5

y\_train = dec\_train[-horizon, :].flatten()

y\_val = dec\_val[-horizon, :].flatten()

y\_test = dec\_test[-horizon, :].flatten()

y\_train = y\_train[dim-1:] - 1

y\_val = y\_val[dim-1:] - 1

y\_test = y\_test[dim-1:] - 1

dec\_train = dec\_train[:40, :].T

dec\_val = dec\_val[:40, :].T

dec\_test = dec\_test[:40, :].T

print("Training data shape:", dec\_train.shape)

print("Validation data shape:", dec\_val.shape)

print("Testing data shape:", dec\_test.shape)  
  
raining data shape: (203800, 40)

Validation data shape: (50950, 40)

Testing data shape: (139587, 40)

class Dataset(data.Dataset):

"""Characterizes a dataset for PyTorch"""

def \_\_init\_\_(self, x, y, num\_classes, dim):

"""Initialization"""

self.num\_classes = num\_classes

self.dim = dim

self.x = x

self.y = y

self.length = x.shape[0] - T - self.dim + 1

x = torch.from\_numpy(x)

self.x = torch.unsqueeze(x, 1)

self.y = torch.from\_numpy(y)

def \_\_len\_\_(self):

"""Denotes the total number of samples"""

return self.length

def \_\_getitem\_\_(self, i):

input = self.x[i:i+self.dim, :]

input = input.permute(1, 0, 2)

return input, self.y[i]  
  
#Hyperparameters

batch\_size = 64

epochs = 20

c\_final = 4 #channel output size of the second conv

n\_heads = 4

c\_in\_axial = 32 #channel output size of the first conv

c\_out\_axial = 32

pool\_kernel = (1, 4)

pool\_stride = (1, 4)

num\_classes = 3

dataset\_val = Dataset(dec\_val, y\_val, num\_classes, dim)

dataset\_test = Dataset(dec\_test, y\_test, num\_classes, dim)

dataset\_train = Dataset(dec\_train, y\_train, num\_classes, dim)

train\_loader = torch.utils.data.DataLoader(dataset=dataset\_train, batch\_size=batch\_size, shuffle=True)

val\_loader = torch.utils.data.DataLoader(dataset=dataset\_val, batch\_size=batch\_size, shuffle=False)

test\_loader = torch.utils.data.DataLoader(dataset=dataset\_test, batch\_size=batch\_size, shuffle=False)

inputs, labels = next(iter(train\_loader))

print("Input batch shape:", inputs.shape)

print("Label batch shape:", labels.shape)  
  
Input batch shape: torch.Size([64, 1, 40, 40])

Label batch shape: torch.Size([64])

**Model Architecture**[**¶**](http://localhost:8892/notebooks/DL_Project/AxialLOB.ipynb#Model-Architecture)

import math

def \_conv1d1x1(in\_channels, out\_channels):

return nn.Sequential(nn.Conv1d(in\_channels, out\_channels, kernel\_size=1, stride=1, bias=False),

nn.BatchNorm1d(out\_channels))

#class taken from https://github.com/jeya-maria-jose/Medical-Transformer/blob/main/lib/models/axialnet.py

class GatedAxialAttention(nn.Module):

def \_\_init\_\_(self, in\_channels, out\_channels, heads, dim, flag):

assert (in\_channels % heads == 0) and (out\_channels % heads == 0)

super().\_\_init\_\_()

self.in\_channels = in\_channels

self.out\_channels = out\_channels

self.heads = heads

self.dim\_head\_v = out\_channels // heads

self.flag = flag #if flag then we do the attention along width

self.dim = dim

self.dim\_head\_qk = self.dim\_head\_v // 2

self.qkv\_channels = self.dim\_head\_v + self.dim\_head\_qk \* 2

# Multi-head self attention

self.to\_qkv = \_conv1d1x1(in\_channels, self.heads \* self.qkv\_channels)

self.bn\_qkv = nn.BatchNorm1d(self.heads \* self.qkv\_channels)

self.bn\_similarity = nn.BatchNorm2d(heads \* 3)

self.bn\_output = nn.BatchNorm1d(self.heads \* self.qkv\_channels)

# Gating mechanism

self.f\_qr = nn.Parameter(torch.tensor(0.3), requires\_grad=False)

self.f\_kr = nn.Parameter(torch.tensor(0.3), requires\_grad=False)

self.f\_sve = nn.Parameter(torch.tensor(0.3), requires\_grad=False)

self.f\_sv = nn.Parameter(torch.tensor(0.5), requires\_grad=False)

# Position embedding

self.relative = nn.Parameter(torch.randn(self.dim\_head\_v \* 2, dim \* 2 - 1), requires\_grad=True)

query\_index = torch.arange(dim).unsqueeze(0)

key\_index = torch.arange(dim).unsqueeze(1)

relative\_index = key\_index - query\_index + dim - 1

self.register\_buffer('flatten\_index', relative\_index.view(-1))

self.reset\_parameters()

# self.print\_para()

def forward(self, x):

if self.flag:

x = x.permute(0, 2, 1, 3)

else:

x = x.permute(0, 3, 1, 2) # N, W, C, H

N, W, C, H = x.shape

x = x.contiguous().view(N \* W, C, H)

# Transformations

x = self.to\_qkv(x)

qkv = self.bn\_qkv(x)

q, k, v = torch.split(qkv.reshape(N \* W, self.heads, self.dim\_head\_v \* 2, H), [self.dim\_head\_v // 2, self.dim\_head\_v // 2, self.dim\_head\_v], dim=2)

# Calculate position embedding

all\_embeddings = torch.index\_select(self.relative, 1, self.flatten\_index).view(self.dim\_head\_v \* 2, self.dim, self.dim)

q\_embedding, k\_embedding, v\_embedding = torch.split(all\_embeddings, [self.dim\_head\_qk, self.dim\_head\_qk, self.dim\_head\_v], dim=0)

qr = torch.einsum('bgci,cij->bgij', q, q\_embedding)

kr = torch.einsum('bgci,cij->bgij', k, k\_embedding).transpose(2, 3)

qk = torch.einsum('bgci, bgcj->bgij', q, k)

# multiply by factors

qr = torch.mul(qr, self.f\_qr)

kr = torch.mul(kr, self.f\_kr)

stacked\_similarity = torch.cat([qk, qr, kr], dim=1)

stacked\_similarity = self.bn\_similarity(stacked\_similarity).view(N \* W, 3, self.heads, H, H).sum(dim=1)

#stacked\_similarity = self.bn\_qr(qr) + self.bn\_kr(kr) + self.bn\_qk(qk)

# (N, heads, H, H, W)

similarity = torch.softmax(stacked\_similarity, dim=3)

sv = torch.einsum('bgij,bgcj->bgci', similarity, v)

sve = torch.einsum('bgij,cij->bgci', similarity, v\_embedding)

# multiply by factors

sv = torch.mul(sv, self.f\_sv)

sve = torch.mul(sve, self.f\_sve)

stacked\_output = torch.cat([sv, sve], dim=-1).view(N \* W, self.out\_channels \* 2, H)

output = self.bn\_output(stacked\_output).view(N, W, self.out\_channels, 2, H).sum(dim=-2)

if self.flag:

output = output.permute(0, 2, 1, 3)

else:

output = output.permute(0, 2, 3, 1)

return output

def reset\_parameters(self):

nn.init.normal\_(self.relative, 0., math.sqrt(1. / self.dim\_head\_v))

class AxialLOB(nn.Module):

def \_\_init\_\_(self, W, H, c\_in, c\_out, c\_final, n\_heads, pool\_kernel, pool\_stride):

super().\_\_init\_\_()

"""

Args:

W and H: the width and height of the input tensors

c\_in, c\_out, and c\_final: the number of channels for the input, intermediate, and final convolutional layers

n\_heads: the number of heads for the multi-head attention mechanism used in the GatedAxialAttention layers.

pool\_kernel and pool\_stride: the kernel size and stride of the final average pooling layer.

"""

#channel output of the CNN\_in is the channel input for the axial layer

self.c\_in = c\_in

self.c\_out = c\_out

self.c\_final = c\_final

self.CNN\_in = nn.Conv2d(in\_channels=1, out\_channels=c\_in, kernel\_size=1)

self.CNN\_out = nn.Conv2d(in\_channels=c\_out, out\_channels=c\_final, kernel\_size=1)

self.CNN\_res2 = nn.Conv2d(in\_channels=c\_out, out\_channels=c\_final, kernel\_size=1)

self.CNN\_res1 = nn.Conv2d(in\_channels=1, out\_channels=c\_out, kernel\_size=1)

self.norm = nn.BatchNorm2d(c\_in)

self.res\_norm2 = nn.BatchNorm2d(c\_final)

self.res\_norm1 = nn.BatchNorm2d(c\_out)

self.norm2 = nn.BatchNorm2d(c\_final)

self.axial\_height\_1 = GatedAxialAttention(c\_out, c\_out, n\_heads, H, flag=False)

self.axial\_width\_1 = GatedAxialAttention(c\_out, c\_out, n\_heads, W, flag=True)

self.axial\_height\_2 = GatedAxialAttention(c\_out, c\_out, n\_heads, H, flag=False)

self.axial\_width\_2 = GatedAxialAttention(c\_out, c\_out, n\_heads, W, flag=True)

self.activation = nn.ReLU()

self.linear = nn.Linear(1600, 3)

self.pooling = nn.AvgPool2d(kernel\_size=pool\_kernel, stride=pool\_stride)

def forward(self, x):

#up branch

#first convolution before the attention

y = self.CNN\_in(x)

y = self.norm(y)

y = self.activation(y)

#attention mechanism through gated multi head axial layer

y = self.axial\_width\_1(y)

y = self.axial\_height\_1(y)

#lower branch

x = self.CNN\_res1(x)

x = self.res\_norm1(x)

x = self.activation(x)

#first residual

y = y + x

z = y.detach().clone()

#second axial layer

y = self.axial\_width\_2(y)

y = self.axial\_height\_2(y)

#second convolution

y = self.CNN\_out(y)

y = self.res\_norm2(y)

y = self.activation(y)

#lower branch

z = self.CNN\_res2(z)

z = self.norm2(z)

z = self.activation(z)

#second res connection

y = y + z

#final part

y = self.pooling(y)

y = torch.flatten(y, 1)

y = self.linear(y)

forecast\_y = torch.softmax(y, dim=1)

return forecast\_y  
**Model Training**[**¶**](http://localhost:8892/notebooks/DL_Project/AxialLOB.ipynb#Model-Training)

axial\_lob\_model = AxialLOB(W, dim, c\_in\_axial, c\_out\_axial, c\_final, n\_heads, pool\_kernel, pool\_stride)

axial\_lob\_model.to(device)

criterion = nn.CrossEntropyLoss()

optimizer = torch.optim.SGD(axial\_lob\_model.parameters(), lr=0.01, momentum=0.9)

scheduler = torch.optim.lr\_scheduler.CosineAnnealingLR(optimizer, T\_max=epochs, eta\_min=0.00001)

def batch\_gd(model, criterion, optimizer, scheduler, epochs, model\_name, train\_loader, val\_loader, force\_save=False):

if not force\_save:

save\_path = Path(model\_name)

if save\_path.exists():

raise ValueError(

f"{model\_name} exists already. Please use a different path/name "

+ "or set force\_save=True to override the current model"

)

train\_losses = np.zeros(epochs)

test\_losses = np.zeros(epochs)

best\_test\_loss = np.inf

best\_test\_epoch = 0

start\_time = datetime.now()

for it in tqdm(range(epochs)):

if (it == 4 and isinstance(model, AxialLOB)):

print("Turning on additional grads for AxialLOB...")

model.axial\_height\_1.f\_qr.requires\_grad = True

model.axial\_height\_1.f\_kr.requires\_grad = True

model.axial\_height\_1.f\_sve.requires\_grad = True

model.axial\_height\_1.f\_sv.requires\_grad = True

model.axial\_width\_1.f\_qr.requires\_grad = True

model.axial\_width\_1.f\_kr.requires\_grad = True

model.axial\_width\_1.f\_sve.requires\_grad = True

model.axial\_width\_1.f\_sv.requires\_grad = True

model.axial\_height\_2.f\_qr.requires\_grad = True

model.axial\_height\_2.f\_kr.requires\_grad = True

model.axial\_height\_2.f\_sve.requires\_grad = True

model.axial\_height\_2.f\_sv.requires\_grad = True

model.axial\_width\_2.f\_qr.requires\_grad = True

model.axial\_width\_2.f\_kr.requires\_grad = True

model.axial\_width\_2.f\_sve.requires\_grad = True

model.axial\_width\_2.f\_sv.requires\_grad = True

model.train()

t0 = datetime.now()

train\_loss = []

for inputs, targets in train\_loader:

# move data to GPU

inputs, targets = inputs.to(device, dtype=torch.float), targets.to(device, dtype=torch.int64)

# zero the parameter gradients

optimizer.zero\_grad()

# Forward pass

outputs = model(inputs)

loss = criterion(outputs, targets)

# Backward and optimize

loss.backward()

optimizer.step()

train\_loss.append(loss.item())

# Get train loss and test loss

train\_loss = np.mean(train\_loss)

model.eval()

test\_loss = []

for inputs, targets in val\_loader:

inputs, targets = inputs.to(device, dtype=torch.float), targets.to(device, dtype=torch.int64)

outputs = model(inputs)

loss = criterion(outputs, targets)

test\_loss.append(loss.item())

test\_loss = np.mean(test\_loss)

# Save losses

train\_losses[it] = train\_loss

test\_losses[it] = test\_loss

scheduler.step()

#We save the best model

if test\_loss < best\_test\_loss:

torch.save(model.state\_dict(), model\_name)

best\_test\_loss = test\_loss

best\_test\_epoch = it

print(f'model saved at {model\_name}')

dt = datetime.now() - t0

print(f'Epoch {it+1}/{epochs}, Train Loss: {train\_loss:.4f}, \

Validation Loss: {test\_loss:.4f}, Duration: {dt}, Best Val Epoch: {best\_test\_epoch}')

time\_training = datetime.now() - start\_time

print(f"Full time to train: {time\_training}. Average time per epoch: {time\_training / epochs}")

return train\_losses, test\_losses  
  
# If the model is already in all\_models, you can skip this block

# unless you want to train the AxialLOB model further

print("------- List Hyper Parameters -------")

print("epochs -> " + str(epochs))

print("horizon -> " + str(T\*10))

print("batch size -> " + str(batch\_size))

print("Optimizer -> " + str(optimizer))

# Save to the all\_models directory

# If it doesn't exist, make it

model\_name = "all\_models/AxialLOB.pt"

model\_path = Path(model\_name)

if TRAIN\_MODELS:

save\_dir = Path("all\_models")

save\_dir.mkdir(parents=True, exist\_ok=True)

# If you want to start completely from scratch,

# comment out the following code and set force\_save = True above

force\_save = False

model\_path = Path(model\_name)

if model\_path.exists():

force\_save = True

print("Loading existing axial\_lob\_model...")

axial\_lob\_model.load\_state\_dict(torch.load(model\_name, map\_location=device))

axial\_lob\_model.to(device)

print("Model loaded. Training on existing model")

train\_losses, val\_losses = batch\_gd(axial\_lob\_model, criterion, optimizer, scheduler, epochs, model\_name, train\_loader, val\_loader, force\_save=force\_save)

plt.figure(figsize=(15,6))

plt.plot(train\_losses, label='train loss')

plt.plot(val\_losses, label='validation loss')

plt.legend()

else:

if not model\_path.exists():

print(f"WARNING: TRAIN\_MODELS={TRAIN\_MODELS} but \"{model\_path}\" does not exist")  
  
------- List Hyper Parameters -------

epochs -> 20

horizon -> 50

batch size -> 64

Optimizer -> SGD (

Parameter Group 0

dampening: 0

differentiable: False

foreach: None

fused: None

initial\_lr: 0.01

lr: 0.01

maximize: False

momentum: 0.9

nesterov: False

weight\_decay: 0

)

**Model Testing**[**¶**](http://localhost:8892/notebooks/DL_Project/AxialLOB.ipynb#Model-Testing)

def test\_model(model, criterion, test\_loader):

n\_correct = 0.

n\_total = 0.

all\_targets = []

all\_predictions = []

test\_loss = []

t0 = datetime.now()

for inputs, targets in test\_loader:

# Move to GPU

inputs, targets = inputs.to(device, dtype=torch.float), targets.to(device, dtype=torch.int64)

# Forward pass

outputs = model(inputs)

loss = criterion(outputs, targets)

# Get prediction

# torch.max returns both max and argmax

\_, predictions = torch.max(outputs, 1)

# update counts

n\_correct += (predictions == targets).sum().item()

n\_total += targets.shape[0]

all\_targets.append(targets.cpu().numpy())

all\_predictions.append(predictions.cpu().numpy())

test\_loss.append(loss.item())

dt = datetime.now() - t0

test\_loss = np.mean(test\_loss)

test\_acc = n\_correct / n\_total

print(f"Test acc: {test\_acc:.4f}. Test Loss: {test\_loss:.4f}. Time on test: {dt}")

all\_targets = np.concatenate(all\_targets)

all\_predictions = np.concatenate(all\_predictions)

return all\_targets, all\_predictions  
  
axial\_lob\_model.load\_state\_dict(torch.load(model\_name, map\_location=device))

axial\_lob\_model.to(device)

all\_targets, all\_predictions = test\_model(axial\_lob\_model, criterion, test\_loader)

print(classification\_report(all\_targets, all\_predictions, digits=4))

c = confusion\_matrix(all\_targets, all\_predictions, normalize="true")

disp = ConfusionMatrixDisplay(c)

disp.plot()

plt.show()  
Test acc: 0.7125. Test Loss: 0.8288. Time on test: 0:01:09.131521

precision recall f1-score support

0 0.6893 0.6414 0.6645 38447

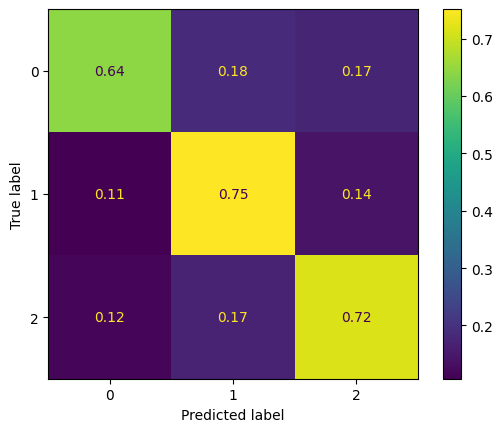
1 0.7927 0.7516 0.7716 65996

2 0.6110 0.7171 0.6598 35100

accuracy 0.7125 139543

macro avg 0.6977 0.7034 0.6986 139543

weighted avg 0.7185 0.7125 0.7140 139543



**Model Comparisons**[**¶**](http://localhost:8892/notebooks/DL_Project/AxialLOB.ipynb#Model-Comparisons)

class CNN(nn.Module):

def \_\_init\_\_(self, W, num\_classes):

super().\_\_init\_\_()

self.conv = nn.Conv2d(1, 16, kernel\_size=3, stride=1, padding=1)

self.pool = nn.AdaptiveAvgPool2d(1)

self.fc = nn.Linear(16, num\_classes)

def forward(self, x):

x = torch.relu(self.conv(x))

x = self.pool(x)

x = x.view(x.size(0), -1)

return self.fc(x)

models\_to\_test["CNN"] = CNN(W, num\_classes).to(device)  
class DeepLOB(nn.Module):

def \_\_init\_\_(self, W, hidden\_size, num\_classes):

super().\_\_init\_\_()

self.conv1 = nn.Conv2d(1, 32, kernel\_size=3, padding=1)

self.bn1 = nn.BatchNorm2d(32)

self.conv2 = nn.Conv2d(32, 64, kernel\_size=3, padding=1)

self.bn2 = nn.BatchNorm2d(64)

self.conv3 = nn.Conv2d(64, 128, kernel\_size=3, padding=1)

self.bn3 = nn.BatchNorm2d(128)

self.lstm = nn.LSTM(W \* 128, hidden\_size, batch\_first=True, bidirectional=True)

self.fc = nn.Sequential(

nn.Linear(2 \* hidden\_size, 128),

nn.ReLU(),

nn.Dropout(0.3),

nn.Linear(128, num\_classes)

)

def forward(self, x):

x = torch.relu(self.bn1(self.conv1(x)))

x = torch.relu(self.bn2(self.conv2(x)))

x = torch.relu(self.bn3(self.conv3(x)))

batch\_size, channels, height, width = x.size()

x = x.view(batch\_size, height, -1)

x, \_ = self.lstm(x)

x = x[:, -1, :]

return self.fc(x)

hidden\_size = 64

models\_to\_test["DeepLOB"] = DeepLOB(W, hidden\_size, num\_classes).to(device)  
  
class B\_TABL(nn.Module):

def \_\_init\_\_(self, W, num\_classes):

super().\_\_init\_\_()

self.fc1 = nn.Linear(W \* W, 128)

self.fc2 = nn.Linear(128, num\_classes)

def forward(self, x):

x = x.view(x.size(0), -1) # Flatten input

x = torch.relu(self.fc1(x))

return self.fc2(x)

class C\_TABL(nn.Module):

def \_\_init\_\_(self, W, num\_classes):

super().\_\_init\_\_()

self.fc1 = nn.Linear(W \* W, 256)

self.fc2 = nn.Linear(256, 128)

self.fc3 = nn.Linear(128, num\_classes)

def forward(self, x):

x = x.view(x.size(0), -1) # Flatten input

x = torch.relu(self.fc1(x))

x = torch.relu(self.fc2(x))

return self.fc3(x)

models\_to\_test["B\_TABL"] = B\_TABL(W, num\_classes).to(device)

models\_to\_test["C\_TABL"] = C\_TABL(W, num\_classes).to(device)  
  
class DeepLOBSeq2Seq(nn.Module):

def \_\_init\_\_(self, W, hidden\_size, num\_classes):

super().\_\_init\_\_()

self.hidden\_size = hidden\_size

self.encoder = nn.LSTM(W, hidden\_size, batch\_first=True, bidirectional=True)

self.decoder = nn.LSTM(2 \* hidden\_size, hidden\_size, batch\_first=True, bidirectional=True)

self.fc = nn.Sequential(

nn.Linear(2 \* hidden\_size, 128),

nn.ReLU(),

nn.Dropout(0.3),

nn.Linear(128, num\_classes)

)

def forward(self, x):

# Reshape input

batch\_size, \_, height, \_ = x.size()

x = x.view(batch\_size, height, -1)

# Encoder

encoder\_output, (hidden, cell) = self.encoder(x)

hidden = torch.cat((hidden[0:1], hidden[1:2]), dim=2)

hidden = hidden.view(2, batch\_size, self.hidden\_size)

cell = torch.cat((cell[0:1], cell[1:2]), dim=2)

cell = cell.view(2, batch\_size, self.hidden\_size)

# Decoder

decoder\_output, \_ = self.decoder(encoder\_output, (hidden, cell))

# Use the last time step from the decoder

x = decoder\_output[:, -1, :]

# Fully connected layer

return self.fc(x)

hidden\_size = 64

models\_to\_test["DeepLOBSeq2Seq"] = DeepLOBSeq2Seq(W, hidden\_size, num\_classes).to(device)  
  
class DeepLOBAttention(nn.Module):

def \_\_init\_\_(self, W, hidden\_size, num\_classes):

super().\_\_init\_\_()

self.encoder = nn.LSTM(W, hidden\_size, batch\_first=True, bidirectional=True)

self.attention = nn.Sequential(

nn.Linear(2 \* hidden\_size, 128),

nn.Tanh(),

nn.Linear(128, 1),

nn.Softmax(dim=1)

)

self.fc = nn.Sequential(

nn.Linear(2 \* hidden\_size, 128),

nn.ReLU(),

nn.Dropout(0.3),

nn.Linear(128, num\_classes)

)

def forward(self, x):

batch\_size, \_, height, \_ = x.size()

x = x.view(batch\_size, height, -1) # Flatten spatial dimensions

x, \_ = self.encoder(x) # Encoder outputs

attention\_weights = self.attention(x) # Compute attention weights

x = torch.sum(attention\_weights \* x, dim=1) # Apply attention

return self.fc(x)

hidden\_size = 64

models\_to\_test["DeepLOBAttention"] = DeepLOBAttention(W, hidden\_size, num\_classes).to(device)  
  
# Load all models

def load\_from\_saved(models\_to\_test: dict[str, nn.Module], pretrained\_loc: str | None = None):

# Load the models that were pretrained on the FI-2010 dataset

all\_loaded\_successfully = True

for model\_name in models\_to\_test.keys():

\_save\_path = Path(f"{pretrained\_loc}/{model\_name}.pt")

if \_save\_path.exists():

models\_to\_test[model\_name].load\_state\_dict(torch.load(\_save\_path, map\_location=device))

models\_to\_test[model\_name].to(device)

else:

print("WARNING: Tried to load from pretrained but did not find any saved models")

all\_loaded\_successfully = False

if all\_loaded\_successfully:

print("All models loaded successfully")

else:

print("\nWARNING: At least one model was not loaded successfully. Please read the warning(s) above.\n")

load\_from\_saved(models\_to\_test, "all\_models")  
  
class FakeScheduler():

"Class so we can call batch\_gd with no scheduler"

def step(self):

pass

def evaluate\_all(models: dict[str, nn.Module],

epochs\_per\_model: int,

train\_loader: data.DataLoader,

val\_loader: data.DataLoader,

test\_loader: data.DataLoader,

train\_model: bool = False,

save\_dir: str = "all\_models"

):

"""

Trains each model (in place) in models on the dataset.

"""

save\_dir\_path = Path(save\_dir)

save\_dir\_path.mkdir(parents=True, exist\_ok=True)

for model\_name, model in models.items():

criterion = nn.CrossEntropyLoss()

if train\_model:

# Train the model and save it

print(f"Training {model\_name}...")

model.to(device)

optimizer = torch.optim.Adam(model.parameters(), lr=0.001)

scheduler = FakeScheduler()

save\_path = save\_dir\_path / f"{model\_name}.pt"

batch\_gd(model, criterion, optimizer, scheduler, epochs\_per\_model, save\_path, train\_loader, val\_loader, force\_save=True)

# Load the new best model

model.load\_state\_dict(torch.load(save\_path, map\_location=device))

model.to(device)

# Evaluate the model

# Load the best saved model

print(f"Evaluating {model\_name} on test set\n")

all\_targets, all\_predictions = test\_model(model, criterion, test\_loader)

print(classification\_report(all\_targets, all\_predictions, digits=4))

c = confusion\_matrix(all\_targets, all\_predictions, normalize="true")

disp = ConfusionMatrixDisplay(c)

disp.plot()

plt.show()

# Replace old model with new model

models[model\_name] = model

print("\n\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n\n")

evaluate\_all(models\_to\_test, epochs, train\_loader, val\_loader, test\_loader, train\_model=TRAIN\_MODELS)  
  
Evaluating CNN on test set

Test acc: 0.5018. Test Loss: 1.0047. Time on test: 0:00:03.224519

precision recall f1-score support

0 0.3190 0.0121 0.0234 38447

1 0.5690 0.8259 0.6738 65996

2 0.3559 0.4289 0.3890 35100

accuracy 0.5018 139543

macro avg 0.4146 0.4223 0.3621 139543

weighted avg 0.4465 0.5018 0.4230 139543

A chart with different colored squares

Description automatically generated

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Evaluating DeepLOB on test set

Test acc: 0.7156. Test Loss: 0.7222. Time on test: 0:00:08.455506

precision recall f1-score support

0 0.7398 0.5804 0.6504 38447

1 0.7187 0.8478 0.7779 65996

2 0.6848 0.6151 0.6481 35100

accuracy 0.7156 139543

macro avg 0.7144 0.6811 0.6922 139543

weighted avg 0.7160 0.7156 0.7101 139543

A chart of different colored squares

Description automatically generated

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Evaluating B\_TABL on test set

Test acc: 0.5995. Test Loss: 0.9399. Time on test: 0:00:02.781998

precision recall f1-score support

0 0.5544 0.5066 0.5294 38447

1 0.6507 0.6907 0.6701 65996

2 0.5413 0.5298 0.5355 35100

accuracy 0.5995 139543

macro avg 0.5821 0.5757 0.5784 139543

weighted avg 0.5967 0.5995 0.5975 139543

A chart of different colored squares

Description automatically generated

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Evaluating C\_TABL on test set

Test acc: 0.5166. Test Loss: 1.0276. Time on test: 0:00:03.101000

precision recall f1-score support

0 0.4796 0.3675 0.4162 38447

1 0.5675 0.6304 0.5973 65996

2 0.4450 0.4660 0.4553 35100

accuracy 0.5166 139543

macro avg 0.4973 0.4880 0.4896 139543

weighted avg 0.5124 0.5166 0.5117 139543

A chart of different colored squares

Description automatically generated

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Evaluating DeepLOBSeq2Seq on test set

Test acc: 0.7163. Test Loss: 0.7140. Time on test: 0:00:04.347000

precision recall f1-score support

0 0.6504 0.6839 0.6667 38447

1 0.8016 0.7645 0.7826 65996

2 0.6415 0.6610 0.6511 35100

accuracy 0.7163 139543

macro avg 0.6978 0.7031 0.7001 139543

weighted avg 0.7196 0.7163 0.7176 139543

A chart with different colored squares

Description automatically generated

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Evaluating DeepLOBAttention on test set

Test acc: 0.6976. Test Loss: 0.7533. Time on test: 0:00:04.148000

precision recall f1-score support

0 0.6564 0.6083 0.6314 38447

1 0.7739 0.7739 0.7739 65996

2 0.6036 0.6520 0.6269 35100

accuracy 0.6976 139543

macro avg 0.6780 0.6781 0.6774 139543

weighted avg 0.6987 0.6976 0.6977 139543

A chart of a number of colored squares

Description automatically generated with medium confidence

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Evaluating CNN on test set

Test acc: 0.5018. Test Loss: 1.0047. Time on test: 0:00:03.224519

precision recall f1-score support

0 0.3190 0.0121 0.0234 38447

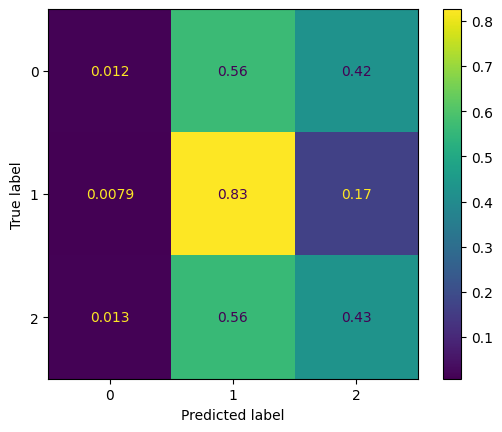
1 0.5690 0.8259 0.6738 65996

2 0.3559 0.4289 0.3890 35100

accuracy 0.5018 139543

macro avg 0.4146 0.4223 0.3621 139543

weighted avg 0.4465 0.5018 0.4230 139543



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Evaluating DeepLOB on test set

Test acc: 0.7156. Test Loss: 0.7222. Time on test: 0:00:08.455506

precision recall f1-score support

0 0.7398 0.5804 0.6504 38447

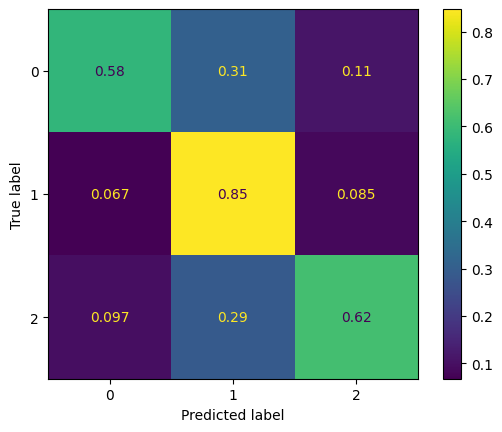
1 0.7187 0.8478 0.7779 65996

2 0.6848 0.6151 0.6481 35100

accuracy 0.7156 139543

macro avg 0.7144 0.6811 0.6922 139543

weighted avg 0.7160 0.7156 0.7101 139543



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Evaluating B\_TABL on test set

Test acc: 0.5995. Test Loss: 0.9399. Time on test: 0:00:02.781998

precision recall f1-score support

0 0.5544 0.5066 0.5294 38447

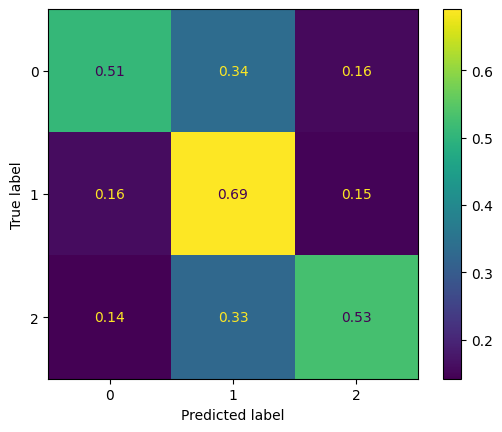
1 0.6507 0.6907 0.6701 65996

2 0.5413 0.5298 0.5355 35100

accuracy 0.5995 139543

macro avg 0.5821 0.5757 0.5784 139543

weighted avg 0.5967 0.5995 0.5975 139543



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Evaluating C\_TABL on test set

Test acc: 0.5166. Test Loss: 1.0276. Time on test: 0:00:03.101000

precision recall f1-score support

0 0.4796 0.3675 0.4162 38447

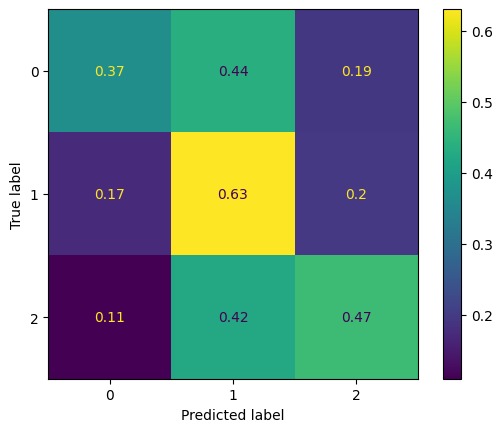
1 0.5675 0.6304 0.5973 65996

2 0.4450 0.4660 0.4553 35100

accuracy 0.5166 139543

macro avg 0.4973 0.4880 0.4896 139543

weighted avg 0.5124 0.5166 0.5117 139543



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Evaluating DeepLOBSeq2Seq on test set

Test acc: 0.7163. Test Loss: 0.7140. Time on test: 0:00:04.347000

precision recall f1-score support

0 0.6504 0.6839 0.6667 38447

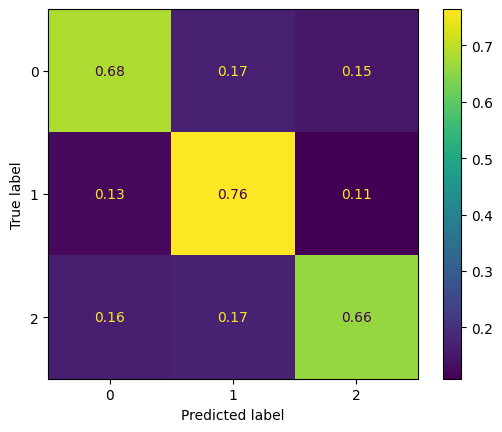
1 0.8016 0.7645 0.7826 65996

2 0.6415 0.6610 0.6511 35100

accuracy 0.7163 139543

macro avg 0.6978 0.7031 0.7001 139543

weighted avg 0.7196 0.7163 0.7176 139543



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Evaluating DeepLOBAttention on test set

Test acc: 0.6976. Test Loss: 0.7533. Time on test: 0:00:04.148000

precision recall f1-score support

0 0.6564 0.6083 0.6314 38447

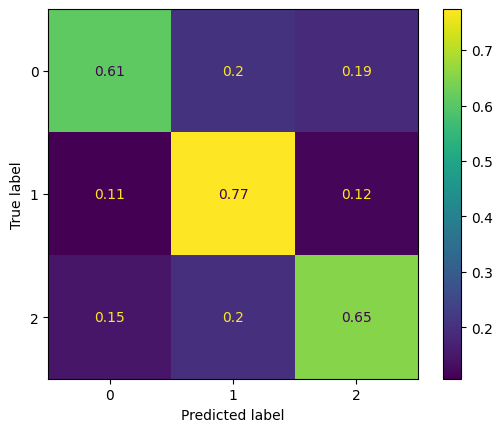
1 0.7739 0.7739 0.7739 65996

2 0.6036 0.6520 0.6269 35100

accuracy 0.6976 139543

macro avg 0.6780 0.6781 0.6774 139543

weighted avg 0.6987 0.6976 0.6977 139543



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Run a Market Simulation with AxialLOB[¶](http://localhost:8892/notebooks/DL_Project/AxialLOB.ipynb" \l "Run-a-Market-Simulation-with-AxialLOB" \t "_self)**

class BuySellDataset(data.Dataset):

"""

Dataset to handle buying and selling simulation

"""

def \_\_init\_\_(self, x, num\_classes, dim, rescale\_prices: bool = True):

"""Initialization"""

self.num\_classes = num\_classes

self.dim = dim

self.x = x

self.prices = self.compute\_mid\_prices(x, rescale\_prices)

self.length = x.shape[0] - T - self.dim + 1

x = torch.from\_numpy(x)

self.x = torch.unsqueeze(x, 1)

self.prices = torch.from\_numpy(self.prices)

def compute\_mid\_prices(self, x, rescale\_prices: bool = True):

"""

Compute the mid-price as the average of ask and bid prices.

Assuming ask and bid prices are alternating in the columns.

"""

mid\_prices = []

for i in range(x.shape[0]):

ask\_prices = x[i, 0] # Assuming best ask is at column 1

bid\_prices = x[i, 20] # Assuming best bid is at column 21

mid\_price = (ask\_prices + bid\_prices) / 2 # Calculate mid-price

mid\_prices.append(mid\_price)

mid\_prices = np.array(mid\_prices)

if rescale\_prices:

mid\_prices = mid\_prices - np.min(mid\_prices) + 0.5 # ensure price >=0.5 so no negative or 0 prices

return mid\_prices

def \_\_len\_\_(self):

"""Denotes the total number of samples"""

return self.length

def \_\_getitem\_\_(self, i):

input = self.x[i:i+self.dim, :]

input = input.permute(1, 0, 2)

return input, self.prices[i]

buy\_sell\_dataset\_test = BuySellDataset(dec\_test, num\_classes, dim, rescale\_prices=True)

buy\_sell\_test\_loader = torch.utils.data.DataLoader(dataset=buy\_sell\_dataset\_test, batch\_size=batch\_size, shuffle=False)

# Trading Strategy Simulation

def simulate\_trading\_debug(preds, prices, initial\_cash=10000, transaction\_cost=0.001):

cash = initial\_cash

stock = 0

print(f"Initial cash: {cash}, transaction cost: {transaction\_cost}")

portfolio\_value = initial\_cash

final\_price = prices[-1]

for i, (pred, price) in enumerate(zip(preds, prices)):

if pred == 2 and cash != 0: # Buy signal

num\_shares = cash / (price \* (1 + transaction\_cost))

cash -= num\_shares \* price \* (1 + transaction\_cost)

stock += num\_shares

print(f"BUY at step {i}: Bought {num\_shares:.2f} shares at {price:.2f}, Cash: {cash:.2f}, Stock: {stock:.2f}")

elif pred == 0 and stock != 0: # Sell signal

cash += stock \* price \* (1 - transaction\_cost)

print(f"SELL at step {i}: Sold {stock:.2f} shares at {price:.2f}, Cash: {cash:.2f}")

stock = 0

portfolio\_value = cash + stock \* price

if portfolio\_value <= 0.001:

final\_price = price

print("Ending simulation early. Model ran out of assets")

break

# Hold signal does nothing

print(f"Final Portfolio Value: {portfolio\_value:.2f}, Remaining Cash: {cash:.2f}, Remaining Stock at price {final\_price:.2f}: {stock:.2f}. Initial price was {prices[0]}")

print(f"Amount earned: {portfolio\_value - initial\_cash}")

return portfolio\_value

# Evaluation with Debugging

def evaluate\_strategy\_debug(model: nn.Module):

"""

Given a trained model, test how well it does at trading

"""

model.eval()

all\_preds = []

prices\_test = []

with torch.no\_grad():

for inputs, prices in buy\_sell\_test\_loader:

inputs = inputs.to(device, dtype=torch.float)

outputs = model(inputs)

preds = torch.argmax(outputs, dim=1)

all\_preds.append(preds.cpu().numpy())

prices\_test.append(prices.cpu().numpy())

preds = np.concatenate(all\_preds)

prices\_test = np.concatenate(prices\_test)

if len(preds) >= 10:

print(f"Predictions: {preds[:10]} ... {preds[-10:]} (first 10 and last 10 predictions)")

return simulate\_trading\_debug(preds, prices\_test)  
  
**Test on AAPL Yahoo Finance Data**[**¶**](http://localhost:8892/notebooks/DL_Project/AxialLOB.ipynb#Test-on-AAPL-Yahoo-Finance-Data)

# Get all the models to train and test on the new dataset

def get\_models\_to\_test(load\_from\_pretrained: bool, pretrained\_loc: str | None = None) -> dict[str, nn.Module]:

models\_to\_test: dict[str, nn.Module] = {}

models\_to\_test["AxialLOB"] = AxialLOB(W, dim, c\_in\_axial, c\_out\_axial, c\_final, n\_heads, pool\_kernel, pool\_stride).to(device)

models\_to\_test["DeepLOB"] = DeepLOB(W, hidden\_size, num\_classes).to(device)

models\_to\_test["B\_TABL"] = B\_TABL(W, num\_classes).to(device)

models\_to\_test["C\_TABL"] = C\_TABL(W, num\_classes).to(device)

models\_to\_test["DeepLOBSeq2Seq"] = DeepLOBSeq2Seq(W, hidden\_size, num\_classes).to(device)

models\_to\_test["DeepLOBAttention"] = DeepLOBAttention(W, hidden\_size, num\_classes).to(device)

if load\_from\_pretrained:

print("Attempting to load from pretrained...")

load\_from\_saved(models\_to\_test, pretrained\_loc)

return models\_to\_test  
import yfinance as yf

import pandas as pd

def fetch\_stock\_data(ticker: str, start\_date: str | None, end\_date: str | None, period: str | None, perform\_normilzation: bool, get\_short\_data: bool) -> pd.DataFrame:

"""

Fetch historical stock data and approximate order book features.

The LOB data provided in the FI-2010 dataset is not directly avaliable.

Args:

ticker (str): Stock ticker symbol.

start\_date (Optional[str]): Start date (YYYY-MM-DD).

end\_date (Optional[str]): End date (YYYY-MM-DD).

Returns:

pd.DataFrame: Data with approximated order book features.

"""

# Download OHLCV data

if get\_short\_data:

if period is None:

raise ValueError("Period must be specified for short data")

df = yf.download(ticker, period=period, interval="1m", progress=False)

print(f"Loading 1 minute interval data over the last {period}...")

else:

if start\_date is None or end\_date is None:

raise ValueError("Start and end date must be specified for longer data")

print(f"Loading 1 day interval data over the period {start\_date} to {end\_date}")

df = yf.download(ticker, start=start\_date, end=end\_date, progress=False)

df = df[['Open', 'High', 'Low', 'Close', 'Volume']].copy()

# Calculate mid-price and range

df['Mid\_Price'] = (df['High'] + df['Low']) / 2

df['Range'] = df['High'] - df['Low']

# Approximate ask/bid prices

for i in range(10):

weight = (i + 1) / 10

df[f'Ask\_Price\_{i+1}'] = df['Mid\_Price'] + weight \* 0.5 \* df['Range']

df[f'Bid\_Price\_{i+1}'] = df['Mid\_Price'] - weight \* 0.5 \* df['Range']

# Approximate ask/bid volumes (split volume evenly as we lack granularity)

total\_volume = df['Volume']

for i in range(10):

df[f'Ask\_Volume\_{i+1}'] = total\_volume / 20 # Split half for ask

df[f'Bid\_Volume\_{i+1}'] = total\_volume / 20 # Split half for bid

# Drop unused columns

df.drop(columns=['Open', 'High', 'Low', 'Close', 'Volume', 'Mid\_Price', 'Range'], inplace=True)

# Drop NaN rows

df.dropna(inplace=True)

if perform\_normilzation:

for column in df.columns:

col\_mean = df[column].mean()

col\_std = df[column].std()

df[column] = (df[column] - col\_mean) / col\_std

return df

def create\_sequences(data, dim, horizon):

"""

Create sequences for training/testing based on FI-2010 format.

Args:

data (np.ndarray): Input data (order book features).

dim (int): Sequence length.

horizon (int): Prediction horizon.

Returns:

tuple: Sequences (features), labels, and prices.

"""

sequences, labels, prices = [], [], []

N = data.shape[0]

for i in range(N - dim - horizon + 1):

seq = data[i:i + dim, :] # Dimensional sequence (10 ask, 10 ask volume, etc.)

# Use future mid-price to define labels

future\_mid\_price = (data[i + dim + horizon - 1, 9] + data[i + dim + horizon - 1, 19]) / 2 # Ask10 + Bid10

current\_mid\_price = (data[i + dim - 1, 9] + data[i + dim - 1, 19]) / 2 # Ask10 + Bid10

price\_change = (future\_mid\_price - current\_mid\_price) / current\_mid\_price

# Label: 2 (Buy), 1 (Hold), 0 (Sell)

label = 2 if price\_change > 0.001 else (0 if price\_change < -0.001 else 1)

sequences.append(seq)

labels.append(label)

prices.append(future\_mid\_price)

return np.array(sequences), np.array(labels, dtype=np.int64), np.array(prices)

class YahooDataset(data.Dataset):

"""Characterizes a dataset for PyTorch"""

def \_\_init\_\_(self, x, y, num\_classes, dim):

"""Initialization"""

self.num\_classes = num\_classes

self.dim = dim

self.x = x

self.y = y

self.length = x.shape[0] - T - self.dim + 1

x = torch.from\_numpy(x)

self.x = torch.unsqueeze(x, 1)

self.y = torch.from\_numpy(y)

def \_\_len\_\_(self):

"""Denotes the total number of samples"""

return self.length

def \_\_getitem\_\_(self, i):

return self.x[i, :], self.y[i]

# Parameters

ticker = "AAPL"

start\_date = "2021-01-01"

end\_date = "2024-10-10"

period = "5d"

# Fetch and prepare data

# If we are using pretrained models on the FI-2010 dataset, we need to normalize the data

# The FI-2010 dataset has a Z-norm applied to it

stock\_data = fetch\_stock\_data(ticker, start\_date=start\_date, end\_date=end\_date, period=None, perform\_normilzation=False, get\_short\_data=False)

data\_np\_long = stock\_data.to\_numpy()

stock\_data = fetch\_stock\_data(ticker, start\_date=None, end\_date=None, period=period, perform\_normilzation=True, get\_short\_data=True)

data\_np\_short = stock\_data.to\_numpy()

# Create sequences

sequences\_long, labels\_long, prices\_long = create\_sequences(data\_np\_long, dim, horizon)

sequences\_short, labels\_short, prices\_short = create\_sequences(data\_np\_short, dim, horizon)

print(f"Shape of Yahoo Finance Data over 1 day intervals: {sequences\_long.shape}")

print(f"Shape of Yahoo Finance Data over 1 minute intervals: {sequences\_short.shape}")

def get\_data(X, y, prices) -> tuple[data.DataLoader, data.DataLoader, data.DataLoader, np.ndarray]:

# Split the data into training, validation, and testing sets

# Perform train-test 80/20 split

train\_size = int(0.8 \* len(X))

X\_train, X\_test = X[:train\_size], X[train\_size:]

y\_train, y\_test = y[:train\_size], y[train\_size:]

prices\_test = prices[train\_size:]

# Split the training data into 90 train, 10 val

train\_size = int(0.9 \* len(X\_train))

X\_train, X\_val = X\_train[:train\_size], X\_train[train\_size:]

y\_train, y\_val = y\_train[:train\_size], y\_train[train\_size:]

# Reuse DataLoader setup

train\_loader = data.DataLoader(

YahooDataset(X\_train, y\_train, num\_classes, dim), batch\_size=batch\_size, shuffle=True

)

test\_loader = data.DataLoader(

YahooDataset(X\_test, y\_test, num\_classes, dim), batch\_size=batch\_size, shuffle=False

)

val\_loader = data.DataLoader(

YahooDataset(X\_val, y\_val, num\_classes, dim), batch\_size=batch\_size, shuffle=False

)

inputs, labels = next(iter(train\_loader))

print("Input batch shape:", inputs.shape)

print("Label batch shape:", labels.shape)

return train\_loader, val\_loader, test\_loader, prices\_test  
  
save\_loc = "AAPL"

epochs = 20

TRAIN\_APPL = True

# Train and evaluate models on long AAPL data

print(f"Evaluating models from scratch on 1 day interval AAPL data from {start\_date} to {end\_date}")

train\_loader\_long, val\_loader\_long, test\_loader\_long, prices\_test\_long = get\_data(sequences\_long, labels\_long, prices\_long)

evaluate\_all(get\_models\_to\_test(False, None), epochs, train\_loader\_long, val\_loader\_long, test\_loader\_long, train\_model=TRAIN\_APPL, save\_dir=save\_loc + "\_long")  
  
Evaluating models from scratch on 1 day interval APPL data from 2021-01-01 to 2024-10-10

Input batch shape: torch.Size([64, 1, 40, 40])

Label batch shape: torch.Size([64])

Training AxialLOB...

5%|▌ | 1/20 [00:01<00:21, 1.13s/it]

model saved at APPL\_long\AxialLOB.pt

Epoch 1/20, Train Loss: 0.9974, Validation Loss: 0.9838, Duration: 0:00:01.130509, Best Val Epoch: 0

10%|█ | 2/20 [00:01<00:17, 1.02it/s]

Epoch 2/20, Train Loss: 0.9506, Validation Loss: 0.9864, Duration: 0:00:00.868999, Best Val Epoch: 0

15%|█▌ | 3/20 [00:02<00:15, 1.07it/s]

model saved at APPL\_long\AxialLOB.pt

Epoch 3/20, Train Loss: 0.9268, Validation Loss: 0.9692, Duration: 0:00:00.877999, Best Val Epoch: 2

20%|██ | 4/20 [00:03<00:14, 1.10it/s]

Epoch 4/20, Train Loss: 0.9036, Validation Loss: 0.9731, Duration: 0:00:00.865999, Best Val Epoch: 2

Turning on additional grads for AxialLOB...

25%|██▌ | 5/20 [00:04<00:13, 1.10it/s]

Epoch 5/20, Train Loss: 0.8755, Validation Loss: 0.9804, Duration: 0:00:00.903999, Best Val Epoch: 2

30%|███ | 6/20 [00:05<00:12, 1.10it/s]

Epoch 6/20, Train Loss: 0.8376, Validation Loss: 0.9726, Duration: 0:00:00.903000, Best Val Epoch: 2

35%|███▌ | 7/20 [00:06<00:11, 1.10it/s]

Epoch 7/20, Train Loss: 0.8219, Validation Loss: 1.0054, Duration: 0:00:00.904001, Best Val Epoch: 2

40%|████ | 8/20 [00:07<00:10, 1.10it/s]

model saved at APPL\_long\AxialLOB.pt

Epoch 8/20, Train Loss: 0.7991, Validation Loss: 0.9235, Duration: 0:00:00.918997, Best Val Epoch: 7

45%|████▌ | 9/20 [00:08<00:10, 1.10it/s]

Epoch 9/20, Train Loss: 0.7677, Validation Loss: 0.9582, Duration: 0:00:00.906999, Best Val Epoch: 7

50%|█████ | 10/20 [00:09<00:09, 1.10it/s]

model saved at APPL\_long\AxialLOB.pt

Epoch 10/20, Train Loss: 0.7416, Validation Loss: 0.8955, Duration: 0:00:00.917000, Best Val Epoch: 9

55%|█████▌ | 11/20 [00:10<00:08, 1.09it/s]

model saved at APPL\_long\AxialLOB.pt

Epoch 11/20, Train Loss: 0.7380, Validation Loss: 0.8531, Duration: 0:00:00.921000, Best Val Epoch: 10

60%|██████ | 12/20 [00:11<00:07, 1.10it/s]

Epoch 12/20, Train Loss: 0.7252, Validation Loss: 0.9255, Duration: 0:00:00.909510, Best Val Epoch: 10

65%|██████▌ | 13/20 [00:11<00:06, 1.10it/s]

Epoch 13/20, Train Loss: 0.6974, Validation Loss: 0.9031, Duration: 0:00:00.908001, Best Val Epoch: 10

70%|███████ | 14/20 [00:12<00:05, 1.10it/s]

Epoch 14/20, Train Loss: 0.6873, Validation Loss: 0.8951, Duration: 0:00:00.907999, Best Val Epoch: 10

75%|███████▌ | 15/20 [00:13<00:04, 1.10it/s]

Epoch 15/20, Train Loss: 0.6678, Validation Loss: 0.9391, Duration: 0:00:00.906999, Best Val Epoch: 10

80%|████████ | 16/20 [00:14<00:03, 1.10it/s]

Epoch 16/20, Train Loss: 0.6631, Validation Loss: 0.9164, Duration: 0:00:00.907996, Best Val Epoch: 10

85%|████████▌ | 17/20 [00:15<00:02, 1.10it/s]

Epoch 17/20, Train Loss: 0.6507, Validation Loss: 0.9419, Duration: 0:00:00.909374, Best Val Epoch: 10

90%|█████████ | 18/20 [00:16<00:01, 1.10it/s]

Epoch 18/20, Train Loss: 0.6505, Validation Loss: 0.9181, Duration: 0:00:00.906997, Best Val Epoch: 10

95%|█████████▌| 19/20 [00:17<00:00, 1.10it/s]

Epoch 19/20, Train Loss: 0.6370, Validation Loss: 0.8738, Duration: 0:00:00.905998, Best Val Epoch: 10

100%|██████████| 20/20 [00:18<00:00, 1.09it/s]

Epoch 20/20, Train Loss: 0.6321, Validation Loss: 0.8860, Duration: 0:00:00.906999, Best Val Epoch: 10

Full time to train: 0:00:18.304394. Average time per epoch: 0:00:00.915220

Evaluating AxialLOB on test set

Test acc: 0.5000. Test Loss: 1.0616. Time on test: 0:00:00.050999

precision recall f1-score support

0 0.3333 0.0645 0.1081 62

1 0.0000 0.0000 0.0000 4

2 0.5159 0.9028 0.6566 72

accuracy 0.5000 138

macro avg 0.2831 0.3224 0.2549 138

weighted avg 0.4189 0.5000 0.3911 138

C:\Users\bmanv\AppData\Local\Temp\ipykernel\_11012\3999389156.py:36: FutureWarning: You are using `torch.load` with `weights\_only=False` (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See <https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models> for more details). In a future release, the default value for `weights\_only` will be flipped to `True`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add\_safe\_globals`. We recommend you start setting `weights\_only=True` for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.

model.load\_state\_dict(torch.load(save\_path, map\_location=device))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

A chart with different colored squares

Description automatically generated

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Training DeepLOB...

5%|▌ | 1/20 [00:00<00:02, 8.70it/s]

model saved at APPL\_long\DeepLOB.pt

Epoch 1/20, Train Loss: 0.9231, Validation Loss: 0.6888, Duration: 0:00:00.114999, Best Val Epoch: 0

10%|█ | 2/20 [00:00<00:01, 9.37it/s]

Epoch 2/20, Train Loss: 0.8643, Validation Loss: 0.7256, Duration: 0:00:00.100001, Best Val Epoch: 0

Epoch 3/20, Train Loss: 0.8565, Validation Loss: 0.6902, Duration: 0:00:00.101000, Best Val Epoch: 0

15%|█▌ | 3/20 [00:00<00:01, 9.60it/s]

Epoch 4/20, Train Loss: 0.8472, Validation Loss: 0.7199, Duration: 0:00:00.099997, Best Val Epoch: 0

25%|██▌ | 5/20 [00:00<00:01, 9.75it/s]

Epoch 5/20, Train Loss: 0.8604, Validation Loss: 0.6998, Duration: 0:00:00.100996, Best Val Epoch: 0

35%|███▌ | 7/20 [00:00<00:01, 9.49it/s]

Epoch 6/20, Train Loss: 0.8560, Validation Loss: 0.7176, Duration: 0:00:00.098997, Best Val Epoch: 0

model saved at APPL\_long\DeepLOB.pt

Epoch 7/20, Train Loss: 0.8552, Validation Loss: 0.6780, Duration: 0:00:00.112000, Best Val Epoch: 6

Epoch 8/20, Train Loss: 0.8457, Validation Loss: 0.7088, Duration: 0:00:00.100000, Best Val Epoch: 6

50%|█████ | 10/20 [00:01<00:01, 9.77it/s]

Epoch 9/20, Train Loss: 0.8452, Validation Loss: 0.6858, Duration: 0:00:00.101001, Best Val Epoch: 6

Epoch 10/20, Train Loss: 0.8402, Validation Loss: 0.6849, Duration: 0:00:00.100000, Best Val Epoch: 6

60%|██████ | 12/20 [00:01<00:00, 9.59it/s]

model saved at APPL\_long\DeepLOB.pt

Epoch 11/20, Train Loss: 0.8419, Validation Loss: 0.6735, Duration: 0:00:00.111000, Best Val Epoch: 10

Epoch 12/20, Train Loss: 0.8286, Validation Loss: 0.7082, Duration: 0:00:00.100997, Best Val Epoch: 10

Epoch 13/20, Train Loss: 0.8061, Validation Loss: 0.7127, Duration: 0:00:00.100999, Best Val Epoch: 10

75%|███████▌ | 15/20 [00:01<00:00, 9.75it/s]

Epoch 14/20, Train Loss: 0.8034, Validation Loss: 0.7213, Duration: 0:00:00.101001, Best Val Epoch: 10

Epoch 15/20, Train Loss: 0.7461, Validation Loss: 0.7370, Duration: 0:00:00.100999, Best Val Epoch: 10

Epoch 16/20, Train Loss: 0.7328, Validation Loss: 0.7539, Duration: 0:00:00.102000, Best Val Epoch: 10

90%|█████████ | 18/20 [00:01<00:00, 9.81it/s]

Epoch 17/20, Train Loss: 0.7159, Validation Loss: 0.9486, Duration: 0:00:00.102001, Best Val Epoch: 10

Epoch 18/20, Train Loss: 0.6383, Validation Loss: 0.8477, Duration: 0:00:00.100000, Best Val Epoch: 10

Epoch 19/20, Train Loss: 0.5817, Validation Loss: 0.9898, Duration: 0:00:00.101000, Best Val Epoch: 10

100%|██████████| 20/20 [00:02<00:00, 9.69it/s]

Epoch 20/20, Train Loss: 0.5239, Validation Loss: 0.8161, Duration: 0:00:00.100000, Best Val Epoch: 10

Full time to train: 0:00:02.065999. Average time per epoch: 0:00:00.103300

Evaluating DeepLOB on test set

Test acc: 0.5290. Test Loss: 0.8689. Time on test: 0:00:00.007999

precision recall f1-score support

0 1.0000 0.0161 0.0317 62

1 0.0000 0.0000 0.0000 4

2 0.5255 1.0000 0.6890 72

accuracy 0.5290 138

macro avg 0.5085 0.3387 0.2402 138

weighted avg 0.7235 0.5290 0.3737 138

C:\Users\bmanv\AppData\Local\Temp\ipykernel\_11012\3999389156.py:36: FutureWarning: You are using `torch.load` with `weights\_only=False` (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See <https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models> for more details). In a future release, the default value for `weights\_only` will be flipped to `True`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add\_safe\_globals`. We recommend you start setting `weights\_only=True` for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.

model.load\_state\_dict(torch.load(save\_path, map\_location=device))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

A chart of a blue and yellow box

Description automatically generated with medium confidence

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Training B\_TABL...

0%| | 0/20 [00:00<?, ?it/s]

model saved at APPL\_long\B\_TABL.pt

Epoch 1/20, Train Loss: 1271557.1609, Validation Loss: 180168.9062, Duration: 0:00:00.021998, Best Val Epoch: 0

model saved at APPL\_long\B\_TABL.pt

Epoch 2/20, Train Loss: 272201.1016, Validation Loss: 32425.3477, Duration: 0:00:00.027001, Best Val Epoch: 1

Epoch 3/20, Train Loss: 216155.3758, Validation Loss: 40117.8633, Duration: 0:00:00.023001, Best Val Epoch: 1

Epoch 4/20, Train Loss: 152359.9680, Validation Loss: 57602.7070, Duration: 0:00:00.016998, Best Val Epoch: 1

25%|██▌ | 5/20 [00:00<00:00, 47.17it/s]

Epoch 5/20, Train Loss: 85899.9246, Validation Loss: 53536.0547, Duration: 0:00:00.016001, Best Val Epoch: 1

model saved at APPL\_long\B\_TABL.pt

Epoch 6/20, Train Loss: 59855.1680, Validation Loss: 13046.7129, Duration: 0:00:00.018998, Best Val Epoch: 5

model saved at APPL\_long\B\_TABL.pt

Epoch 7/20, Train Loss: 35579.9489, Validation Loss: 12179.9521, Duration: 0:00:00.021001, Best Val Epoch: 6

model saved at APPL\_long\B\_TABL.pt

Epoch 8/20, Train Loss: 21103.4128, Validation Loss: 11314.2021, Duration: 0:00:00.017999, Best Val Epoch: 7

Epoch 9/20, Train Loss: 18849.8347, Validation Loss: 15978.0732, Duration: 0:00:00.017000, Best Val Epoch: 7

model saved at APPL\_long\B\_TABL.pt

Epoch 10/20, Train Loss: 19293.3765, Validation Loss: 10612.6055, Duration: 0:00:00.020000, Best Val Epoch: 9

55%|█████▌ | 11/20 [00:00<00:00, 50.21it/s]

model saved at APPL\_long\B\_TABL.pt

Epoch 11/20, Train Loss: 15642.6544, Validation Loss: 9466.3301, Duration: 0:00:00.020000, Best Val Epoch: 10

model saved at APPL\_long\B\_TABL.pt

Epoch 12/20, Train Loss: 30690.7900, Validation Loss: 6820.0386, Duration: 0:00:00.022000, Best Val Epoch: 11

Epoch 13/20, Train Loss: 27652.5277, Validation Loss: 19680.3262, Duration: 0:00:00.017000, Best Val Epoch: 11

model saved at APPL\_long\B\_TABL.pt

Epoch 14/20, Train Loss: 28326.1183, Validation Loss: 3960.6638, Duration: 0:00:00.019001, Best Val Epoch: 13

Epoch 15/20, Train Loss: 25262.4808, Validation Loss: 13036.4980, Duration: 0:00:00.019000, Best Val Epoch: 13

100%|██████████| 20/20 [00:00<00:00, 51.02it/s]

Epoch 16/20, Train Loss: 35877.1607, Validation Loss: 4230.7261, Duration: 0:00:00.019999, Best Val Epoch: 13

Epoch 17/20, Train Loss: 39107.0215, Validation Loss: 5478.6680, Duration: 0:00:00.018002, Best Val Epoch: 13

Epoch 18/20, Train Loss: 21620.3074, Validation Loss: 16236.4268, Duration: 0:00:00.019000, Best Val Epoch: 13

Epoch 19/20, Train Loss: 27373.7725, Validation Loss: 6429.9028, Duration: 0:00:00.019999, Best Val Epoch: 13

Epoch 20/20, Train Loss: 27233.3571, Validation Loss: 16724.4375, Duration: 0:00:00.016000, Best Val Epoch: 13

Full time to train: 0:00:00.394000. Average time per epoch: 0:00:00.019700

Evaluating B\_TABL on test set

Test acc: 0.4493. Test Loss: 29170.8509. Time on test: 0:00:00.001999

precision recall f1-score support

0 0.3684 0.2258 0.2800 62

1 0.0000 0.0000 0.0000 4

2 0.4898 0.6667 0.5647 72

accuracy 0.4493 138

macro avg 0.2861 0.2975 0.2816 138

weighted avg 0.4211 0.4493 0.4204 138

C:\Users\bmanv\AppData\Local\Temp\ipykernel\_11012\3999389156.py:36: FutureWarning: You are using `torch.load` with `weights\_only=False` (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See <https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models> for more details). In a future release, the default value for `weights\_only` will be flipped to `True`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add\_safe\_globals`. We recommend you start setting `weights\_only=True` for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.

model.load\_state\_dict(torch.load(save\_path, map\_location=device))

A chart with different colored squares

Description automatically generated

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Training C\_TABL...

25%|██▌ | 5/20 [00:00<00:00, 46.30it/s]

model saved at APPL\_long\C\_TABL.pt

Epoch 1/20, Train Loss: 622055.3344, Validation Loss: 54832.7383, Duration: 0:00:00.021999, Best Val Epoch: 0

Epoch 2/20, Train Loss: 185903.0451, Validation Loss: 167757.7188, Duration: 0:00:00.020998, Best Val Epoch: 0

Epoch 3/20, Train Loss: 166549.9197, Validation Loss: 141097.7344, Duration: 0:00:00.022001, Best Val Epoch: 0

model saved at APPL\_long\C\_TABL.pt

Epoch 4/20, Train Loss: 163087.0297, Validation Loss: 7845.5166, Duration: 0:00:00.024000, Best Val Epoch: 3

Epoch 5/20, Train Loss: 72051.0463, Validation Loss: 39469.3164, Duration: 0:00:00.018999, Best Val Epoch: 3

Epoch 6/20, Train Loss: 53444.4625, Validation Loss: 54152.9883, Duration: 0:00:00.019001, Best Val Epoch: 3

Epoch 7/20, Train Loss: 57533.3345, Validation Loss: 31464.9336, Duration: 0:00:00.017999, Best Val Epoch: 3

Epoch 8/20, Train Loss: 60941.1012, Validation Loss: 14633.6289, Duration: 0:00:00.017000, Best Val Epoch: 3

model saved at APPL\_long\C\_TABL.pt

Epoch 9/20, Train Loss: 45510.8074, Validation Loss: 4818.6821, Duration: 0:00:00.019002, Best Val Epoch: 8

Epoch 10/20, Train Loss: 16956.1570, Validation Loss: 18331.8047, Duration: 0:00:00.023999, Best Val Epoch: 8

80%|████████ | 16/20 [00:00<00:00, 47.63it/s]

Epoch 11/20, Train Loss: 24978.7020, Validation Loss: 32698.1953, Duration: 0:00:00.023000, Best Val Epoch: 8

Epoch 12/20, Train Loss: 34100.8108, Validation Loss: 9652.2852, Duration: 0:00:00.020999, Best Val Epoch: 8

Epoch 13/20, Train Loss: 21783.6362, Validation Loss: 16799.7344, Duration: 0:00:00.021001, Best Val Epoch: 8

Epoch 14/20, Train Loss: 9189.9433, Validation Loss: 12230.1211, Duration: 0:00:00.019999, Best Val Epoch: 8

Epoch 15/20, Train Loss: 10964.1611, Validation Loss: 5526.0215, Duration: 0:00:00.020000, Best Val Epoch: 8

model saved at APPL\_long\C\_TABL.pt

Epoch 16/20, Train Loss: 8129.4810, Validation Loss: 2531.4236, Duration: 0:00:00.024001, Best Val Epoch: 15

model saved at APPL\_long\C\_TABL.pt

Epoch 17/20, Train Loss: 4717.1405, Validation Loss: 2069.9568, Duration: 0:00:00.023998, Best Val Epoch: 16

model saved at APPL\_long\C\_TABL.pt

Epoch 18/20, Train Loss: 3980.6877, Validation Loss: 1480.3196, Duration: 0:00:00.023000, Best Val Epoch: 17

Epoch 19/20, Train Loss: 2726.3964, Validation Loss: 2281.7458, Duration: 0:00:00.017000, Best Val Epoch: 17

Epoch 20/20, Train Loss: 4519.1181, Validation Loss: 4644.3926, Duration: 0:00:00.019000, Best Val Epoch: 17

100%|██████████| 20/20 [00:00<00:00, 47.62it/s]

Full time to train: 0:00:00.422004. Average time per epoch: 0:00:00.021100

Evaluating C\_TABL on test set

Test acc: 0.3768. Test Loss: 6914.1140. Time on test: 0:00:00.003001

precision recall f1-score support

0 0.3103 0.1452 0.1978 62

1 0.0000 0.0000 0.0000 4

2 0.4526 0.5972 0.5150 72

accuracy 0.3768 138

macro avg 0.2543 0.2475 0.2376 138

weighted avg 0.3756 0.3768 0.3575 138

C:\Users\bmanv\AppData\Local\Temp\ipykernel\_11012\3999389156.py:36: FutureWarning: You are using `torch.load` with `weights\_only=False` (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See <https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models> for more details). In a future release, the default value for `weights\_only` will be flipped to `True`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add\_safe\_globals`. We recommend you start setting `weights\_only=True` for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.

model.load\_state\_dict(torch.load(save\_path, map\_location=device))

A chart with different colored squares

Description automatically generated

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Training DeepLOBSeq2Seq...

10%|█ | 2/20 [00:00<00:00, 19.42it/s]

model saved at APPL\_long\DeepLOBSeq2Seq.pt

Epoch 1/20, Train Loss: 0.9855, Validation Loss: 0.7453, Duration: 0:00:00.053000, Best Val Epoch: 0

model saved at APPL\_long\DeepLOBSeq2Seq.pt

Epoch 2/20, Train Loss: 0.8720, Validation Loss: 0.6869, Duration: 0:00:00.050000, Best Val Epoch: 1

Epoch 3/20, Train Loss: 0.8506, Validation Loss: 0.6973, Duration: 0:00:00.050001, Best Val Epoch: 1

Epoch 4/20, Train Loss: 0.8545, Validation Loss: 0.7103, Duration: 0:00:00.048002, Best Val Epoch: 1

Epoch 5/20, Train Loss: 0.8451, Validation Loss: 0.7146, Duration: 0:00:00.052000, Best Val Epoch: 1

25%|██▌ | 5/20 [00:00<00:00, 19.72it/s]

Epoch 6/20, Train Loss: 0.8551, Validation Loss: 0.7194, Duration: 0:00:00.049996, Best Val Epoch: 1

Epoch 7/20, Train Loss: 0.8703, Validation Loss: 0.6870, Duration: 0:00:00.049000, Best Val Epoch: 1

40%|████ | 8/20 [00:00<00:00, 19.99it/s]

Epoch 8/20, Train Loss: 0.8505, Validation Loss: 0.7417, Duration: 0:00:00.049002, Best Val Epoch: 1

Epoch 9/20, Train Loss: 0.8616, Validation Loss: 0.7129, Duration: 0:00:00.049999, Best Val Epoch: 1

55%|█████▌ | 11/20 [00:00<00:00, 20.68it/s]

Epoch 10/20, Train Loss: 0.8546, Validation Loss: 0.7046, Duration: 0:00:00.042999, Best Val Epoch: 1

Epoch 11/20, Train Loss: 0.8736, Validation Loss: 0.6962, Duration: 0:00:00.045001, Best Val Epoch: 1

Epoch 12/20, Train Loss: 0.8494, Validation Loss: 0.7130, Duration: 0:00:00.047000, Best Val Epoch: 1

Epoch 13/20, Train Loss: 0.8500, Validation Loss: 0.7120, Duration: 0:00:00.047000, Best Val Epoch: 1

Epoch 14/20, Train Loss: 0.8605, Validation Loss: 0.6977, Duration: 0:00:00.048999, Best Val Epoch: 1

70%|███████ | 14/20 [00:00<00:00, 20.68it/s]

Epoch 15/20, Train Loss: 0.8487, Validation Loss: 0.6970, Duration: 0:00:00.041001, Best Val Epoch: 1

Epoch 16/20, Train Loss: 0.8567, Validation Loss: 0.7031, Duration: 0:00:00.044002, Best Val Epoch: 1

Epoch 17/20, Train Loss: 0.8532, Validation Loss: 0.7058, Duration: 0:00:00.042999, Best Val Epoch: 1

85%|████████▌ | 17/20 [00:00<00:00, 21.40it/s]

Epoch 18/20, Train Loss: 0.8532, Validation Loss: 0.7035, Duration: 0:00:00.043998, Best Val Epoch: 1

Epoch 19/20, Train Loss: 0.8433, Validation Loss: 0.7014, Duration: 0:00:00.042000, Best Val Epoch: 1

100%|██████████| 20/20 [00:00<00:00, 21.16it/s]

C:\Users\bmanv\AppData\Local\Temp\ipykernel\_11012\3999389156.py:36: FutureWarning: You are using `torch.load` with `weights\_only=False` (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See <https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models> for more details). In a future release, the default value for `weights\_only` will be flipped to `True`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add\_safe\_globals`. We recommend you start setting `weights\_only=True` for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.

model.load\_state\_dict(torch.load(save\_path, map\_location=device))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

Epoch 20/20, Train Loss: 0.8545, Validation Loss: 0.7052, Duration: 0:00:00.042001, Best Val Epoch: 1

Full time to train: 0:00:00.947001. Average time per epoch: 0:00:00.047350

Evaluating DeepLOBSeq2Seq on test set

Test acc: 0.5217. Test Loss: 0.8630. Time on test: 0:00:00.003999

precision recall f1-score support

0 0.0000 0.0000 0.0000 62

1 0.0000 0.0000 0.0000 4

2 0.5217 1.0000 0.6857 72

accuracy 0.5217 138

macro avg 0.1739 0.3333 0.2286 138

weighted avg 0.2722 0.5217 0.3578 138

A chart of a blue and yellow box

Description automatically generated with medium confidence

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Training DeepLOBAttention...

0%| | 0/20 [00:00<?, ?it/s]

model saved at APPL\_long\DeepLOBAttention.pt

Epoch 1/20, Train Loss: 0.9796, Validation Loss: 0.7918, Duration: 0:00:00.043000, Best Val Epoch: 0

model saved at APPL\_long\DeepLOBAttention.pt

Epoch 2/20, Train Loss: 0.8824, Validation Loss: 0.7083, Duration: 0:00:00.039998, Best Val Epoch: 1

15%|█▌ | 3/20 [00:00<00:00, 25.00it/s]

model saved at APPL\_long\DeepLOBAttention.pt

Epoch 3/20, Train Loss: 0.8551, Validation Loss: 0.7065, Duration: 0:00:00.037001, Best Val Epoch: 2

Epoch 4/20, Train Loss: 0.8475, Validation Loss: 0.7075, Duration: 0:00:00.037999, Best Val Epoch: 2

model saved at APPL\_long\DeepLOBAttention.pt

Epoch 5/20, Train Loss: 0.8561, Validation Loss: 0.6818, Duration: 0:00:00.038001, Best Val Epoch: 4

Epoch 6/20, Train Loss: 0.8532, Validation Loss: 0.7027, Duration: 0:00:00.039000, Best Val Epoch: 4

30%|███ | 6/20 [00:00<00:00, 25.37it/s]

Epoch 7/20, Train Loss: 0.8480, Validation Loss: 0.7111, Duration: 0:00:00.035998, Best Val Epoch: 4

Epoch 8/20, Train Loss: 0.8503, Validation Loss: 0.6990, Duration: 0:00:00.035001, Best Val Epoch: 4

45%|████▌ | 9/20 [00:00<00:00, 26.63it/s]

Epoch 9/20, Train Loss: 0.8479, Validation Loss: 0.7003, Duration: 0:00:00.033999, Best Val Epoch: 4

Epoch 10/20, Train Loss: 0.8602, Validation Loss: 0.7004, Duration: 0:00:00.037001, Best Val Epoch: 4

Epoch 11/20, Train Loss: 0.8514, Validation Loss: 0.7059, Duration: 0:00:00.036001, Best Val Epoch: 4

60%|██████ | 12/20 [00:00<00:00, 26.98it/s]

Epoch 12/20, Train Loss: 0.8532, Validation Loss: 0.7149, Duration: 0:00:00.035998, Best Val Epoch: 4

Epoch 13/20, Train Loss: 0.8585, Validation Loss: 0.7059, Duration: 0:00:00.038000, Best Val Epoch: 4

Epoch 14/20, Train Loss: 0.8598, Validation Loss: 0.7093, Duration: 0:00:00.037001, Best Val Epoch: 4

75%|███████▌ | 15/20 [00:00<00:00, 27.17it/s]

Epoch 15/20, Train Loss: 0.8432, Validation Loss: 0.6977, Duration: 0:00:00.033002, Best Val Epoch: 4

Epoch 16/20, Train Loss: 0.8546, Validation Loss: 0.6896, Duration: 0:00:00.036998, Best Val Epoch: 4

Epoch 17/20, Train Loss: 0.8526, Validation Loss: 0.6941, Duration: 0:00:00.038001, Best Val Epoch: 4

90%|█████████ | 18/20 [00:00<00:00, 27.04it/s]

Epoch 18/20, Train Loss: 0.8538, Validation Loss: 0.7147, Duration: 0:00:00.035998, Best Val Epoch: 4

Epoch 19/20, Train Loss: 0.8534, Validation Loss: 0.7027, Duration: 0:00:00.038998, Best Val Epoch: 4

Epoch 20/20, Train Loss: 0.8586, Validation Loss: 0.7002, Duration: 0:00:00.036997, Best Val Epoch: 4

100%|██████████| 20/20 [00:00<00:00, 26.67it/s]

Full time to train: 0:00:00.752003. Average time per epoch: 0:00:00.037600

Evaluating DeepLOBAttention on test set

Test acc: 0.5217. Test Loss: 0.8611. Time on test: 0:00:00.005999

precision recall f1-score support

0 0.0000 0.0000 0.0000 62

1 0.0000 0.0000 0.0000 4

2 0.5217 1.0000 0.6857 72

accuracy 0.5217 138

macro avg 0.1739 0.3333 0.2286 138

weighted avg 0.2722 0.5217 0.3578 138

C:\Users\bmanv\AppData\Local\Temp\ipykernel\_11012\3999389156.py:36: FutureWarning: You are using `torch.load` with `weights\_only=False` (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See <https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models> for more details). In a future release, the default value for `weights\_only` will be flipped to `True`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add\_safe\_globals`. We recommend you start setting `weights\_only=True` for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.

model.load\_state\_dict(torch.load(save\_path, map\_location=device))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

A chart of a blue and yellow box

Description automatically generated with medium confidence

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Train and evaluate models on short AAPL data

print(f"Evaluating models from scratch on 1 minute interval AAPL data over {period}")

train\_loader\_short, val\_loader\_short, test\_loader\_short, prices\_test\_short = get\_data(sequences\_short, labels\_short, prices\_short)  
  
Evaluating models from scratch on 1 minute interval APPL data over 5d

Input batch shape: torch.Size([64, 1, 40, 40])

Label batch shape: torch.Size([64])

[ ]:

print("Train models from scratch on AAPL data")

evaluate\_all(get\_models\_to\_test(False, None), epochs, train\_loader\_short, val\_loader\_short, test\_loader\_short, train\_model=TRAIN\_APPL, save\_dir=save\_loc + "\_short(scratch)")  
  
Train models from scratch on APPL data

Training AxialLOB...

5%|▌ | 1/20 [00:02<00:40, 2.16s/it]

model saved at APPL\_short(scratch)\AxialLOB.pt

Epoch 1/20, Train Loss: 0.9765, Validation Loss: 0.9603, Duration: 0:00:02.157508, Best Val Epoch: 0

10%|█ | 2/20 [00:04<00:36, 2.01s/it]

Epoch 2/20, Train Loss: 0.9409, Validation Loss: 0.9653, Duration: 0:00:01.905000, Best Val Epoch: 0

15%|█▌ | 3/20 [00:05<00:33, 1.96s/it]

Epoch 3/20, Train Loss: 0.9312, Validation Loss: 1.0012, Duration: 0:00:01.904000, Best Val Epoch: 0

20%|██ | 4/20 [00:07<00:31, 1.94s/it]

Epoch 4/20, Train Loss: 0.9212, Validation Loss: 0.9933, Duration: 0:00:01.916000, Best Val Epoch: 0

Turning on additional grads for AxialLOB...

25%|██▌ | 5/20 [00:09<00:29, 1.96s/it]

Epoch 5/20, Train Loss: 0.9165, Validation Loss: 1.0316, Duration: 0:00:01.996261, Best Val Epoch: 0

30%|███ | 6/20 [00:11<00:27, 1.97s/it]

Epoch 6/20, Train Loss: 0.8926, Validation Loss: 1.0165, Duration: 0:00:01.995000, Best Val Epoch: 0

35%|███▌ | 7/20 [00:13<00:25, 1.98s/it]

Epoch 7/20, Train Loss: 0.8820, Validation Loss: 1.0148, Duration: 0:00:01.991999, Best Val Epoch: 0

40%|████ | 8/20 [00:15<00:23, 1.98s/it]

Epoch 8/20, Train Loss: 0.8828, Validation Loss: 1.0103, Duration: 0:00:01.993001, Best Val Epoch: 0

45%|████▌ | 9/20 [00:17<00:21, 1.99s/it]

Epoch 9/20, Train Loss: 0.8751, Validation Loss: 1.0475, Duration: 0:00:01.994999, Best Val Epoch: 0

50%|█████ | 10/20 [00:19<00:19, 1.99s/it]

Epoch 10/20, Train Loss: 0.8636, Validation Loss: 1.0539, Duration: 0:00:01.999999, Best Val Epoch: 0

55%|█████▌ | 11/20 [00:21<00:17, 1.99s/it]

Epoch 11/20, Train Loss: 0.8512, Validation Loss: 1.0556, Duration: 0:00:01.992997, Best Val Epoch: 0

60%|██████ | 12/20 [00:23<00:15, 1.99s/it]

Epoch 12/20, Train Loss: 0.8440, Validation Loss: 1.0637, Duration: 0:00:01.992678, Best Val Epoch: 0

65%|██████▌ | 13/20 [00:25<00:13, 1.99s/it]

Epoch 13/20, Train Loss: 0.8312, Validation Loss: 1.0449, Duration: 0:00:01.994001, Best Val Epoch: 0

70%|███████ | 14/20 [00:27<00:11, 2.00s/it]

Epoch 14/20, Train Loss: 0.8203, Validation Loss: 1.0731, Duration: 0:00:01.998998, Best Val Epoch: 0

75%|███████▌ | 15/20 [00:29<00:09, 2.00s/it]

Epoch 15/20, Train Loss: 0.8169, Validation Loss: 1.0565, Duration: 0:00:01.998000, Best Val Epoch: 0

80%|████████ | 16/20 [00:31<00:07, 2.00s/it]

Epoch 16/20, Train Loss: 0.8076, Validation Loss: 1.0663, Duration: 0:00:02.001001, Best Val Epoch: 0

85%|████████▌ | 17/20 [00:33<00:05, 2.00s/it]

Epoch 17/20, Train Loss: 0.7953, Validation Loss: 1.0614, Duration: 0:00:01.998999, Best Val Epoch: 0

90%|█████████ | 18/20 [00:35<00:03, 2.00s/it]

Epoch 18/20, Train Loss: 0.7903, Validation Loss: 1.0707, Duration: 0:00:02.002000, Best Val Epoch: 0

95%|█████████▌| 19/20 [00:37<00:02, 2.00s/it]

Epoch 19/20, Train Loss: 0.7969, Validation Loss: 1.0733, Duration: 0:00:02.001996, Best Val Epoch: 0

100%|██████████| 20/20 [00:39<00:00, 1.99s/it]

Epoch 20/20, Train Loss: 0.7799, Validation Loss: 1.0767, Duration: 0:00:02.000163, Best Val Epoch: 0

Full time to train: 0:00:39.855620. Average time per epoch: 0:00:01.992781

Evaluating AxialLOB on test set

Test acc: 0.4926. Test Loss: 0.9685. Time on test: 0:00:00.120000

precision recall f1-score support

0 0.4167 0.0303 0.0565 165

1 0.0000 0.0000 0.0000 4

2 0.4954 0.9583 0.6531 168

accuracy 0.4926 337

macro avg 0.3040 0.3295 0.2365 337

weighted avg 0.4510 0.4926 0.3533 337

C:\Users\bmanv\AppData\Local\Temp\ipykernel\_11012\3999389156.py:36: FutureWarning: You are using `torch.load` with `weights\_only=False` (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See <https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models> for more details). In a future release, the default value for `weights\_only` will be flipped to `True`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add\_safe\_globals`. We recommend you start setting `weights\_only=True` for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.

model.load\_state\_dict(torch.load(save\_path, map\_location=device))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

A chart with different colored squares

Description automatically generated

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Training DeepLOB...

5%|▌ | 1/20 [00:00<00:04, 4.29it/s]

model saved at APPL\_short(scratch)\DeepLOB.pt

Epoch 1/20, Train Loss: 0.8190, Validation Loss: 0.7701, Duration: 0:00:00.233000, Best Val Epoch: 0

10%|█ | 2/20 [00:00<00:04, 4.28it/s]

model saved at APPL\_short(scratch)\DeepLOB.pt

Epoch 2/20, Train Loss: 0.7429, Validation Loss: 0.7446, Duration: 0:00:00.234000, Best Val Epoch: 1

15%|█▌ | 3/20 [00:00<00:03, 4.40it/s]

Epoch 3/20, Train Loss: 0.7341, Validation Loss: 0.7543, Duration: 0:00:00.219999, Best Val Epoch: 1

20%|██ | 4/20 [00:00<00:03, 4.44it/s]

Epoch 4/20, Train Loss: 0.7315, Validation Loss: 0.7690, Duration: 0:00:00.221000, Best Val Epoch: 1

25%|██▌ | 5/20 [00:01<00:03, 4.48it/s]

Epoch 5/20, Train Loss: 0.7313, Validation Loss: 0.7462, Duration: 0:00:00.220001, Best Val Epoch: 1

30%|███ | 6/20 [00:01<00:03, 4.49it/s]

Epoch 6/20, Train Loss: 0.7278, Validation Loss: 0.7616, Duration: 0:00:00.219998, Best Val Epoch: 1

35%|███▌ | 7/20 [00:01<00:02, 4.50it/s]

Epoch 7/20, Train Loss: 0.7234, Validation Loss: 0.7474, Duration: 0:00:00.219998, Best Val Epoch: 1

40%|████ | 8/20 [00:01<00:02, 4.50it/s]

Epoch 8/20, Train Loss: 0.7122, Validation Loss: 0.7595, Duration: 0:00:00.221998, Best Val Epoch: 1

45%|████▌ | 9/20 [00:02<00:02, 4.51it/s]

Epoch 9/20, Train Loss: 0.7052, Validation Loss: 0.7645, Duration: 0:00:00.219998, Best Val Epoch: 1

50%|█████ | 10/20 [00:02<00:02, 4.50it/s]

Epoch 10/20, Train Loss: 0.7015, Validation Loss: 0.7911, Duration: 0:00:00.220999, Best Val Epoch: 1

55%|█████▌ | 11/20 [00:02<00:02, 4.50it/s]

Epoch 11/20, Train Loss: 0.6709, Validation Loss: 0.8030, Duration: 0:00:00.222001, Best Val Epoch: 1

60%|██████ | 12/20 [00:02<00:01, 4.52it/s]

Epoch 12/20, Train Loss: 0.6628, Validation Loss: 0.8893, Duration: 0:00:00.218000, Best Val Epoch: 1

65%|██████▌ | 13/20 [00:02<00:01, 4.51it/s]

Epoch 13/20, Train Loss: 0.6590, Validation Loss: 0.8283, Duration: 0:00:00.221000, Best Val Epoch: 1

70%|███████ | 14/20 [00:03<00:01, 4.52it/s]

Epoch 14/20, Train Loss: 0.6571, Validation Loss: 0.8847, Duration: 0:00:00.220000, Best Val Epoch: 1

75%|███████▌ | 15/20 [00:03<00:01, 4.53it/s]

Epoch 15/20, Train Loss: 0.6302, Validation Loss: 0.8736, Duration: 0:00:00.219999, Best Val Epoch: 1

80%|████████ | 16/20 [00:03<00:00, 4.53it/s]

Epoch 16/20, Train Loss: 0.6075, Validation Loss: 0.9612, Duration: 0:00:00.218997, Best Val Epoch: 1

85%|████████▌ | 17/20 [00:03<00:00, 4.54it/s]

Epoch 17/20, Train Loss: 0.5837, Validation Loss: 0.9919, Duration: 0:00:00.218999, Best Val Epoch: 1

90%|█████████ | 18/20 [00:04<00:00, 4.53it/s]

Epoch 18/20, Train Loss: 0.5810, Validation Loss: 1.0345, Duration: 0:00:00.219999, Best Val Epoch: 1

95%|█████████▌| 19/20 [00:04<00:00, 4.53it/s]

Epoch 19/20, Train Loss: 0.5540, Validation Loss: 1.0661, Duration: 0:00:00.221001, Best Val Epoch: 1

100%|██████████| 20/20 [00:04<00:00, 4.50it/s]

Epoch 20/20, Train Loss: 0.5298, Validation Loss: 1.1740, Duration: 0:00:00.220000, Best Val Epoch: 1

Full time to train: 0:00:04.447000. Average time per epoch: 0:00:00.222350

Evaluating DeepLOB on test set

Test acc: 0.4955. Test Loss: 0.7428. Time on test: 0:00:00.020000

precision recall f1-score support

0 0.4927 0.8182 0.6150 165

1 0.0000 0.0000 0.0000 4

2 0.5079 0.1905 0.2771 168

accuracy 0.4955 337

macro avg 0.3335 0.3362 0.2974 337

weighted avg 0.4944 0.4955 0.4392 337

C:\Users\bmanv\AppData\Local\Temp\ipykernel\_11012\3999389156.py:36: FutureWarning: You are using `torch.load` with `weights\_only=False` (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See <https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models> for more details). In a future release, the default value for `weights\_only` will be flipped to `True`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add\_safe\_globals`. We recommend you start setting `weights\_only=True` for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.

model.load\_state\_dict(torch.load(save\_path, map\_location=device))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

A chart of different colored squares

Description automatically generated

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Training B\_TABL...

0%| | 0/20 [00:00<?, ?it/s]

model saved at APPL\_short(scratch)\B\_TABL.pt

Epoch 1/20, Train Loss: 0.9129, Validation Loss: 0.8654, Duration: 0:00:00.039999, Best Val Epoch: 0

Epoch 2/20, Train Loss: 0.7827, Validation Loss: 0.9772, Duration: 0:00:00.037000, Best Val Epoch: 0

15%|█▌ | 3/20 [00:00<00:00, 27.03it/s]

Epoch 3/20, Train Loss: 0.7309, Validation Loss: 0.8715, Duration: 0:00:00.034000, Best Val Epoch: 0

model saved at APPL\_short(scratch)\B\_TABL.pt

Epoch 4/20, Train Loss: 0.7170, Validation Loss: 0.8644, Duration: 0:00:00.039004, Best Val Epoch: 3

Epoch 5/20, Train Loss: 0.7133, Validation Loss: 0.9251, Duration: 0:00:00.035997, Best Val Epoch: 3

Epoch 6/20, Train Loss: 0.6770, Validation Loss: 0.8726, Duration: 0:00:00.039999, Best Val Epoch: 3

30%|███ | 6/20 [00:00<00:00, 26.47it/s]

Epoch 7/20, Train Loss: 0.6835, Validation Loss: 0.9714, Duration: 0:00:00.034000, Best Val Epoch: 3

Epoch 8/20, Train Loss: 0.6829, Validation Loss: 1.1106, Duration: 0:00:00.035000, Best Val Epoch: 3

45%|████▌ | 9/20 [00:00<00:00, 27.50it/s]

Epoch 9/20, Train Loss: 0.6563, Validation Loss: 0.9640, Duration: 0:00:00.034001, Best Val Epoch: 3

Epoch 10/20, Train Loss: 0.6450, Validation Loss: 0.9402, Duration: 0:00:00.042002, Best Val Epoch: 3

Epoch 11/20, Train Loss: 0.6550, Validation Loss: 1.0092, Duration: 0:00:00.039998, Best Val Epoch: 3

60%|██████ | 12/20 [00:00<00:00, 26.55it/s]

Epoch 12/20, Train Loss: 0.6438, Validation Loss: 1.0366, Duration: 0:00:00.036000, Best Val Epoch: 3

Epoch 13/20, Train Loss: 0.6115, Validation Loss: 0.9717, Duration: 0:00:00.040000, Best Val Epoch: 3

Epoch 14/20, Train Loss: 0.6176, Validation Loss: 1.0962, Duration: 0:00:00.036000, Best Val Epoch: 3

75%|███████▌ | 15/20 [00:00<00:00, 26.89it/s]

Epoch 15/20, Train Loss: 0.6092, Validation Loss: 1.0021, Duration: 0:00:00.032999, Best Val Epoch: 3

Epoch 16/20, Train Loss: 0.6155, Validation Loss: 1.0926, Duration: 0:00:00.037000, Best Val Epoch: 3

Epoch 17/20, Train Loss: 0.5941, Validation Loss: 1.0495, Duration: 0:00:00.034002, Best Val Epoch: 3

90%|█████████ | 18/20 [00:00<00:00, 27.19it/s]

Epoch 18/20, Train Loss: 0.5823, Validation Loss: 1.0450, Duration: 0:00:00.036999, Best Val Epoch: 3

Epoch 19/20, Train Loss: 0.5708, Validation Loss: 1.0857, Duration: 0:00:00.033995, Best Val Epoch: 3

Epoch 20/20, Train Loss: 0.5581, Validation Loss: 1.1116, Duration: 0:00:00.032999, Best Val Epoch: 3

100%|██████████| 20/20 [00:00<00:00, 27.17it/s]

Full time to train: 0:00:00.736997. Average time per epoch: 0:00:00.036850

Evaluating B\_TABL on test set

Test acc: 0.5134. Test Loss: 0.8398. Time on test: 0:00:00.009002

precision recall f1-score support

0 0.5038 0.8121 0.6218 165

1 0.0000 0.0000 0.0000 4

2 0.5493 0.2321 0.3264 168

accuracy 0.5134 337

macro avg 0.3510 0.3481 0.3161 337

weighted avg 0.5205 0.5134 0.4671 337

C:\Users\bmanv\AppData\Local\Temp\ipykernel\_11012\3999389156.py:36: FutureWarning: You are using `torch.load` with `weights\_only=False` (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See <https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models> for more details). In a future release, the default value for `weights\_only` will be flipped to `True`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add\_safe\_globals`. We recommend you start setting `weights\_only=True` for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.

model.load\_state\_dict(torch.load(save\_path, map\_location=device))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

A chart of different colored squares

Description automatically generated

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Training C\_TABL...

0%| | 0/20 [00:00<?, ?it/s]

model saved at APPL\_short(scratch)\C\_TABL.pt

Epoch 1/20, Train Loss: 0.8427, Validation Loss: 0.8177, Duration: 0:00:00.048001, Best Val Epoch: 0

model saved at APPL\_short(scratch)\C\_TABL.pt

Epoch 2/20, Train Loss: 0.7416, Validation Loss: 0.8041, Duration: 0:00:00.045999, Best Val Epoch: 1

15%|█▌ | 3/20 [00:00<00:00, 20.98it/s]

model saved at APPL\_short(scratch)\C\_TABL.pt

Epoch 3/20, Train Loss: 0.7002, Validation Loss: 0.8037, Duration: 0:00:00.047001, Best Val Epoch: 2

Epoch 4/20, Train Loss: 0.6874, Validation Loss: 0.8399, Duration: 0:00:00.042996, Best Val Epoch: 2

Epoch 5/20, Train Loss: 0.6742, Validation Loss: 1.0313, Duration: 0:00:00.040000, Best Val Epoch: 2

30%|███ | 6/20 [00:00<00:00, 22.26it/s]

Epoch 6/20, Train Loss: 0.6723, Validation Loss: 0.9279, Duration: 0:00:00.045000, Best Val Epoch: 2

Epoch 7/20, Train Loss: 0.6591, Validation Loss: 0.8839, Duration: 0:00:00.039000, Best Val Epoch: 2

Epoch 8/20, Train Loss: 0.6394, Validation Loss: 0.9871, Duration: 0:00:00.037000, Best Val Epoch: 2

45%|████▌ | 9/20 [00:00<00:00, 23.69it/s]

Epoch 9/20, Train Loss: 0.6376, Validation Loss: 0.9533, Duration: 0:00:00.040000, Best Val Epoch: 2

Epoch 10/20, Train Loss: 0.6149, Validation Loss: 1.0085, Duration: 0:00:00.042001, Best Val Epoch: 2

60%|██████ | 12/20 [00:00<00:00, 23.74it/s]

Epoch 11/20, Train Loss: 0.5929, Validation Loss: 0.9645, Duration: 0:00:00.041997, Best Val Epoch: 2

Epoch 12/20, Train Loss: 0.5996, Validation Loss: 1.1941, Duration: 0:00:00.041000, Best Val Epoch: 2

Epoch 13/20, Train Loss: 0.6016, Validation Loss: 0.9345, Duration: 0:00:00.043000, Best Val Epoch: 2

Epoch 14/20, Train Loss: 0.5600, Validation Loss: 1.2100, Duration: 0:00:00.039000, Best Val Epoch: 2

75%|███████▌ | 15/20 [00:00<00:00, 23.83it/s]

Epoch 15/20, Train Loss: 0.5664, Validation Loss: 1.0302, Duration: 0:00:00.042002, Best Val Epoch: 2

90%|█████████ | 18/20 [00:00<00:00, 23.95it/s]

Epoch 16/20, Train Loss: 0.5808, Validation Loss: 1.2038, Duration: 0:00:00.039999, Best Val Epoch: 2

Epoch 17/20, Train Loss: 0.5697, Validation Loss: 0.9672, Duration: 0:00:00.041998, Best Val Epoch: 2

Epoch 18/20, Train Loss: 0.5578, Validation Loss: 1.0333, Duration: 0:00:00.040000, Best Val Epoch: 2

Epoch 19/20, Train Loss: 0.5207, Validation Loss: 1.0369, Duration: 0:00:00.042997, Best Val Epoch: 2

Epoch 20/20, Train Loss: 0.5018, Validation Loss: 1.1640, Duration: 0:00:00.039001, Best Val Epoch: 2

100%|██████████| 20/20 [00:00<00:00, 23.61it/s]

Full time to train: 0:00:00.849002. Average time per epoch: 0:00:00.042450

Evaluating C\_TABL on test set

Test acc: 0.5015. Test Loss: 0.7851. Time on test: 0:00:00.006001

precision recall f1-score support

0 0.5000 0.6606 0.5692 165

1 0.0000 0.0000 0.0000 4

2 0.5042 0.3571 0.4181 168

accuracy 0.5015 337

macro avg 0.3347 0.3392 0.3291 337

weighted avg 0.4962 0.5015 0.4871 337

C:\Users\bmanv\AppData\Local\Temp\ipykernel\_11012\3999389156.py:36: FutureWarning: You are using `torch.load` with `weights\_only=False` (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See <https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models> for more details). In a future release, the default value for `weights\_only` will be flipped to `True`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add\_safe\_globals`. We recommend you start setting `weights\_only=True` for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.

model.load\_state\_dict(torch.load(save\_path, map\_location=device))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

A chart with different colored squares

Description automatically generated

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Training DeepLOBSeq2Seq...

5%|▌ | 1/20 [00:00<00:01, 10.00it/s]

model saved at APPL\_short(scratch)\DeepLOBSeq2Seq.pt

Epoch 1/20, Train Loss: 0.9633, Validation Loss: 0.7677, Duration: 0:00:00.100001, Best Val Epoch: 0

model saved at APPL\_short(scratch)\DeepLOBSeq2Seq.pt

Epoch 2/20, Train Loss: 0.7521, Validation Loss: 0.7548, Duration: 0:00:00.097000, Best Val Epoch: 1

model saved at APPL\_short(scratch)\DeepLOBSeq2Seq.pt

15%|█▌ | 3/20 [00:00<00:01, 10.31it/s]

Epoch 3/20, Train Loss: 0.7422, Validation Loss: 0.7456, Duration: 0:00:00.094998, Best Val Epoch: 2

Epoch 4/20, Train Loss: 0.7474, Validation Loss: 0.7634, Duration: 0:00:00.101002, Best Val Epoch: 2

25%|██▌ | 5/20 [00:00<00:01, 9.99it/s]

Epoch 5/20, Train Loss: 0.7413, Validation Loss: 0.7490, Duration: 0:00:00.104999, Best Val Epoch: 2

Epoch 6/20, Train Loss: 0.7366, Validation Loss: 0.7473, Duration: 0:00:00.094000, Best Val Epoch: 2

model saved at APPL\_short(scratch)\DeepLOBSeq2Seq.pt

Epoch 7/20, Train Loss: 0.7336, Validation Loss: 0.7421, Duration: 0:00:00.092000, Best Val Epoch: 6

45%|████▌ | 9/20 [00:00<00:01, 10.28it/s]

Epoch 8/20, Train Loss: 0.7358, Validation Loss: 0.7514, Duration: 0:00:00.091000, Best Val Epoch: 6

Epoch 9/20, Train Loss: 0.7357, Validation Loss: 0.7613, Duration: 0:00:00.103002, Best Val Epoch: 6

55%|█████▌ | 11/20 [00:01<00:00, 10.24it/s]

Epoch 10/20, Train Loss: 0.7350, Validation Loss: 0.7529, Duration: 0:00:00.101999, Best Val Epoch: 6

Epoch 11/20, Train Loss: 0.7312, Validation Loss: 0.7448, Duration: 0:00:00.094004, Best Val Epoch: 6

Epoch 12/20, Train Loss: 0.7355, Validation Loss: 0.7479, Duration: 0:00:00.095998, Best Val Epoch: 6

65%|██████▌ | 13/20 [00:01<00:00, 10.35it/s]

Epoch 13/20, Train Loss: 0.7302, Validation Loss: 0.7464, Duration: 0:00:00.092000, Best Val Epoch: 6

Epoch 14/20, Train Loss: 0.7266, Validation Loss: 0.7589, Duration: 0:00:00.106000, Best Val Epoch: 6

model saved at APPL\_short(scratch)\DeepLOBSeq2Seq.pt

Epoch 15/20, Train Loss: 0.7292, Validation Loss: 0.7403, Duration: 0:00:00.092000, Best Val Epoch: 14

85%|████████▌ | 17/20 [00:01<00:00, 10.46it/s]

Epoch 16/20, Train Loss: 0.7321, Validation Loss: 0.7496, Duration: 0:00:00.089999, Best Val Epoch: 14

model saved at APPL\_short(scratch)\DeepLOBSeq2Seq.pt

Epoch 17/20, Train Loss: 0.7269, Validation Loss: 0.7321, Duration: 0:00:00.092001, Best Val Epoch: 16

Epoch 18/20, Train Loss: 0.7285, Validation Loss: 0.7624, Duration: 0:00:00.099999, Best Val Epoch: 16

100%|██████████| 20/20 [00:01<00:00, 10.33it/s]

Epoch 19/20, Train Loss: 0.7240, Validation Loss: 0.7583, Duration: 0:00:00.092001, Best Val Epoch: 16

Epoch 20/20, Train Loss: 0.7261, Validation Loss: 0.7358, Duration: 0:00:00.098000, Best Val Epoch: 16

Full time to train: 0:00:01.938000. Average time per epoch: 0:00:00.096900

Evaluating DeepLOBSeq2Seq on test set

Test acc: 0.5727. Test Loss: 0.7269. Time on test: 0:00:00.010000

precision recall f1-score support

0 0.5439 0.7879 0.6436 165

1 0.0000 0.0000 0.0000 4

2 0.6429 0.3750 0.4737 168

accuracy 0.5727 337

macro avg 0.3956 0.3876 0.3724 337

weighted avg 0.5868 0.5727 0.5512 337

C:\Users\bmanv\AppData\Local\Temp\ipykernel\_11012\3999389156.py:36: FutureWarning: You are using `torch.load` with `weights\_only=False` (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See <https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models> for more details). In a future release, the default value for `weights\_only` will be flipped to `True`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add\_safe\_globals`. We recommend you start setting `weights\_only=True` for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.

model.load\_state\_dict(torch.load(save\_path, map\_location=device))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

A chart of a graph

Description automatically generated with medium confidence

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Training DeepLOBAttention...

10%|█ | 2/20 [00:00<00:01, 11.11it/s]

model saved at APPL\_short(scratch)\DeepLOBAttention.pt

Epoch 1/20, Train Loss: 0.9143, Validation Loss: 0.8144, Duration: 0:00:00.094001, Best Val Epoch: 0

model saved at APPL\_short(scratch)\DeepLOBAttention.pt

Epoch 2/20, Train Loss: 0.7588, Validation Loss: 0.7510, Duration: 0:00:00.086000, Best Val Epoch: 1

Epoch 3/20, Train Loss: 0.7376, Validation Loss: 0.8048, Duration: 0:00:00.084999, Best Val Epoch: 1

20%|██ | 4/20 [00:00<00:01, 11.76it/s]

model saved at APPL\_short(scratch)\DeepLOBAttention.pt

Epoch 4/20, Train Loss: 0.7383, Validation Loss: 0.7453, Duration: 0:00:00.076002, Best Val Epoch: 3

Epoch 5/20, Train Loss: 0.7362, Validation Loss: 0.7682, Duration: 0:00:00.084998, Best Val Epoch: 3

30%|███ | 6/20 [00:00<00:01, 11.55it/s]

Epoch 6/20, Train Loss: 0.7350, Validation Loss: 0.7514, Duration: 0:00:00.092001, Best Val Epoch: 3

40%|████ | 8/20 [00:00<00:01, 11.71it/s]

Epoch 7/20, Train Loss: 0.7336, Validation Loss: 0.7540, Duration: 0:00:00.089000, Best Val Epoch: 3

Epoch 8/20, Train Loss: 0.7324, Validation Loss: 0.7606, Duration: 0:00:00.077000, Best Val Epoch: 3

Epoch 9/20, Train Loss: 0.7319, Validation Loss: 0.7714, Duration: 0:00:00.089999, Best Val Epoch: 3

50%|█████ | 10/20 [00:00<00:00, 11.49it/s]

Epoch 10/20, Train Loss: 0.7305, Validation Loss: 0.7527, Duration: 0:00:00.088002, Best Val Epoch: 3

Epoch 11/20, Train Loss: 0.7313, Validation Loss: 0.7545, Duration: 0:00:00.080999, Best Val Epoch: 3

60%|██████ | 12/20 [00:01<00:00, 11.65it/s]

Epoch 12/20, Train Loss: 0.7271, Validation Loss: 0.7620, Duration: 0:00:00.086002, Best Val Epoch: 3

70%|███████ | 14/20 [00:01<00:00, 11.69it/s]

Epoch 13/20, Train Loss: 0.7242, Validation Loss: 0.7732, Duration: 0:00:00.082000, Best Val Epoch: 3

Epoch 14/20, Train Loss: 0.7271, Validation Loss: 0.7671, Duration: 0:00:00.086999, Best Val Epoch: 3

Epoch 15/20, Train Loss: 0.7233, Validation Loss: 0.7759, Duration: 0:00:00.085999, Best Val Epoch: 3

80%|████████ | 16/20 [00:01<00:00, 11.63it/s]

Epoch 16/20, Train Loss: 0.7210, Validation Loss: 0.7823, Duration: 0:00:00.086003, Best Val Epoch: 3

Epoch 17/20, Train Loss: 0.7246, Validation Loss: 0.7773, Duration: 0:00:00.077999, Best Val Epoch: 3

90%|█████████ | 18/20 [00:01<00:00, 11.63it/s]

Epoch 18/20, Train Loss: 0.7194, Validation Loss: 0.7813, Duration: 0:00:00.094001, Best Val Epoch: 3

Epoch 19/20, Train Loss: 0.7200, Validation Loss: 0.7873, Duration: 0:00:00.091999, Best Val Epoch: 3

Epoch 20/20, Train Loss: 0.7167, Validation Loss: 0.7990, Duration: 0:00:00.085000, Best Val Epoch: 3

100%|██████████| 20/20 [00:01<00:00, 11.56it/s]

Full time to train: 0:00:01.731001. Average time per epoch: 0:00:00.086550

Evaluating DeepLOBAttention on test set

Test acc: 0.5074. Test Loss: 0.7414. Time on test: 0:00:00.009000

precision recall f1-score support

0 0.6364 0.0424 0.0795 165

1 0.0000 0.0000 0.0000 4

2 0.5031 0.9762 0.6640 168

accuracy 0.5074 337

macro avg 0.3798 0.3395 0.2478 337

weighted avg 0.5624 0.5074 0.3699 337

C:\Users\bmanv\AppData\Local\Temp\ipykernel\_11012\3999389156.py:36: FutureWarning: You are using `torch.load` with `weights\_only=False` (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See <https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models> for more details). In a future release, the default value for `weights\_only` will be flipped to `True`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add\_safe\_globals`. We recommend you start setting `weights\_only=True` for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.

model.load\_state\_dict(torch.load(save\_path, map\_location=device))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

A chart with numbers and a few colored squares

Description automatically generated with medium confidence

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

print("Train pre-trained models from FI-2010 on AAPL data")

evaluate\_all(get\_models\_to\_test(True, "all\_models"), epochs, train\_loader\_short, val\_loader\_short, test\_loader\_short, train\_model=TRAIN\_APPL, save\_dir=save\_loc + "\_short(pretrained)")  
  
print("Train models from scratch on AAPL data")

evaluate\_all(get\_models\_to\_test(False, None), epochs, train\_loader\_short, val\_loader\_short, test\_loader\_short, train\_model=TRAIN\_APPL, save\_dir=save\_loc + "\_short(scratch)")

Train models from scratch on APPL data

Training AxialLOB...

5%|▌ | 1/20 [00:02<00:40, 2.16s/it]

model saved at APPL\_short(scratch)\AxialLOB.pt

Epoch 1/20, Train Loss: 0.9765, Validation Loss: 0.9603, Duration: 0:00:02.157508, Best Val Epoch: 0

10%|█ | 2/20 [00:04<00:36, 2.01s/it]

Epoch 2/20, Train Loss: 0.9409, Validation Loss: 0.9653, Duration: 0:00:01.905000, Best Val Epoch: 0

15%|█▌ | 3/20 [00:05<00:33, 1.96s/it]

Epoch 3/20, Train Loss: 0.9312, Validation Loss: 1.0012, Duration: 0:00:01.904000, Best Val Epoch: 0

20%|██ | 4/20 [00:07<00:31, 1.94s/it]

Epoch 4/20, Train Loss: 0.9212, Validation Loss: 0.9933, Duration: 0:00:01.916000, Best Val Epoch: 0

Turning on additional grads for AxialLOB...

25%|██▌ | 5/20 [00:09<00:29, 1.96s/it]

Epoch 5/20, Train Loss: 0.9165, Validation Loss: 1.0316, Duration: 0:00:01.996261, Best Val Epoch: 0

30%|███ | 6/20 [00:11<00:27, 1.97s/it]

Epoch 6/20, Train Loss: 0.8926, Validation Loss: 1.0165, Duration: 0:00:01.995000, Best Val Epoch: 0

35%|███▌ | 7/20 [00:13<00:25, 1.98s/it]

Epoch 7/20, Train Loss: 0.8820, Validation Loss: 1.0148, Duration: 0:00:01.991999, Best Val Epoch: 0

40%|████ | 8/20 [00:15<00:23, 1.98s/it]

Epoch 8/20, Train Loss: 0.8828, Validation Loss: 1.0103, Duration: 0:00:01.993001, Best Val Epoch: 0

45%|████▌ | 9/20 [00:17<00:21, 1.99s/it]

Epoch 9/20, Train Loss: 0.8751, Validation Loss: 1.0475, Duration: 0:00:01.994999, Best Val Epoch: 0

50%|█████ | 10/20 [00:19<00:19, 1.99s/it]

Epoch 10/20, Train Loss: 0.8636, Validation Loss: 1.0539, Duration: 0:00:01.999999, Best Val Epoch: 0

55%|█████▌ | 11/20 [00:21<00:17, 1.99s/it]

Epoch 11/20, Train Loss: 0.8512, Validation Loss: 1.0556, Duration: 0:00:01.992997, Best Val Epoch: 0

60%|██████ | 12/20 [00:23<00:15, 1.99s/it]

Epoch 12/20, Train Loss: 0.8440, Validation Loss: 1.0637, Duration: 0:00:01.992678, Best Val Epoch: 0

65%|██████▌ | 13/20 [00:25<00:13, 1.99s/it]

Epoch 13/20, Train Loss: 0.8312, Validation Loss: 1.0449, Duration: 0:00:01.994001, Best Val Epoch: 0

70%|███████ | 14/20 [00:27<00:11, 2.00s/it]

Epoch 14/20, Train Loss: 0.8203, Validation Loss: 1.0731, Duration: 0:00:01.998998, Best Val Epoch: 0

75%|███████▌ | 15/20 [00:29<00:09, 2.00s/it]

Epoch 15/20, Train Loss: 0.8169, Validation Loss: 1.0565, Duration: 0:00:01.998000, Best Val Epoch: 0

80%|████████ | 16/20 [00:31<00:07, 2.00s/it]

Epoch 16/20, Train Loss: 0.8076, Validation Loss: 1.0663, Duration: 0:00:02.001001, Best Val Epoch: 0

85%|████████▌ | 17/20 [00:33<00:05, 2.00s/it]

Epoch 17/20, Train Loss: 0.7953, Validation Loss: 1.0614, Duration: 0:00:01.998999, Best Val Epoch: 0

90%|█████████ | 18/20 [00:35<00:03, 2.00s/it]

Epoch 18/20, Train Loss: 0.7903, Validation Loss: 1.0707, Duration: 0:00:02.002000, Best Val Epoch: 0

95%|█████████▌| 19/20 [00:37<00:02, 2.00s/it]

Epoch 19/20, Train Loss: 0.7969, Validation Loss: 1.0733, Duration: 0:00:02.001996, Best Val Epoch: 0

100%|██████████| 20/20 [00:39<00:00, 1.99s/it]

Epoch 20/20, Train Loss: 0.7799, Validation Loss: 1.0767, Duration: 0:00:02.000163, Best Val Epoch: 0

Full time to train: 0:00:39.855620. Average time per epoch: 0:00:01.992781

Evaluating AxialLOB on test set

Test acc: 0.4926. Test Loss: 0.9685. Time on test: 0:00:00.120000

precision recall f1-score support

0 0.4167 0.0303 0.0565 165

1 0.0000 0.0000 0.0000 4

2 0.4954 0.9583 0.6531 168

accuracy 0.4926 337

macro avg 0.3040 0.3295 0.2365 337

weighted avg 0.4510 0.4926 0.3533 337

C:\Users\bmanv\AppData\Local\Temp\ipykernel\_11012\3999389156.py:36: FutureWarning: You are using `torch.load` with `weights\_only=False` (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See <https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models> for more details). In a future release, the default value for `weights\_only` will be flipped to `True`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add\_safe\_globals`. We recommend you start setting `weights\_only=True` for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.

model.load\_state\_dict(torch.load(save\_path, map\_location=device))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

A chart with different colored squares

Description automatically generated

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Training DeepLOB...

5%|▌ | 1/20 [00:00<00:04, 4.29it/s]

model saved at APPL\_short(scratch)\DeepLOB.pt

Epoch 1/20, Train Loss: 0.8190, Validation Loss: 0.7701, Duration: 0:00:00.233000, Best Val Epoch: 0

10%|█ | 2/20 [00:00<00:04, 4.28it/s]

model saved at APPL\_short(scratch)\DeepLOB.pt

Epoch 2/20, Train Loss: 0.7429, Validation Loss: 0.7446, Duration: 0:00:00.234000, Best Val Epoch: 1

15%|█▌ | 3/20 [00:00<00:03, 4.40it/s]

Epoch 3/20, Train Loss: 0.7341, Validation Loss: 0.7543, Duration: 0:00:00.219999, Best Val Epoch: 1

20%|██ | 4/20 [00:00<00:03, 4.44it/s]

Epoch 4/20, Train Loss: 0.7315, Validation Loss: 0.7690, Duration: 0:00:00.221000, Best Val Epoch: 1

25%|██▌ | 5/20 [00:01<00:03, 4.48it/s]

Epoch 5/20, Train Loss: 0.7313, Validation Loss: 0.7462, Duration: 0:00:00.220001, Best Val Epoch: 1

30%|███ | 6/20 [00:01<00:03, 4.49it/s]

Epoch 6/20, Train Loss: 0.7278, Validation Loss: 0.7616, Duration: 0:00:00.219998, Best Val Epoch: 1

35%|███▌ | 7/20 [00:01<00:02, 4.50it/s]

Epoch 7/20, Train Loss: 0.7234, Validation Loss: 0.7474, Duration: 0:00:00.219998, Best Val Epoch: 1

40%|████ | 8/20 [00:01<00:02, 4.50it/s]

Epoch 8/20, Train Loss: 0.7122, Validation Loss: 0.7595, Duration: 0:00:00.221998, Best Val Epoch: 1

45%|████▌ | 9/20 [00:02<00:02, 4.51it/s]

Epoch 9/20, Train Loss: 0.7052, Validation Loss: 0.7645, Duration: 0:00:00.219998, Best Val Epoch: 1

50%|█████ | 10/20 [00:02<00:02, 4.50it/s]

Epoch 10/20, Train Loss: 0.7015, Validation Loss: 0.7911, Duration: 0:00:00.220999, Best Val Epoch: 1

55%|█████▌ | 11/20 [00:02<00:02, 4.50it/s]

Epoch 11/20, Train Loss: 0.6709, Validation Loss: 0.8030, Duration: 0:00:00.222001, Best Val Epoch: 1

60%|██████ | 12/20 [00:02<00:01, 4.52it/s]

Epoch 12/20, Train Loss: 0.6628, Validation Loss: 0.8893, Duration: 0:00:00.218000, Best Val Epoch: 1

65%|██████▌ | 13/20 [00:02<00:01, 4.51it/s]

Epoch 13/20, Train Loss: 0.6590, Validation Loss: 0.8283, Duration: 0:00:00.221000, Best Val Epoch: 1

70%|███████ | 14/20 [00:03<00:01, 4.52it/s]

Epoch 14/20, Train Loss: 0.6571, Validation Loss: 0.8847, Duration: 0:00:00.220000, Best Val Epoch: 1

75%|███████▌ | 15/20 [00:03<00:01, 4.53it/s]

Epoch 15/20, Train Loss: 0.6302, Validation Loss: 0.8736, Duration: 0:00:00.219999, Best Val Epoch: 1

80%|████████ | 16/20 [00:03<00:00, 4.53it/s]

Epoch 16/20, Train Loss: 0.6075, Validation Loss: 0.9612, Duration: 0:00:00.218997, Best Val Epoch: 1

85%|████████▌ | 17/20 [00:03<00:00, 4.54it/s]

Epoch 17/20, Train Loss: 0.5837, Validation Loss: 0.9919, Duration: 0:00:00.218999, Best Val Epoch: 1

90%|█████████ | 18/20 [00:04<00:00, 4.53it/s]

Epoch 18/20, Train Loss: 0.5810, Validation Loss: 1.0345, Duration: 0:00:00.219999, Best Val Epoch: 1

95%|█████████▌| 19/20 [00:04<00:00, 4.53it/s]

Epoch 19/20, Train Loss: 0.5540, Validation Loss: 1.0661, Duration: 0:00:00.221001, Best Val Epoch: 1

100%|██████████| 20/20 [00:04<00:00, 4.50it/s]

Epoch 20/20, Train Loss: 0.5298, Validation Loss: 1.1740, Duration: 0:00:00.220000, Best Val Epoch: 1

Full time to train: 0:00:04.447000. Average time per epoch: 0:00:00.222350

Evaluating DeepLOB on test set

Test acc: 0.4955. Test Loss: 0.7428. Time on test: 0:00:00.020000

precision recall f1-score support

0 0.4927 0.8182 0.6150 165

1 0.0000 0.0000 0.0000 4

2 0.5079 0.1905 0.2771 168

accuracy 0.4955 337

macro avg 0.3335 0.3362 0.2974 337

weighted avg 0.4944 0.4955 0.4392 337

C:\Users\bmanv\AppData\Local\Temp\ipykernel\_11012\3999389156.py:36: FutureWarning: You are using `torch.load` with `weights\_only=False` (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See <https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models> for more details). In a future release, the default value for `weights\_only` will be flipped to `True`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add\_safe\_globals`. We recommend you start setting `weights\_only=True` for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.

model.load\_state\_dict(torch.load(save\_path, map\_location=device))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

A chart of different colored squares

Description automatically generated

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Training B\_TABL...

0%| | 0/20 [00:00<?, ?it/s]

model saved at APPL\_short(scratch)\B\_TABL.pt

Epoch 1/20, Train Loss: 0.9129, Validation Loss: 0.8654, Duration: 0:00:00.039999, Best Val Epoch: 0

Epoch 2/20, Train Loss: 0.7827, Validation Loss: 0.9772, Duration: 0:00:00.037000, Best Val Epoch: 0

15%|█▌ | 3/20 [00:00<00:00, 27.03it/s]

Epoch 3/20, Train Loss: 0.7309, Validation Loss: 0.8715, Duration: 0:00:00.034000, Best Val Epoch: 0

model saved at APPL\_short(scratch)\B\_TABL.pt

Epoch 4/20, Train Loss: 0.7170, Validation Loss: 0.8644, Duration: 0:00:00.039004, Best Val Epoch: 3

Epoch 5/20, Train Loss: 0.7133, Validation Loss: 0.9251, Duration: 0:00:00.035997, Best Val Epoch: 3

Epoch 6/20, Train Loss: 0.6770, Validation Loss: 0.8726, Duration: 0:00:00.039999, Best Val Epoch: 3

30%|███ | 6/20 [00:00<00:00, 26.47it/s]

Epoch 7/20, Train Loss: 0.6835, Validation Loss: 0.9714, Duration: 0:00:00.034000, Best Val Epoch: 3

Epoch 8/20, Train Loss: 0.6829, Validation Loss: 1.1106, Duration: 0:00:00.035000, Best Val Epoch: 3

45%|████▌ | 9/20 [00:00<00:00, 27.50it/s]

Epoch 9/20, Train Loss: 0.6563, Validation Loss: 0.9640, Duration: 0:00:00.034001, Best Val Epoch: 3

Epoch 10/20, Train Loss: 0.6450, Validation Loss: 0.9402, Duration: 0:00:00.042002, Best Val Epoch: 3

Epoch 11/20, Train Loss: 0.6550, Validation Loss: 1.0092, Duration: 0:00:00.039998, Best Val Epoch: 3

60%|██████ | 12/20 [00:00<00:00, 26.55it/s]

Epoch 12/20, Train Loss: 0.6438, Validation Loss: 1.0366, Duration: 0:00:00.036000, Best Val Epoch: 3

Epoch 13/20, Train Loss: 0.6115, Validation Loss: 0.9717, Duration: 0:00:00.040000, Best Val Epoch: 3

Epoch 14/20, Train Loss: 0.6176, Validation Loss: 1.0962, Duration: 0:00:00.036000, Best Val Epoch: 3

75%|███████▌ | 15/20 [00:00<00:00, 26.89it/s]

Epoch 15/20, Train Loss: 0.6092, Validation Loss: 1.0021, Duration: 0:00:00.032999, Best Val Epoch: 3

Epoch 16/20, Train Loss: 0.6155, Validation Loss: 1.0926, Duration: 0:00:00.037000, Best Val Epoch: 3

Epoch 17/20, Train Loss: 0.5941, Validation Loss: 1.0495, Duration: 0:00:00.034002, Best Val Epoch: 3

90%|█████████ | 18/20 [00:00<00:00, 27.19it/s]

Epoch 18/20, Train Loss: 0.5823, Validation Loss: 1.0450, Duration: 0:00:00.036999, Best Val Epoch: 3

Epoch 19/20, Train Loss: 0.5708, Validation Loss: 1.0857, Duration: 0:00:00.033995, Best Val Epoch: 3

Epoch 20/20, Train Loss: 0.5581, Validation Loss: 1.1116, Duration: 0:00:00.032999, Best Val Epoch: 3

100%|██████████| 20/20 [00:00<00:00, 27.17it/s]

Full time to train: 0:00:00.736997. Average time per epoch: 0:00:00.036850

Evaluating B\_TABL on test set

Test acc: 0.5134. Test Loss: 0.8398. Time on test: 0:00:00.009002

precision recall f1-score support

0 0.5038 0.8121 0.6218 165

1 0.0000 0.0000 0.0000 4

2 0.5493 0.2321 0.3264 168

accuracy 0.5134 337

macro avg 0.3510 0.3481 0.3161 337

weighted avg 0.5205 0.5134 0.4671 337

C:\Users\bmanv\AppData\Local\Temp\ipykernel\_11012\3999389156.py:36: FutureWarning: You are using `torch.load` with `weights\_only=False` (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See <https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models> for more details). In a future release, the default value for `weights\_only` will be flipped to `True`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add\_safe\_globals`. We recommend you start setting `weights\_only=True` for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.

model.load\_state\_dict(torch.load(save\_path, map\_location=device))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

A chart of different colored squares

Description automatically generated

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Training C\_TABL...

0%| | 0/20 [00:00<?, ?it/s]

model saved at APPL\_short(scratch)\C\_TABL.pt

Epoch 1/20, Train Loss: 0.8427, Validation Loss: 0.8177, Duration: 0:00:00.048001, Best Val Epoch: 0

model saved at APPL\_short(scratch)\C\_TABL.pt

Epoch 2/20, Train Loss: 0.7416, Validation Loss: 0.8041, Duration: 0:00:00.045999, Best Val Epoch: 1

15%|█▌ | 3/20 [00:00<00:00, 20.98it/s]

model saved at APPL\_short(scratch)\C\_TABL.pt

Epoch 3/20, Train Loss: 0.7002, Validation Loss: 0.8037, Duration: 0:00:00.047001, Best Val Epoch: 2

Epoch 4/20, Train Loss: 0.6874, Validation Loss: 0.8399, Duration: 0:00:00.042996, Best Val Epoch: 2

Epoch 5/20, Train Loss: 0.6742, Validation Loss: 1.0313, Duration: 0:00:00.040000, Best Val Epoch: 2

30%|███ | 6/20 [00:00<00:00, 22.26it/s]

Epoch 6/20, Train Loss: 0.6723, Validation Loss: 0.9279, Duration: 0:00:00.045000, Best Val Epoch: 2

Epoch 7/20, Train Loss: 0.6591, Validation Loss: 0.8839, Duration: 0:00:00.039000, Best Val Epoch: 2

Epoch 8/20, Train Loss: 0.6394, Validation Loss: 0.9871, Duration: 0:00:00.037000, Best Val Epoch: 2

45%|████▌ | 9/20 [00:00<00:00, 23.69it/s]

Epoch 9/20, Train Loss: 0.6376, Validation Loss: 0.9533, Duration: 0:00:00.040000, Best Val Epoch: 2

Epoch 10/20, Train Loss: 0.6149, Validation Loss: 1.0085, Duration: 0:00:00.042001, Best Val Epoch: 2

60%|██████ | 12/20 [00:00<00:00, 23.74it/s]

Epoch 11/20, Train Loss: 0.5929, Validation Loss: 0.9645, Duration: 0:00:00.041997, Best Val Epoch: 2

Epoch 12/20, Train Loss: 0.5996, Validation Loss: 1.1941, Duration: 0:00:00.041000, Best Val Epoch: 2

Epoch 13/20, Train Loss: 0.6016, Validation Loss: 0.9345, Duration: 0:00:00.043000, Best Val Epoch: 2

Epoch 14/20, Train Loss: 0.5600, Validation Loss: 1.2100, Duration: 0:00:00.039000, Best Val Epoch: 2

75%|███████▌ | 15/20 [00:00<00:00, 23.83it/s]

Epoch 15/20, Train Loss: 0.5664, Validation Loss: 1.0302, Duration: 0:00:00.042002, Best Val Epoch: 2

90%|█████████ | 18/20 [00:00<00:00, 23.95it/s]

Epoch 16/20, Train Loss: 0.5808, Validation Loss: 1.2038, Duration: 0:00:00.039999, Best Val Epoch: 2

Epoch 17/20, Train Loss: 0.5697, Validation Loss: 0.9672, Duration: 0:00:00.041998, Best Val Epoch: 2

Epoch 18/20, Train Loss: 0.5578, Validation Loss: 1.0333, Duration: 0:00:00.040000, Best Val Epoch: 2

Epoch 19/20, Train Loss: 0.5207, Validation Loss: 1.0369, Duration: 0:00:00.042997, Best Val Epoch: 2

Epoch 20/20, Train Loss: 0.5018, Validation Loss: 1.1640, Duration: 0:00:00.039001, Best Val Epoch: 2

100%|██████████| 20/20 [00:00<00:00, 23.61it/s]

Full time to train: 0:00:00.849002. Average time per epoch: 0:00:00.042450

Evaluating C\_TABL on test set

Test acc: 0.5015. Test Loss: 0.7851. Time on test: 0:00:00.006001

precision recall f1-score support

0 0.5000 0.6606 0.5692 165

1 0.0000 0.0000 0.0000 4

2 0.5042 0.3571 0.4181 168

accuracy 0.5015 337

macro avg 0.3347 0.3392 0.3291 337

weighted avg 0.4962 0.5015 0.4871 337

C:\Users\bmanv\AppData\Local\Temp\ipykernel\_11012\3999389156.py:36: FutureWarning: You are using `torch.load` with `weights\_only=False` (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See <https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models> for more details). In a future release, the default value for `weights\_only` will be flipped to `True`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add\_safe\_globals`. We recommend you start setting `weights\_only=True` for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.

model.load\_state\_dict(torch.load(save\_path, map\_location=device))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

A chart with different colored squares

Description automatically generated

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Training DeepLOBSeq2Seq...

5%|▌ | 1/20 [00:00<00:01, 10.00it/s]

model saved at APPL\_short(scratch)\DeepLOBSeq2Seq.pt

Epoch 1/20, Train Loss: 0.9633, Validation Loss: 0.7677, Duration: 0:00:00.100001, Best Val Epoch: 0

model saved at APPL\_short(scratch)\DeepLOBSeq2Seq.pt

Epoch 2/20, Train Loss: 0.7521, Validation Loss: 0.7548, Duration: 0:00:00.097000, Best Val Epoch: 1

model saved at APPL\_short(scratch)\DeepLOBSeq2Seq.pt

15%|█▌ | 3/20 [00:00<00:01, 10.31it/s]

Epoch 3/20, Train Loss: 0.7422, Validation Loss: 0.7456, Duration: 0:00:00.094998, Best Val Epoch: 2

Epoch 4/20, Train Loss: 0.7474, Validation Loss: 0.7634, Duration: 0:00:00.101002, Best Val Epoch: 2

25%|██▌ | 5/20 [00:00<00:01, 9.99it/s]

Epoch 5/20, Train Loss: 0.7413, Validation Loss: 0.7490, Duration: 0:00:00.104999, Best Val Epoch: 2

Epoch 6/20, Train Loss: 0.7366, Validation Loss: 0.7473, Duration: 0:00:00.094000, Best Val Epoch: 2

model saved at APPL\_short(scratch)\DeepLOBSeq2Seq.pt

Epoch 7/20, Train Loss: 0.7336, Validation Loss: 0.7421, Duration: 0:00:00.092000, Best Val Epoch: 6

45%|████▌ | 9/20 [00:00<00:01, 10.28it/s]

Epoch 8/20, Train Loss: 0.7358, Validation Loss: 0.7514, Duration: 0:00:00.091000, Best Val Epoch: 6

Epoch 9/20, Train Loss: 0.7357, Validation Loss: 0.7613, Duration: 0:00:00.103002, Best Val Epoch: 6

55%|█████▌ | 11/20 [00:01<00:00, 10.24it/s]

Epoch 10/20, Train Loss: 0.7350, Validation Loss: 0.7529, Duration: 0:00:00.101999, Best Val Epoch: 6

Epoch 11/20, Train Loss: 0.7312, Validation Loss: 0.7448, Duration: 0:00:00.094004, Best Val Epoch: 6

Epoch 12/20, Train Loss: 0.7355, Validation Loss: 0.7479, Duration: 0:00:00.095998, Best Val Epoch: 6

65%|██████▌ | 13/20 [00:01<00:00, 10.35it/s]

Epoch 13/20, Train Loss: 0.7302, Validation Loss: 0.7464, Duration: 0:00:00.092000, Best Val Epoch: 6

Epoch 14/20, Train Loss: 0.7266, Validation Loss: 0.7589, Duration: 0:00:00.106000, Best Val Epoch: 6

model saved at APPL\_short(scratch)\DeepLOBSeq2Seq.pt

Epoch 15/20, Train Loss: 0.7292, Validation Loss: 0.7403, Duration: 0:00:00.092000, Best Val Epoch: 14

85%|████████▌ | 17/20 [00:01<00:00, 10.46it/s]

Epoch 16/20, Train Loss: 0.7321, Validation Loss: 0.7496, Duration: 0:00:00.089999, Best Val Epoch: 14

model saved at APPL\_short(scratch)\DeepLOBSeq2Seq.pt

Epoch 17/20, Train Loss: 0.7269, Validation Loss: 0.7321, Duration: 0:00:00.092001, Best Val Epoch: 16

Epoch 18/20, Train Loss: 0.7285, Validation Loss: 0.7624, Duration: 0:00:00.099999, Best Val Epoch: 16

100%|██████████| 20/20 [00:01<00:00, 10.33it/s]

Epoch 19/20, Train Loss: 0.7240, Validation Loss: 0.7583, Duration: 0:00:00.092001, Best Val Epoch: 16

Epoch 20/20, Train Loss: 0.7261, Validation Loss: 0.7358, Duration: 0:00:00.098000, Best Val Epoch: 16

Full time to train: 0:00:01.938000. Average time per epoch: 0:00:00.096900

Evaluating DeepLOBSeq2Seq on test set

Test acc: 0.5727. Test Loss: 0.7269. Time on test: 0:00:00.010000

precision recall f1-score support

0 0.5439 0.7879 0.6436 165

1 0.0000 0.0000 0.0000 4

2 0.6429 0.3750 0.4737 168

accuracy 0.5727 337

macro avg 0.3956 0.3876 0.3724 337

weighted avg 0.5868 0.5727 0.5512 337

C:\Users\bmanv\AppData\Local\Temp\ipykernel\_11012\3999389156.py:36: FutureWarning: You are using `torch.load` with `weights\_only=False` (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See <https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models> for more details). In a future release, the default value for `weights\_only` will be flipped to `True`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add\_safe\_globals`. We recommend you start setting `weights\_only=True` for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.

model.load\_state\_dict(torch.load(save\_path, map\_location=device))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

A chart of a graph

Description automatically generated with medium confidence

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Training DeepLOBAttention...

10%|█ | 2/20 [00:00<00:01, 11.11it/s]

model saved at APPL\_short(scratch)\DeepLOBAttention.pt

Epoch 1/20, Train Loss: 0.9143, Validation Loss: 0.8144, Duration: 0:00:00.094001, Best Val Epoch: 0

model saved at APPL\_short(scratch)\DeepLOBAttention.pt

Epoch 2/20, Train Loss: 0.7588, Validation Loss: 0.7510, Duration: 0:00:00.086000, Best Val Epoch: 1

Epoch 3/20, Train Loss: 0.7376, Validation Loss: 0.8048, Duration: 0:00:00.084999, Best Val Epoch: 1

20%|██ | 4/20 [00:00<00:01, 11.76it/s]

model saved at APPL\_short(scratch)\DeepLOBAttention.pt

Epoch 4/20, Train Loss: 0.7383, Validation Loss: 0.7453, Duration: 0:00:00.076002, Best Val Epoch: 3

Epoch 5/20, Train Loss: 0.7362, Validation Loss: 0.7682, Duration: 0:00:00.084998, Best Val Epoch: 3

30%|███ | 6/20 [00:00<00:01, 11.55it/s]

Epoch 6/20, Train Loss: 0.7350, Validation Loss: 0.7514, Duration: 0:00:00.092001, Best Val Epoch: 3

40%|████ | 8/20 [00:00<00:01, 11.71it/s]

Epoch 7/20, Train Loss: 0.7336, Validation Loss: 0.7540, Duration: 0:00:00.089000, Best Val Epoch: 3

Epoch 8/20, Train Loss: 0.7324, Validation Loss: 0.7606, Duration: 0:00:00.077000, Best Val Epoch: 3

Epoch 9/20, Train Loss: 0.7319, Validation Loss: 0.7714, Duration: 0:00:00.089999, Best Val Epoch: 3

50%|█████ | 10/20 [00:00<00:00, 11.49it/s]

Epoch 10/20, Train Loss: 0.7305, Validation Loss: 0.7527, Duration: 0:00:00.088002, Best Val Epoch: 3

Epoch 11/20, Train Loss: 0.7313, Validation Loss: 0.7545, Duration: 0:00:00.080999, Best Val Epoch: 3

60%|██████ | 12/20 [00:01<00:00, 11.65it/s]

Epoch 12/20, Train Loss: 0.7271, Validation Loss: 0.7620, Duration: 0:00:00.086002, Best Val Epoch: 3

70%|███████ | 14/20 [00:01<00:00, 11.69it/s]

Epoch 13/20, Train Loss: 0.7242, Validation Loss: 0.7732, Duration: 0:00:00.082000, Best Val Epoch: 3

Epoch 14/20, Train Loss: 0.7271, Validation Loss: 0.7671, Duration: 0:00:00.086999, Best Val Epoch: 3

Epoch 15/20, Train Loss: 0.7233, Validation Loss: 0.7759, Duration: 0:00:00.085999, Best Val Epoch: 3

80%|████████ | 16/20 [00:01<00:00, 11.63it/s]

Epoch 16/20, Train Loss: 0.7210, Validation Loss: 0.7823, Duration: 0:00:00.086003, Best Val Epoch: 3

Epoch 17/20, Train Loss: 0.7246, Validation Loss: 0.7773, Duration: 0:00:00.077999, Best Val Epoch: 3

90%|█████████ | 18/20 [00:01<00:00, 11.63it/s]

Epoch 18/20, Train Loss: 0.7194, Validation Loss: 0.7813, Duration: 0:00:00.094001, Best Val Epoch: 3

Epoch 19/20, Train Loss: 0.7200, Validation Loss: 0.7873, Duration: 0:00:00.091999, Best Val Epoch: 3

Epoch 20/20, Train Loss: 0.7167, Validation Loss: 0.7990, Duration: 0:00:00.085000, Best Val Epoch: 3

100%|██████████| 20/20 [00:01<00:00, 11.56it/s]

Full time to train: 0:00:01.731001. Average time per epoch: 0:00:00.086550

Evaluating DeepLOBAttention on test set

Test acc: 0.5074. Test Loss: 0.7414. Time on test: 0:00:00.009000

precision recall f1-score support

0 0.6364 0.0424 0.0795 165

1 0.0000 0.0000 0.0000 4

2 0.5031 0.9762 0.6640 168

accuracy 0.5074 337

macro avg 0.3798 0.3395 0.2478 337

weighted avg 0.5624 0.5074 0.3699 337

C:\Users\bmanv\AppData\Local\Temp\ipykernel\_11012\3999389156.py:36: FutureWarning: You are using `torch.load` with `weights\_only=False` (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See <https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models> for more details). In a future release, the default value for `weights\_only` will be flipped to `True`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add\_safe\_globals`. We recommend you start setting `weights\_only=True` for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.

model.load\_state\_dict(torch.load(save\_path, map\_location=device))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

A chart with numbers and a few colored squares

Description automatically generated with medium confidence

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

print("Train pre-trained models from FI-2010 on AAPL data")

evaluate\_all(get\_models\_to\_test(True, "all\_models"), epochs, train\_loader\_short, val\_loader\_short, test\_loader\_short, train\_model=TRAIN\_APPL, save\_dir=save\_loc + "\_short(pretrained)")  
  
Train pre-trained models from FI-2010 on APPL data

Attempting to load from pretrained...

All models loaded successfully

Training AxialLOB...

5%|▌ | 1/20 [00:02<00:42, 2.23s/it]

model saved at APPL\_short(pretrained)\AxialLOB.pt

Epoch 1/20, Train Loss: 1.0489, Validation Loss: 0.9633, Duration: 0:00:02.227505, Best Val Epoch: 0

10%|█ | 2/20 [00:04<00:37, 2.06s/it]

Epoch 2/20, Train Loss: 0.9876, Validation Loss: 1.0235, Duration: 0:00:01.943227, Best Val Epoch: 0

15%|█▌ | 3/20 [00:06<00:34, 2.01s/it]

Epoch 3/20, Train Loss: 0.9636, Validation Loss: 1.0037, Duration: 0:00:01.952982, Best Val Epoch: 0

20%|██ | 4/20 [00:08<00:31, 1.99s/it]

Epoch 4/20, Train Loss: 0.9533, Validation Loss: 0.9746, Duration: 0:00:01.956000, Best Val Epoch: 0

Turning on additional grads for AxialLOB...

25%|██▌ | 5/20 [00:10<00:30, 2.01s/it]

Epoch 5/20, Train Loss: 0.9094, Validation Loss: 1.0336, Duration: 0:00:02.033998, Best Val Epoch: 0

30%|███ | 6/20 [00:12<00:28, 2.02s/it]

Epoch 6/20, Train Loss: 0.9025, Validation Loss: 1.0492, Duration: 0:00:02.032000, Best Val Epoch: 0

35%|███▌ | 7/20 [00:14<00:26, 2.02s/it]

Epoch 7/20, Train Loss: 0.8799, Validation Loss: 1.0197, Duration: 0:00:02.016000, Best Val Epoch: 0

40%|████ | 8/20 [00:16<00:24, 2.02s/it]

Epoch 8/20, Train Loss: 0.8557, Validation Loss: 1.0614, Duration: 0:00:02.016000, Best Val Epoch: 0

45%|████▌ | 9/20 [00:18<00:22, 2.02s/it]

Epoch 9/20, Train Loss: 0.8397, Validation Loss: 1.0261, Duration: 0:00:02.016997, Best Val Epoch: 0

50%|█████ | 10/20 [00:20<00:20, 2.02s/it]

Epoch 10/20, Train Loss: 0.8317, Validation Loss: 1.0355, Duration: 0:00:02.014040, Best Val Epoch: 0

55%|█████▌ | 11/20 [00:22<00:18, 2.02s/it]

Epoch 11/20, Train Loss: 0.8282, Validation Loss: 1.0266, Duration: 0:00:02.014999, Best Val Epoch: 0

60%|██████ | 12/20 [00:24<00:16, 2.02s/it]

Epoch 12/20, Train Loss: 0.8139, Validation Loss: 1.0518, Duration: 0:00:02.014000, Best Val Epoch: 0

65%|██████▌ | 13/20 [00:26<00:14, 2.02s/it]

Epoch 13/20, Train Loss: 0.7942, Validation Loss: 1.0575, Duration: 0:00:02.020997, Best Val Epoch: 0

70%|███████ | 14/20 [00:28<00:12, 2.02s/it]

Epoch 14/20, Train Loss: 0.7789, Validation Loss: 1.0923, Duration: 0:00:02.028999, Best Val Epoch: 0

75%|███████▌ | 15/20 [00:30<00:10, 2.02s/it]

Epoch 15/20, Train Loss: 0.7793, Validation Loss: 1.0852, Duration: 0:00:02.026002, Best Val Epoch: 0

80%|████████ | 16/20 [00:32<00:08, 2.02s/it]

Epoch 16/20, Train Loss: 0.7640, Validation Loss: 1.1094, Duration: 0:00:02.028999, Best Val Epoch: 0

85%|████████▌ | 17/20 [00:34<00:06, 2.03s/it]

Epoch 17/20, Train Loss: 0.7583, Validation Loss: 1.0984, Duration: 0:00:02.025000, Best Val Epoch: 0

90%|█████████ | 18/20 [00:36<00:04, 2.03s/it]

Epoch 18/20, Train Loss: 0.7340, Validation Loss: 1.1146, Duration: 0:00:02.035996, Best Val Epoch: 0

95%|█████████▌| 19/20 [00:38<00:02, 2.03s/it]

Epoch 19/20, Train Loss: 0.7269, Validation Loss: 1.0728, Duration: 0:00:02.038001, Best Val Epoch: 0

100%|██████████| 20/20 [00:40<00:00, 2.02s/it]

Epoch 20/20, Train Loss: 0.7295, Validation Loss: 1.1079, Duration: 0:00:02.039998, Best Val Epoch: 0

Full time to train: 0:00:40.500764. Average time per epoch: 0:00:02.025038

Evaluating AxialLOB on test set

Test acc: 0.4866. Test Loss: 1.0109. Time on test: 0:00:00.121000

precision recall f1-score support

0 0.4810 0.6121 0.5387 165

1 0.0000 0.0000 0.0000 4

2 0.4961 0.3750 0.4271 168

accuracy 0.4866 337

macro avg 0.3257 0.3290 0.3219 337

weighted avg 0.4828 0.4866 0.4767 337

C:\Users\bmanv\AppData\Local\Temp\ipykernel\_11012\3999389156.py:36: FutureWarning: You are using `torch.load` with `weights\_only=False` (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See <https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models> for more details). In a future release, the default value for `weights\_only` will be flipped to `True`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add\_safe\_globals`. We recommend you start setting `weights\_only=True` for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.

model.load\_state\_dict(torch.load(save\_path, map\_location=device))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

A chart of a graph

Description automatically generated with medium confidence

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Training DeepLOB...

5%|▌ | 1/20 [00:00<00:04, 4.24it/s]

model saved at APPL\_short(pretrained)\DeepLOB.pt

Epoch 1/20, Train Loss: 0.7949, Validation Loss: 0.7611, Duration: 0:00:00.235995, Best Val Epoch: 0

10%|█ | 2/20 [00:00<00:04, 4.25it/s]

model saved at APPL\_short(pretrained)\DeepLOB.pt

Epoch 2/20, Train Loss: 0.7352, Validation Loss: 0.7585, Duration: 0:00:00.234001, Best Val Epoch: 1

15%|█▌ | 3/20 [00:00<00:03, 4.34it/s]

Epoch 3/20, Train Loss: 0.7094, Validation Loss: 0.7792, Duration: 0:00:00.222999, Best Val Epoch: 1

20%|██ | 4/20 [00:00<00:03, 4.38it/s]

Epoch 4/20, Train Loss: 0.6778, Validation Loss: 0.8681, Duration: 0:00:00.224001, Best Val Epoch: 1

25%|██▌ | 5/20 [00:01<00:03, 4.42it/s]

Epoch 5/20, Train Loss: 0.6484, Validation Loss: 0.7874, Duration: 0:00:00.223000, Best Val Epoch: 1

30%|███ | 6/20 [00:01<00:03, 4.43it/s]

Epoch 6/20, Train Loss: 0.6269, Validation Loss: 0.8750, Duration: 0:00:00.222997, Best Val Epoch: 1

35%|███▌ | 7/20 [00:01<00:02, 4.45it/s]

Epoch 7/20, Train Loss: 0.5539, Validation Loss: 0.8818, Duration: 0:00:00.223000, Best Val Epoch: 1

40%|████ | 8/20 [00:01<00:02, 4.42it/s]

Epoch 8/20, Train Loss: 0.4895, Validation Loss: 0.8460, Duration: 0:00:00.226995, Best Val Epoch: 1

45%|████▌ | 9/20 [00:02<00:02, 4.42it/s]

Epoch 9/20, Train Loss: 0.4092, Validation Loss: 1.0999, Duration: 0:00:00.225001, Best Val Epoch: 1

50%|█████ | 10/20 [00:02<00:02, 4.43it/s]

Epoch 10/20, Train Loss: 0.3341, Validation Loss: 1.1511, Duration: 0:00:00.224999, Best Val Epoch: 1

55%|█████▌ | 11/20 [00:02<00:02, 4.42it/s]

Epoch 11/20, Train Loss: 0.2420, Validation Loss: 1.3689, Duration: 0:00:00.227999, Best Val Epoch: 1

60%|██████ | 12/20 [00:02<00:01, 4.42it/s]

Epoch 12/20, Train Loss: 0.1773, Validation Loss: 1.4908, Duration: 0:00:00.222998, Best Val Epoch: 1

65%|██████▌ | 13/20 [00:02<00:01, 4.43it/s]

Epoch 13/20, Train Loss: 0.1597, Validation Loss: 1.6264, Duration: 0:00:00.224000, Best Val Epoch: 1

70%|███████ | 14/20 [00:03<00:01, 4.44it/s]

Epoch 14/20, Train Loss: 0.1065, Validation Loss: 1.7030, Duration: 0:00:00.223999, Best Val Epoch: 1

75%|███████▌ | 15/20 [00:03<00:01, 4.43it/s]

Epoch 15/20, Train Loss: 0.0676, Validation Loss: 1.7868, Duration: 0:00:00.224999, Best Val Epoch: 1

80%|████████ | 16/20 [00:03<00:00, 4.43it/s]

Epoch 16/20, Train Loss: 0.0551, Validation Loss: 1.7930, Duration: 0:00:00.224997, Best Val Epoch: 1

85%|████████▌ | 17/20 [00:03<00:00, 4.43it/s]

Epoch 17/20, Train Loss: 0.0322, Validation Loss: 1.9274, Duration: 0:00:00.224998, Best Val Epoch: 1

90%|█████████ | 18/20 [00:04<00:00, 4.44it/s]

Epoch 18/20, Train Loss: 0.0173, Validation Loss: 2.1203, Duration: 0:00:00.221999, Best Val Epoch: 1

95%|█████████▌| 19/20 [00:04<00:00, 4.44it/s]

Epoch 19/20, Train Loss: 0.0116, Validation Loss: 2.2710, Duration: 0:00:00.226001, Best Val Epoch: 1

100%|██████████| 20/20 [00:04<00:00, 4.42it/s]

Epoch 20/20, Train Loss: 0.0136, Validation Loss: 2.2912, Duration: 0:00:00.224997, Best Val Epoch: 1

Full time to train: 0:00:04.529998. Average time per epoch: 0:00:00.226500

Evaluating DeepLOB on test set

Test acc: 0.5460. Test Loss: 0.7410. Time on test: 0:00:00.019999

precision recall f1-score support

0 0.5254 0.7515 0.6185 165

1 0.0000 0.0000 0.0000 4

2 0.5941 0.3571 0.4461 168

accuracy 0.5460 337

macro avg 0.3732 0.3696 0.3549 337

weighted avg 0.5534 0.5460 0.5252 337

C:\Users\bmanv\AppData\Local\Temp\ipykernel\_11012\3999389156.py:36: FutureWarning: You are using `torch.load` with `weights\_only=False` (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See <https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models> for more details). In a future release, the default value for `weights\_only` will be flipped to `True`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add\_safe\_globals`. We recommend you start setting `weights\_only=True` for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.

model.load\_state\_dict(torch.load(save\_path, map\_location=device))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

A chart of a graph

Description automatically generated with medium confidence

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Training B\_TABL...

0%| | 0/20 [00:00<?, ?it/s]

model saved at APPL\_short(pretrained)\B\_TABL.pt

Epoch 1/20, Train Loss: 1.2436, Validation Loss: 0.9004, Duration: 0:00:00.042001, Best Val Epoch: 0

15%|█▌ | 3/20 [00:00<00:00, 26.32it/s]

model saved at APPL\_short(pretrained)\B\_TABL.pt

Epoch 2/20, Train Loss: 0.7879, Validation Loss: 0.8565, Duration: 0:00:00.036999, Best Val Epoch: 1

Epoch 3/20, Train Loss: 0.6838, Validation Loss: 0.9387, Duration: 0:00:00.035000, Best Val Epoch: 1

Epoch 4/20, Train Loss: 0.6785, Validation Loss: 0.9020, Duration: 0:00:00.038002, Best Val Epoch: 1

Epoch 5/20, Train Loss: 0.6834, Validation Loss: 0.9116, Duration: 0:00:00.042999, Best Val Epoch: 1

30%|███ | 6/20 [00:00<00:00, 25.27it/s]

Epoch 6/20, Train Loss: 0.6553, Validation Loss: 1.0695, Duration: 0:00:00.039000, Best Val Epoch: 1

Epoch 7/20, Train Loss: 0.6185, Validation Loss: 1.0430, Duration: 0:00:00.037000, Best Val Epoch: 1

60%|██████ | 12/20 [00:00<00:00, 27.23it/s]

Epoch 8/20, Train Loss: 0.5913, Validation Loss: 1.1974, Duration: 0:00:00.033000, Best Val Epoch: 1

Epoch 9/20, Train Loss: 0.5854, Validation Loss: 1.0115, Duration: 0:00:00.035002, Best Val Epoch: 1

Epoch 10/20, Train Loss: 0.5760, Validation Loss: 1.0005, Duration: 0:00:00.035998, Best Val Epoch: 1

Epoch 11/20, Train Loss: 0.5666, Validation Loss: 1.0327, Duration: 0:00:00.035000, Best Val Epoch: 1

Epoch 12/20, Train Loss: 0.5480, Validation Loss: 1.1874, Duration: 0:00:00.035001, Best Val Epoch: 1

Epoch 13/20, Train Loss: 0.5489, Validation Loss: 1.0819, Duration: 0:00:00.037000, Best Val Epoch: 1

90%|█████████ | 18/20 [00:00<00:00, 27.31it/s]

Epoch 14/20, Train Loss: 0.5287, Validation Loss: 1.1841, Duration: 0:00:00.035999, Best Val Epoch: 1

Epoch 15/20, Train Loss: 0.5198, Validation Loss: 1.4096, Duration: 0:00:00.035000, Best Val Epoch: 1

Epoch 16/20, Train Loss: 0.5136, Validation Loss: 1.1238, Duration: 0:00:00.038001, Best Val Epoch: 1

Epoch 17/20, Train Loss: 0.5089, Validation Loss: 1.1764, Duration: 0:00:00.038001, Best Val Epoch: 1

Epoch 18/20, Train Loss: 0.4913, Validation Loss: 1.2267, Duration: 0:00:00.033998, Best Val Epoch: 1

100%|██████████| 20/20 [00:00<00:00, 27.06it/s]

Epoch 19/20, Train Loss: 0.5011, Validation Loss: 1.2029, Duration: 0:00:00.034002, Best Val Epoch: 1

Epoch 20/20, Train Loss: 0.4834, Validation Loss: 1.2734, Duration: 0:00:00.036999, Best Val Epoch: 1

Full time to train: 0:00:00.741000. Average time per epoch: 0:00:00.037050

Evaluating B\_TABL on test set

Test acc: 0.5074. Test Loss: 0.8365. Time on test: 0:00:00.005999

precision recall f1-score support

0 0.4977 0.6485 0.5632 165

1 0.0000 0.0000 0.0000 4

2 0.5246 0.3810 0.4414 168

accuracy 0.5074 337

macro avg 0.3408 0.3431 0.3348 337

weighted avg 0.5052 0.5074 0.4958 337

C:\Users\bmanv\AppData\Local\Temp\ipykernel\_11012\3999389156.py:36: FutureWarning: You are using `torch.load` with `weights\_only=False` (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See <https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models> for more details). In a future release, the default value for `weights\_only` will be flipped to `True`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add\_safe\_globals`. We recommend you start setting `weights\_only=True` for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.

model.load\_state\_dict(torch.load(save\_path, map\_location=device))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

A chart of a graph

Description automatically generated with medium confidence

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Training C\_TABL...

0%| | 0/20 [00:00<?, ?it/s]

model saved at APPL\_short(pretrained)\C\_TABL.pt

15%|█▌ | 3/20 [00:00<00:00, 21.28it/s]

Epoch 1/20, Train Loss: 0.9320, Validation Loss: 0.7797, Duration: 0:00:00.053998, Best Val Epoch: 0

model saved at APPL\_short(pretrained)\C\_TABL.pt

Epoch 2/20, Train Loss: 0.7391, Validation Loss: 0.7749, Duration: 0:00:00.044000, Best Val Epoch: 1

Epoch 3/20, Train Loss: 0.7164, Validation Loss: 0.8006, Duration: 0:00:00.043000, Best Val Epoch: 1

Epoch 4/20, Train Loss: 0.6885, Validation Loss: 0.8302, Duration: 0:00:00.043001, Best Val Epoch: 1

Epoch 5/20, Train Loss: 0.6769, Validation Loss: 0.7952, Duration: 0:00:00.042998, Best Val Epoch: 1

45%|████▌ | 9/20 [00:00<00:00, 22.96it/s]

Epoch 6/20, Train Loss: 0.6603, Validation Loss: 0.8020, Duration: 0:00:00.044001, Best Val Epoch: 1

Epoch 7/20, Train Loss: 0.6495, Validation Loss: 0.8019, Duration: 0:00:00.041000, Best Val Epoch: 1

Epoch 8/20, Train Loss: 0.6280, Validation Loss: 0.8427, Duration: 0:00:00.043000, Best Val Epoch: 1

Epoch 9/20, Train Loss: 0.6188, Validation Loss: 0.9464, Duration: 0:00:00.040999, Best Val Epoch: 1

Epoch 10/20, Train Loss: 0.6217, Validation Loss: 0.8002, Duration: 0:00:00.042999, Best Val Epoch: 1

Epoch 11/20, Train Loss: 0.6031, Validation Loss: 0.7988, Duration: 0:00:00.040001, Best Val Epoch: 1

60%|██████ | 12/20 [00:00<00:00, 23.15it/s]

Epoch 12/20, Train Loss: 0.5897, Validation Loss: 0.8181, Duration: 0:00:00.044001, Best Val Epoch: 1

Epoch 13/20, Train Loss: 0.5818, Validation Loss: 0.8399, Duration: 0:00:00.040000, Best Val Epoch: 1

Epoch 14/20, Train Loss: 0.5641, Validation Loss: 0.8245, Duration: 0:00:00.044003, Best Val Epoch: 1

75%|███████▌ | 15/20 [00:00<00:00, 22.87it/s]

Epoch 15/20, Train Loss: 0.5532, Validation Loss: 0.8479, Duration: 0:00:00.047997, Best Val Epoch: 1

Epoch 16/20, Train Loss: 0.5714, Validation Loss: 0.8394, Duration: 0:00:00.045999, Best Val Epoch: 1

90%|█████████ | 18/20 [00:00<00:00, 22.82it/s]

Epoch 17/20, Train Loss: 0.5587, Validation Loss: 0.8331, Duration: 0:00:00.043000, Best Val Epoch: 1

Epoch 18/20, Train Loss: 0.5514, Validation Loss: 1.0593, Duration: 0:00:00.041999, Best Val Epoch: 1

Epoch 19/20, Train Loss: 0.5400, Validation Loss: 0.8215, Duration: 0:00:00.047999, Best Val Epoch: 1

100%|██████████| 20/20 [00:00<00:00, 22.57it/s]

Epoch 20/20, Train Loss: 0.5280, Validation Loss: 0.8333, Duration: 0:00:00.045000, Best Val Epoch: 1

Full time to train: 0:00:00.886997. Average time per epoch: 0:00:00.044350

Evaluating C\_TABL on test set

Test acc: 0.5045. Test Loss: 0.7982. Time on test: 0:00:00.007000

precision recall f1-score support

0 0.5333 0.0970 0.1641 165

1 0.0000 0.0000 0.0000 4

2 0.5016 0.9167 0.6484 168

accuracy 0.5045 337

macro avg 0.3450 0.3379 0.2708 337

weighted avg 0.5112 0.5045 0.4036 337

C:\Users\bmanv\AppData\Local\Temp\ipykernel\_11012\3999389156.py:36: FutureWarning: You are using `torch.load` with `weights\_only=False` (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See <https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models> for more details). In a future release, the default value for `weights\_only` will be flipped to `True`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add\_safe\_globals`. We recommend you start setting `weights\_only=True` for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.

model.load\_state\_dict(torch.load(save\_path, map\_location=device))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

A chart of a number of colored squares

Description automatically generated with medium confidence

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Training DeepLOBSeq2Seq...

5%|▌ | 1/20 [00:00<00:01, 9.52it/s]

model saved at APPL\_short(pretrained)\DeepLOBSeq2Seq.pt

Epoch 1/20, Train Loss: 1.2168, Validation Loss: 0.9025, Duration: 0:00:00.105002, Best Val Epoch: 0

model saved at APPL\_short(pretrained)\DeepLOBSeq2Seq.pt

Epoch 2/20, Train Loss: 0.7933, Validation Loss: 0.7542, Duration: 0:00:00.097997, Best Val Epoch: 1

Epoch 3/20, Train Loss: 0.7321, Validation Loss: 0.7577, Duration: 0:00:00.094000, Best Val Epoch: 1

15%|█▌ | 3/20 [00:00<00:01, 10.17it/s]

Epoch 4/20, Train Loss: 0.7142, Validation Loss: 0.7712, Duration: 0:00:00.097000, Best Val Epoch: 1

model saved at APPL\_short(pretrained)\DeepLOBSeq2Seq.pt

Epoch 5/20, Train Loss: 0.7150, Validation Loss: 0.7467, Duration: 0:00:00.098000, Best Val Epoch: 4

35%|███▌ | 7/20 [00:00<00:01, 10.11it/s]

model saved at APPL\_short(pretrained)\DeepLOBSeq2Seq.pt

Epoch 6/20, Train Loss: 0.7016, Validation Loss: 0.7381, Duration: 0:00:00.101998, Best Val Epoch: 5

model saved at APPL\_short(pretrained)\DeepLOBSeq2Seq.pt

Epoch 7/20, Train Loss: 0.6962, Validation Loss: 0.7289, Duration: 0:00:00.097002, Best Val Epoch: 6

model saved at APPL\_short(pretrained)\DeepLOBSeq2Seq.pt

Epoch 8/20, Train Loss: 0.6943, Validation Loss: 0.7258, Duration: 0:00:00.098997, Best Val Epoch: 7

45%|████▌ | 9/20 [00:00<00:01, 10.14it/s]

model saved at APPL\_short(pretrained)\DeepLOBSeq2Seq.pt

Epoch 9/20, Train Loss: 0.6794, Validation Loss: 0.7106, Duration: 0:00:00.095999, Best Val Epoch: 8

Epoch 10/20, Train Loss: 0.6848, Validation Loss: 0.7465, Duration: 0:00:00.094999, Best Val Epoch: 8

Epoch 11/20, Train Loss: 0.6692, Validation Loss: 0.7237, Duration: 0:00:00.094999, Best Val Epoch: 8

65%|██████▌ | 13/20 [00:01<00:00, 10.40it/s]

Epoch 12/20, Train Loss: 0.6707, Validation Loss: 0.7720, Duration: 0:00:00.092000, Best Val Epoch: 8

Epoch 13/20, Train Loss: 0.6672, Validation Loss: 0.7482, Duration: 0:00:00.092998, Best Val Epoch: 8

Epoch 14/20, Train Loss: 0.6475, Validation Loss: 0.7286, Duration: 0:00:00.096000, Best Val Epoch: 8

75%|███████▌ | 15/20 [00:01<00:00, 10.40it/s]

Epoch 15/20, Train Loss: 0.6308, Validation Loss: 0.7897, Duration: 0:00:00.094999, Best Val Epoch: 8

Epoch 16/20, Train Loss: 0.6191, Validation Loss: 0.7612, Duration: 0:00:00.089997, Best Val Epoch: 8

Epoch 17/20, Train Loss: 0.6141, Validation Loss: 0.7355, Duration: 0:00:00.092001, Best Val Epoch: 8

100%|██████████| 20/20 [00:01<00:00, 10.43it/s]

Epoch 18/20, Train Loss: 0.6104, Validation Loss: 0.7428, Duration: 0:00:00.098999, Best Val Epoch: 8

Epoch 19/20, Train Loss: 0.5986, Validation Loss: 0.7858, Duration: 0:00:00.086997, Best Val Epoch: 8

Epoch 20/20, Train Loss: 0.5862, Validation Loss: 0.7387, Duration: 0:00:00.087999, Best Val Epoch: 8

Full time to train: 0:00:01.919994. Average time per epoch: 0:00:00.096000

Evaluating DeepLOBSeq2Seq on test set

Test acc: 0.5579. Test Loss: 0.7506. Time on test: 0:00:00.010000

precision recall f1-score support

0 0.5354 0.7333 0.6189 165

1 0.0000 0.0000 0.0000 4

2 0.6036 0.3988 0.4803 168

accuracy 0.5579 337

macro avg 0.3797 0.3774 0.3664 337

weighted avg 0.5630 0.5579 0.5425 337

C:\Users\bmanv\AppData\Local\Temp\ipykernel\_11012\3999389156.py:36: FutureWarning: You are using `torch.load` with `weights\_only=False` (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See <https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models> for more details). In a future release, the default value for `weights\_only` will be flipped to `True`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add\_safe\_globals`. We recommend you start setting `weights\_only=True` for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.

model.load\_state\_dict(torch.load(save\_path, map\_location=device))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

A chart of a graph

Description automatically generated with medium confidence

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Training DeepLOBAttention...

5%|▌ | 1/20 [00:00<00:01, 9.71it/s]

model saved at APPL\_short(pretrained)\DeepLOBAttention.pt

Epoch 1/20, Train Loss: 1.1241, Validation Loss: 0.9101, Duration: 0:00:00.102998, Best Val Epoch: 0

model saved at APPL\_short(pretrained)\DeepLOBAttention.pt

Epoch 2/20, Train Loss: 0.7661, Validation Loss: 0.8197, Duration: 0:00:00.086000, Best Val Epoch: 1

15%|█▌ | 3/20 [00:00<00:01, 11.58it/s]

model saved at APPL\_short(pretrained)\DeepLOBAttention.pt

Epoch 3/20, Train Loss: 0.7434, Validation Loss: 0.7789, Duration: 0:00:00.075001, Best Val Epoch: 2

25%|██▌ | 5/20 [00:00<00:01, 11.85it/s]

Epoch 4/20, Train Loss: 0.7254, Validation Loss: 0.7881, Duration: 0:00:00.078998, Best Val Epoch: 2

model saved at APPL\_short(pretrained)\DeepLOBAttention.pt

Epoch 5/20, Train Loss: 0.7106, Validation Loss: 0.7670, Duration: 0:00:00.086001, Best Val Epoch: 4

Epoch 6/20, Train Loss: 0.7174, Validation Loss: 0.7698, Duration: 0:00:00.079998, Best Val Epoch: 4

35%|███▌ | 7/20 [00:00<00:01, 12.06it/s]

model saved at APPL\_short(pretrained)\DeepLOBAttention.pt

Epoch 7/20, Train Loss: 0.7088, Validation Loss: 0.7547, Duration: 0:00:00.081001, Best Val Epoch: 6

Epoch 8/20, Train Loss: 0.6946, Validation Loss: 0.8258, Duration: 0:00:00.077998, Best Val Epoch: 6

45%|████▌ | 9/20 [00:00<00:00, 12.22it/s]

Epoch 9/20, Train Loss: 0.7033, Validation Loss: 0.7838, Duration: 0:00:00.079999, Best Val Epoch: 6

55%|█████▌ | 11/20 [00:00<00:00, 12.37it/s]

model saved at APPL\_short(pretrained)\DeepLOBAttention.pt

Epoch 10/20, Train Loss: 0.6961, Validation Loss: 0.7451, Duration: 0:00:00.080999, Best Val Epoch: 9

Epoch 11/20, Train Loss: 0.6893, Validation Loss: 0.8089, Duration: 0:00:00.077000, Best Val Epoch: 9

Epoch 12/20, Train Loss: 0.6868, Validation Loss: 0.8566, Duration: 0:00:00.083001, Best Val Epoch: 9

65%|██████▌ | 13/20 [00:01<00:00, 12.41it/s]

Epoch 13/20, Train Loss: 0.6749, Validation Loss: 0.7914, Duration: 0:00:00.077000, Best Val Epoch: 9

Epoch 14/20, Train Loss: 0.6704, Validation Loss: 0.8417, Duration: 0:00:00.077999, Best Val Epoch: 9

75%|███████▌ | 15/20 [00:01<00:00, 12.52it/s]

Epoch 15/20, Train Loss: 0.6648, Validation Loss: 0.8168, Duration: 0:00:00.077999, Best Val Epoch: 9

85%|████████▌ | 17/20 [00:01<00:00, 12.59it/s]

Epoch 16/20, Train Loss: 0.6546, Validation Loss: 0.8118, Duration: 0:00:00.081000, Best Val Epoch: 9

Epoch 17/20, Train Loss: 0.6442, Validation Loss: 0.7936, Duration: 0:00:00.076001, Best Val Epoch: 9

Epoch 18/20, Train Loss: 0.6538, Validation Loss: 0.8162, Duration: 0:00:00.086999, Best Val Epoch: 9

100%|██████████| 20/20 [00:01<00:00, 12.16it/s]

Epoch 19/20, Train Loss: 0.6430, Validation Loss: 0.8811, Duration: 0:00:00.084002, Best Val Epoch: 9

Epoch 20/20, Train Loss: 0.6407, Validation Loss: 0.8203, Duration: 0:00:00.089998, Best Val Epoch: 9

Full time to train: 0:00:01.647001. Average time per epoch: 0:00:00.082350

Evaluating DeepLOBAttention on test set

Test acc: 0.5401. Test Loss: 0.7605. Time on test: 0:00:00.016000

precision recall f1-score support

0 0.5214 0.7394 0.6115 165

1 0.0000 0.0000 0.0000 4

2 0.5825 0.3571 0.4428 168

accuracy 0.5401 337

macro avg 0.3680 0.3655 0.3514 337

weighted avg 0.5457 0.5401 0.5202 337

C:\Users\bmanv\AppData\Local\Temp\ipykernel\_11012\3999389156.py:36: FutureWarning: You are using `torch.load` with `weights\_only=False` (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See <https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models> for more details). In a future release, the default value for `weights\_only` will be flipped to `True`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add\_safe\_globals`. We recommend you start setting `weights\_only=True` for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.

model.load\_state\_dict(torch.load(save\_path, map\_location=device))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

C:\Users\bmanv\AppData\Roaming\Python\Python311\site-packages\sklearn\metrics\\_classification.py:1471: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, msg\_start, len(result))

A chart of a graph

Description automatically generated with medium confidence

# Run a simulation of models on long AAPL data

def evaluate\_strategy\_with\_apple(model: nn.Module, test\_loader: data.DataLoader, prices\_test: np.ndarray):

model.to(device)

model.eval()

all\_preds = []

with torch.no\_grad():

for inputs, \_ in test\_loader:

inputs = inputs.to(device, dtype=torch.float)

outputs = model(inputs)

preds = torch.argmax(outputs, dim=1)

all\_preds.append(preds.cpu().numpy())

preds = np.concatenate(all\_preds)

return simulate\_trading\_debug(preds, prices\_test)

# Evaluate the DeepLOB Attention model

print("RUNNING ON 1 DAY INTERVAL DATA\n\n")

for model\_name, model in get\_models\_to\_test(True, save\_loc + "\_long").items():

results = evaluate\_strategy\_with\_apple(model, test\_loader\_long, prices\_long)

print(f"Results for {model\_name}:\n{results}")

print("\n\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n\n")

print("\n\nRUNNING ON 1 MINUTE INTERVAL DATA\n\n")

for model\_name, model in get\_models\_to\_test(True, save\_loc + "\_short(pretrained)").items():

results = evaluate\_strategy\_with\_apple(model, test\_loader\_short, prices\_short)

print(f"Results for {model\_name}:\n{results}")

print("\n\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n\n")  
  
RUNNING ON 1 DAY INTERVAL DATA

Attempting to load from pretrained...

All models loaded successfully

C:\Users\bmanv\AppData\Local\Temp\ipykernel\_11012\438007077.py:8: FutureWarning: You are using `torch.load` with `weights\_only=False` (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See <https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models> for more details). In a future release, the default value for `weights\_only` will be flipped to `True`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add\_safe\_globals`. We recommend you start setting `weights\_only=True` for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.

models\_to\_test[model\_name].load\_state\_dict(torch.load(\_save\_path, map\_location=device))

Initial cash: 10000, transaction cost: 0.001

BUY at step 0: Bought 83.78 shares at 119.24, Cash: 0.00, Stock: 83.78

SELL at step 44: Sold 83.78 shares at 127.46, Cash: 10667.58

BUY at step 45: Bought 82.16 shares at 129.70, Cash: 0.00, Stock: 82.16

SELL at step 47: Sold 82.16 shares at 123.21, Cash: 10113.14

BUY at step 48: Bought 82.44 shares at 122.55, Cash: -0.00, Stock: 82.44

BUY at step 49: Bought -0.00 shares at 124.50, Cash: -0.00, Stock: 82.44

BUY at step 50: Bought -0.00 shares at 126.10, Cash: 0.00, Stock: 82.44

BUY at step 51: Bought 0.00 shares at 125.39, Cash: 0.00, Stock: 82.44

SELL at step 78: Sold 82.44 shares at 133.14, Cash: 10965.52

BUY at step 79: Bought 82.40 shares at 132.94, Cash: 0.00, Stock: 82.40

SELL at step 82: Sold 82.40 shares at 136.06, Cash: 11200.23

BUY at step 83: Bought 82.30 shares at 135.96, Cash: 0.00, Stock: 82.30

SELL at step 104: Sold 82.30 shares at 145.46, Cash: 11959.43

BUY at step 106: Bought 81.57 shares at 146.47, Cash: 0.00, Stock: 81.57

SELL at step 109: Sold 81.57 shares at 145.67, Cash: 11870.24

BUY at step 111: Bought 81.40 shares at 145.68, Cash: -0.00, Stock: 81.40

BUY at step 112: Bought -0.00 shares at 146.24, Cash: 0.00, Stock: 81.40

SELL at step 115: Sold 81.40 shares at 149.41, Cash: 12150.25

BUY at step 116: Bought 82.73 shares at 146.72, Cash: 0.00, Stock: 82.73

SELL at step 119: Sold 82.73 shares at 148.18, Cash: 12246.33

BUY at step 122: Bought 82.82 shares at 147.71, Cash: 0.00, Stock: 82.82

Final Portfolio Value: 12104.16, Remaining Cash: 0.00, Remaining Stock at price 225.45: 82.82. Initial price was 119.24250221252441

Amount earned: 2104.1637892677736

Results for AxialLOB:

12104.163789267774

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Initial cash: 10000, transaction cost: 0.001

BUY at step 0: Bought 83.78 shares at 119.24, Cash: 0.00, Stock: 83.78

SELL at step 119: Sold 83.78 shares at 148.18, Cash: 12401.74

BUY at step 120: Bought 82.95 shares at 149.36, Cash: -0.00, Stock: 82.95

BUY at step 121: Bought -0.00 shares at 148.12, Cash: 0.00, Stock: 82.95

Final Portfolio Value: 12122.16, Remaining Cash: 0.00, Remaining Stock at price 225.45: 82.95. Initial price was 119.24250221252441

Amount earned: 2122.1552693775666

Results for DeepLOB:

12122.155269377567

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Initial cash: 10000, transaction cost: 0.001

BUY at step 0: Bought 83.78 shares at 119.24, Cash: 0.00, Stock: 83.78

SELL at step 9: Sold 83.78 shares at 122.78, Cash: 10276.09

BUY at step 10: Bought 85.07 shares at 120.68, Cash: 0.00, Stock: 85.07

BUY at step 11: Bought 0.00 shares at 119.90, Cash: 0.00, Stock: 85.07

SELL at step 12: Sold 85.07 shares at 120.71, Cash: 10258.43

BUY at step 13: Bought 83.73 shares at 122.40, Cash: 0.00, Stock: 83.73

SELL at step 19: Sold 83.73 shares at 121.45, Cash: 10157.95

BUY at step 20: Bought 82.70 shares at 122.70, Cash: 0.00, Stock: 82.70

SELL at step 23: Sold 82.70 shares at 125.49, Cash: 10367.86

BUY at step 24: Bought 80.44 shares at 128.75, Cash: 0.00, Stock: 80.44

SELL at step 30: Sold 80.44 shares at 133.45, Cash: 10724.85

BUY at step 31: Bought 80.19 shares at 133.61, Cash: 0.00, Stock: 80.19

SELL at step 32: Sold 80.19 shares at 132.27, Cash: 10596.77

BUY at step 33: Bought 80.44 shares at 131.61, Cash: 0.00, Stock: 80.44

BUY at step 34: Bought 0.00 shares at 131.75, Cash: 0.00, Stock: 80.44

BUY at step 35: Bought 0.00 shares at 132.53, Cash: 0.00, Stock: 80.44

SELL at step 37: Sold 80.44 shares at 134.27, Cash: 10789.85

BUY at step 39: Bought 81.03 shares at 133.03, Cash: -0.00, Stock: 81.03

BUY at step 40: Bought -0.00 shares at 131.38, Cash: -0.00, Stock: 81.03

SELL at step 41: Sold 81.03 shares at 132.11, Cash: 10694.03

BUY at step 42: Bought 83.92 shares at 127.30, Cash: 0.00, Stock: 83.92

SELL at step 47: Sold 83.92 shares at 123.21, Cash: 10329.65

BUY at step 50: Bought 81.83 shares at 126.10, Cash: -0.00, Stock: 81.83

BUY at step 51: Bought -0.00 shares at 125.39, Cash: 0.00, Stock: 81.83

SELL at step 52: Sold 81.83 shares at 125.06, Cash: 10223.28

BUY at step 53: Bought 82.95 shares at 123.12, Cash: 0.00, Stock: 82.95

SELL at step 59: Sold 82.95 shares at 125.40, Cash: 10392.01

BUY at step 61: Bought 83.64 shares at 124.12, Cash: 0.00, Stock: 83.64

SELL at step 65: Sold 83.64 shares at 125.02, Cash: 10446.45

BUY at step 66: Bought 82.50 shares at 126.49, Cash: 0.00, Stock: 82.50

SELL at step 67: Sold 82.50 shares at 126.67, Cash: 10440.62

BUY at step 68: Bought 82.63 shares at 126.22, Cash: 0.00, Stock: 82.63

SELL at step 71: Sold 82.63 shares at 129.54, Cash: 10693.83

BUY at step 72: Bought 82.97 shares at 128.76, Cash: 0.00, Stock: 82.97

SELL at step 77: Sold 82.97 shares at 133.37, Cash: 11053.93

BUY at step 79: Bought 83.06 shares at 132.94, Cash: 0.00, Stock: 83.06

SELL at step 81: Sold 83.06 shares at 134.62, Cash: 11170.63

BUY at step 82: Bought 82.02 shares at 136.06, Cash: -0.00, Stock: 82.02

BUY at step 83: Bought -0.00 shares at 135.96, Cash: 0.00, Stock: 82.02

SELL at step 86: Sold 82.02 shares at 142.94, Cash: 11711.72

BUY at step 87: Bought 82.92 shares at 141.09, Cash: 0.00, Stock: 82.92

SELL at step 93: Sold 82.92 shares at 146.37, Cash: 12125.00

BUY at step 95: Bought 84.42 shares at 143.48, Cash: -0.00, Stock: 84.42

BUY at step 96: Bought -0.00 shares at 144.82, Cash: -0.00, Stock: 84.42

BUY at step 97: Bought -0.00 shares at 146.11, Cash: 0.00, Stock: 84.42

SELL at step 101: Sold 84.42 shares at 143.09, Cash: 12068.41

BUY at step 102: Bought 83.25 shares at 144.83, Cash: 0.00, Stock: 83.25

BUY at step 103: Bought 0.00 shares at 144.39, Cash: 0.00, Stock: 83.25

SELL at step 107: Sold 83.25 shares at 146.38, Cash: 12173.40

BUY at step 109: Bought 83.49 shares at 145.67, Cash: 0.00, Stock: 83.49

BUY at step 110: Bought 0.00 shares at 145.60, Cash: 0.00, Stock: 83.49

SELL at step 113: Sold 83.49 shares at 148.42, Cash: 12378.34

BUY at step 116: Bought 84.28 shares at 146.72, Cash: 0.00, Stock: 84.28

SELL at step 119: Sold 84.28 shares at 148.18, Cash: 12476.22

BUY at step 122: Bought 84.38 shares at 147.71, Cash: 0.00, Stock: 84.38

BUY at step 123: Bought 0.00 shares at 147.07, Cash: 0.00, Stock: 84.38

SELL at step 126: Sold 84.38 shares at 152.67, Cash: 12869.29

BUY at step 129: Bought 83.08 shares at 154.75, Cash: -0.00, Stock: 83.08

BUY at step 130: Bought -0.00 shares at 154.36, Cash: 0.00, Stock: 83.08

BUY at step 131: Bought 0.00 shares at 154.22, Cash: 0.00, Stock: 83.08

SELL at step 132: Sold 83.08 shares at 149.55, Cash: 12411.89

BUY at step 136: Bought 84.10 shares at 147.44, Cash: 0.00, Stock: 84.10

Final Portfolio Value: 12290.48, Remaining Cash: 0.00, Remaining Stock at price 225.45: 84.10. Initial price was 119.24250221252441

Amount earned: 2290.475505072651

Results for B\_TABL:

12290.47550507265

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Initial cash: 10000, transaction cost: 0.001

BUY at step 0: Bought 83.78 shares at 119.24, Cash: 0.00, Stock: 83.78

SELL at step 10: Sold 83.78 shares at 120.68, Cash: 10100.12

BUY at step 13: Bought 82.43 shares at 122.40, Cash: 0.00, Stock: 82.43

SELL at step 18: Sold 82.43 shares at 119.05, Cash: 9804.07

BUY at step 19: Bought 80.65 shares at 121.45, Cash: 0.00, Stock: 80.65

SELL at step 23: Sold 80.65 shares at 125.49, Cash: 10110.07

BUY at step 24: Bought 78.44 shares at 128.75, Cash: -0.00, Stock: 78.44

BUY at step 25: Bought -0.00 shares at 129.92, Cash: 0.00, Stock: 78.44

SELL at step 32: Sold 78.44 shares at 132.27, Cash: 10365.81

BUY at step 33: Bought 78.69 shares at 131.61, Cash: 0.00, Stock: 78.69

BUY at step 34: Bought 0.00 shares at 131.75, Cash: 0.00, Stock: 78.69

BUY at step 35: Bought 0.00 shares at 132.53, Cash: 0.00, Stock: 78.69

SELL at step 51: Sold 78.69 shares at 125.39, Cash: 9856.46

BUY at step 53: Bought 79.98 shares at 123.12, Cash: 0.00, Stock: 79.98

SELL at step 61: Sold 79.98 shares at 124.12, Cash: 9916.57

BUY at step 63: Bought 80.32 shares at 123.34, Cash: 0.00, Stock: 80.32

SELL at step 64: Sold 80.32 shares at 124.14, Cash: 9960.44

BUY at step 67: Bought 78.55 shares at 126.67, Cash: 0.00, Stock: 78.55

SELL at step 77: Sold 78.55 shares at 133.37, Cash: 10465.73

BUY at step 82: Bought 76.84 shares at 136.06, Cash: -0.00, Stock: 76.84

BUY at step 83: Bought -0.00 shares at 135.96, Cash: 0.00, Stock: 76.84

SELL at step 91: Sold 76.84 shares at 147.92, Cash: 11354.77

BUY at step 95: Bought 79.06 shares at 143.48, Cash: 0.00, Stock: 79.06

SELL at step 101: Sold 79.06 shares at 143.09, Cash: 11301.77

BUY at step 102: Bought 77.96 shares at 144.83, Cash: 0.00, Stock: 77.96

SELL at step 105: Sold 77.96 shares at 145.54, Cash: 11334.58

BUY at step 110: Bought 77.77 shares at 145.60, Cash: 0.00, Stock: 77.77

SELL at step 113: Sold 77.77 shares at 148.42, Cash: 11530.63

BUY at step 116: Bought 78.51 shares at 146.72, Cash: 0.00, Stock: 78.51

SELL at step 120: Sold 78.51 shares at 149.36, Cash: 11714.85

BUY at step 121: Bought 79.01 shares at 148.12, Cash: 0.00, Stock: 79.01

SELL at step 126: Sold 79.01 shares at 152.67, Cash: 12050.99

BUY at step 128: Bought 78.54 shares at 153.28, Cash: 0.00, Stock: 78.54

SELL at step 132: Sold 78.54 shares at 149.55, Cash: 11733.86

BUY at step 136: Bought 79.51 shares at 147.44, Cash: -0.00, Stock: 79.51

BUY at step 137: Bought -0.00 shares at 146.14, Cash: 0.00, Stock: 79.51

Final Portfolio Value: 11619.08, Remaining Cash: 0.00, Remaining Stock at price 225.45: 79.51. Initial price was 119.24250221252441

Amount earned: 1619.0771969544785

Results for C\_TABL:

11619.077196954478

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Initial cash: 10000, transaction cost: 0.001

BUY at step 0: Bought 83.78 shares at 119.24, Cash: 0.00, Stock: 83.78

Final Portfolio Value: 12243.66, Remaining Cash: 0.00, Remaining Stock at price 225.45: 83.78. Initial price was 119.24250221252441

Amount earned: 2243.6628635309644

Results for DeepLOBSeq2Seq:

12243.662863530964

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Initial cash: 10000, transaction cost: 0.001

BUY at step 0: Bought 83.78 shares at 119.24, Cash: 0.00, Stock: 83.78

Final Portfolio Value: 12243.66, Remaining Cash: 0.00, Remaining Stock at price 225.45: 83.78. Initial price was 119.24250221252441

Amount earned: 2243.6628635309644

Results for DeepLOBAttention:

12243.662863530964

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

RUNNING ON 1 MINUTE INTERVAL DATA

Attempting to load from pretrained...

All models loaded successfully

Initial cash: 10000, transaction cost: 0.001

BUY at step 0: Bought -5774.39 shares at -1.73, Cash: -0.00, Stock: -5774.39

BUY at step 1: Bought 0.00 shares at -1.61, Cash: -0.00, Stock: -5774.39

BUY at step 2: Bought 0.00 shares at -1.56, Cash: -0.00, Stock: -5774.39

SELL at step 3: Sold -5774.39 shares at -1.60, Cash: 9217.15

BUY at step 4: Bought -5875.74 shares at -1.57, Cash: -0.00, Stock: -5875.74

BUY at step 5: Bought 0.00 shares at -1.57, Cash: 0.00, Stock: -5875.74

SELL at step 19: Sold -5875.74 shares at -1.20, Cash: 7071.03

BUY at step 20: Bought -6802.45 shares at -1.04, Cash: -0.00, Stock: -6802.45

BUY at step 21: Bought 0.00 shares at -1.03, Cash: 0.00, Stock: -6802.45

SELL at step 32: Sold -6802.45 shares at -1.20, Cash: 8151.02

BUY at step 33: Bought -6663.14 shares at -1.22, Cash: 0.00, Stock: -6663.14

SELL at step 35: Sold -6663.14 shares at -1.22, Cash: 8093.67

BUY at step 39: Bought -6597.94 shares at -1.23, Cash: -0.00, Stock: -6597.94

SELL at step 40: Sold -6597.94 shares at -1.23, Cash: 8091.53

BUY at step 41: Bought -6372.95 shares at -1.27, Cash: 0.00, Stock: -6372.95

BUY at step 42: Bought -0.00 shares at -1.24, Cash: 0.00, Stock: -6372.95

SELL at step 43: Sold -6372.95 shares at -1.14, Cash: 7228.41

BUY at step 48: Bought -4914.27 shares at -1.47, Cash: -0.00, Stock: -4914.27

BUY at step 49: Bought 0.00 shares at -1.38, Cash: 0.00, Stock: -4914.27

SELL at step 52: Sold -4914.27 shares at -1.59, Cash: 7798.30

BUY at step 55: Bought -5203.81 shares at -1.50, Cash: 0.00, Stock: -5203.81

SELL at step 56: Sold -5203.81 shares at -1.55, Cash: 8051.18

BUY at step 60: Bought -5636.63 shares at -1.43, Cash: 0.00, Stock: -5636.63

BUY at step 61: Bought -0.00 shares at -1.34, Cash: 0.00, Stock: -5636.63

SELL at step 62: Sold -5636.63 shares at -1.36, Cash: 7676.08

BUY at step 65: Bought -6107.55 shares at -1.26, Cash: 0.00, Stock: -6107.55

SELL at step 66: Sold -6107.55 shares at -1.29, Cash: 7889.08

BUY at step 75: Bought -5175.23 shares at -1.52, Cash: 0.00, Stock: -5175.23

SELL at step 76: Sold -5175.23 shares at -1.53, Cash: 7915.27

BUY at step 85: Bought -4774.85 shares at -1.66, Cash: 0.00, Stock: -4774.85

SELL at step 86: Sold -4774.85 shares at -1.59, Cash: 7560.63

BUY at step 112: Bought -5910.78 shares at -1.28, Cash: 0.00, Stock: -5910.78

SELL at step 113: Sold -5910.78 shares at -1.32, Cash: 7808.55

BUY at step 140: Bought -5144.89 shares at -1.52, Cash: 0.00, Stock: -5144.89

SELL at step 141: Sold -5144.89 shares at -1.49, Cash: 7639.99

BUY at step 151: Bought -5137.51 shares at -1.49, Cash: 0.00, Stock: -5137.51

SELL at step 153: Sold -5137.51 shares at -1.56, Cash: 8004.30

BUY at step 157: Bought -4941.42 shares at -1.62, Cash: -0.00, Stock: -4941.42

SELL at step 158: Sold -4941.42 shares at -1.61, Cash: 7942.16

BUY at step 167: Bought -4710.05 shares at -1.68, Cash: 0.00, Stock: -4710.05

SELL at step 168: Sold -4710.05 shares at -1.76, Cash: 8272.31

BUY at step 169: Bought -4725.50 shares at -1.75, Cash: -0.00, Stock: -4725.50

SELL at step 170: Sold -4725.50 shares at -1.70, Cash: 8020.48

BUY at step 181: Bought -4424.91 shares at -1.81, Cash: -0.00, Stock: -4424.91

SELL at step 182: Sold -4424.91 shares at -1.85, Cash: 8164.14

BUY at step 185: Bought -4385.76 shares at -1.86, Cash: 0.00, Stock: -4385.76

SELL at step 188: Sold -4385.76 shares at -1.77, Cash: 7747.41

BUY at step 199: Bought -4277.27 shares at -1.81, Cash: -0.00, Stock: -4277.27

BUY at step 200: Bought 0.00 shares at -1.80, Cash: 0.00, Stock: -4277.27

SELL at step 202: Sold -4277.27 shares at -1.81, Cash: 7739.20

BUY at step 204: Bought -4164.15 shares at -1.86, Cash: 0.00, Stock: -4164.15

SELL at step 206: Sold -4164.15 shares at -1.78, Cash: 7393.62

BUY at step 209: Bought -4124.57 shares at -1.79, Cash: 0.00, Stock: -4124.57

SELL at step 210: Sold -4124.57 shares at -1.84, Cash: 7571.47

BUY at step 214: Bought -4307.74 shares at -1.76, Cash: -0.00, Stock: -4307.74

BUY at step 215: Bought 0.00 shares at -1.77, Cash: 0.00, Stock: -4307.74

SELL at step 216: Sold -4307.74 shares at -1.78, Cash: 7651.94

BUY at step 217: Bought -4244.65 shares at -1.80, Cash: 0.00, Stock: -4244.65

SELL at step 218: Sold -4244.65 shares at -1.84, Cash: 7816.94

BUY at step 220: Bought -4024.62 shares at -1.94, Cash: 0.00, Stock: -4024.62

SELL at step 222: Sold -4024.62 shares at -1.98, Cash: 7951.99

BUY at step 223: Bought -4056.16 shares at -1.96, Cash: 0.00, Stock: -4056.16

SELL at step 228: Sold -4056.16 shares at -1.97, Cash: 7997.48

BUY at step 231: Bought -4240.14 shares at -1.88, Cash: 0.00, Stock: -4240.14

BUY at step 232: Bought -0.00 shares at -1.85, Cash: 0.00, Stock: -4240.14

SELL at step 237: Sold -4240.14 shares at -1.81, Cash: 7661.28

BUY at step 239: Bought -4016.48 shares at -1.91, Cash: 0.00, Stock: -4016.48

BUY at step 240: Bought -0.00 shares at -1.92, Cash: -0.00, Stock: -4016.48

BUY at step 241: Bought 0.00 shares at -1.94, Cash: 0.00, Stock: -4016.48

BUY at step 242: Bought -0.00 shares at -1.98, Cash: 0.00, Stock: -4016.48

BUY at step 243: Bought -0.00 shares at -1.97, Cash: 0.00, Stock: -4016.48

BUY at step 244: Bought -0.00 shares at -1.94, Cash: 0.00, Stock: -4016.48

SELL at step 246: Sold -4016.48 shares at -1.90, Cash: 7625.07

BUY at step 248: Bought -3934.33 shares at -1.94, Cash: 0.00, Stock: -3934.33

SELL at step 256: Sold -3934.33 shares at -1.90, Cash: 7484.54

BUY at step 257: Bought -4087.00 shares at -1.83, Cash: 0.00, Stock: -4087.00

SELL at step 258: Sold -4087.00 shares at -1.84, Cash: 7503.97

BUY at step 264: Bought -4236.41 shares at -1.77, Cash: 0.00, Stock: -4236.41

SELL at step 265: Sold -4236.41 shares at -1.77, Cash: 7485.42

BUY at step 266: Bought -4084.66 shares at -1.83, Cash: 0.00, Stock: -4084.66

SELL at step 268: Sold -4084.66 shares at -1.82, Cash: 7428.17

BUY at step 269: Bought -4083.75 shares at -1.82, Cash: 0.00, Stock: -4083.75

SELL at step 270: Sold -4083.75 shares at -1.83, Cash: 7471.78

BUY at step 271: Bought -4114.45 shares at -1.81, Cash: 0.00, Stock: -4114.45

SELL at step 272: Sold -4114.45 shares at -1.79, Cash: 7365.33

BUY at step 273: Bought -4083.59 shares at -1.80, Cash: 0.00, Stock: -4083.59

SELL at step 274: Sold -4083.59 shares at -1.77, Cash: 7225.82

BUY at step 275: Bought -4108.98 shares at -1.76, Cash: 0.00, Stock: -4108.98

SELL at step 276: Sold -4108.98 shares at -1.76, Cash: 7204.39

BUY at step 277: Bought -4150.01 shares at -1.73, Cash: 0.00, Stock: -4150.01

SELL at step 278: Sold -4150.01 shares at -1.73, Cash: 7161.82

BUY at step 285: Bought -4265.59 shares at -1.68, Cash: 0.00, Stock: -4265.59

SELL at step 287: Sold -4265.59 shares at -1.63, Cash: 6964.62

BUY at step 292: Bought -4113.81 shares at -1.69, Cash: 0.00, Stock: -4113.81

SELL at step 293: Sold -4113.81 shares at -1.72, Cash: 7048.67

BUY at step 301: Bought -4033.63 shares at -1.75, Cash: -0.00, Stock: -4033.63

BUY at step 302: Bought 0.00 shares at -1.71, Cash: -0.00, Stock: -4033.63

SELL at step 303: Sold -4033.63 shares at -1.69, Cash: 6825.60

BUY at step 306: Bought -3807.72 shares at -1.79, Cash: 0.00, Stock: -3807.72

SELL at step 307: Sold -3807.72 shares at -1.79, Cash: 6795.86

BUY at step 308: Bought -3836.64 shares at -1.77, Cash: 0.00, Stock: -3836.64

BUY at step 309: Bought -0.00 shares at -1.73, Cash: 0.00, Stock: -3836.64

SELL at step 313: Sold -3836.64 shares at -1.59, Cash: 6111.33

BUY at step 316: Bought -3958.84 shares at -1.54, Cash: 0.00, Stock: -3958.84

SELL at step 317: Sold -3958.84 shares at -1.51, Cash: 5976.04

BUY at step 318: Bought -3780.66 shares at -1.58, Cash: 0.00, Stock: -3780.66

SELL at step 325: Sold -3780.66 shares at -1.55, Cash: 5871.06

BUY at step 326: Bought -3795.96 shares at -1.55, Cash: 0.00, Stock: -3795.96

SELL at step 328: Sold -3795.96 shares at -1.56, Cash: 5904.47

BUY at step 329: Bought -3805.01 shares at -1.55, Cash: 0.00, Stock: -3805.01

SELL at step 330: Sold -3805.01 shares at -1.53, Cash: 5826.44

BUY at step 334: Bought -3662.36 shares at -1.59, Cash: 0.00, Stock: -3662.36

BUY at step 335: Bought -0.00 shares at -1.58, Cash: 0.00, Stock: -3662.36

BUY at step 336: Bought -0.00 shares at -1.60, Cash: 0.00, Stock: -3662.36

Final Portfolio Value: 5862.37, Remaining Cash: 0.00, Remaining Stock at price 0.45: -3662.36. Initial price was -1.7300534069260411

Amount earned: -4137.625356460065

Results for AxialLOB:

5862.374643539935

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Initial cash: 10000, transaction cost: 0.001

BUY at step 0: Bought -5774.39 shares at -1.73, Cash: -0.00, Stock: -5774.39

BUY at step 1: Bought 0.00 shares at -1.61, Cash: -0.00, Stock: -5774.39

BUY at step 2: Bought 0.00 shares at -1.56, Cash: -0.00, Stock: -5774.39

BUY at step 3: Bought 0.00 shares at -1.60, Cash: 0.00, Stock: -5774.39

SELL at step 4: Sold -5774.39 shares at -1.57, Cash: 9040.07

BUY at step 6: Bought -5870.77 shares at -1.54, Cash: 0.00, Stock: -5870.77

SELL at step 15: Sold -5870.77 shares at -1.42, Cash: 8334.04

BUY at step 20: Bought -8017.49 shares at -1.04, Cash: 0.00, Stock: -8017.49

SELL at step 22: Sold -8017.49 shares at -1.04, Cash: 8292.08

BUY at step 25: Bought -7970.57 shares at -1.04, Cash: 0.00, Stock: -7970.57

SELL at step 33: Sold -7970.57 shares at -1.22, Cash: 9730.93

BUY at step 35: Bought -7995.01 shares at -1.22, Cash: 0.00, Stock: -7995.01

SELL at step 36: Sold -7995.01 shares at -1.21, Cash: 9638.51

BUY at step 38: Bought -8544.81 shares at -1.13, Cash: 0.00, Stock: -8544.81

SELL at step 39: Sold -8544.81 shares at -1.23, Cash: 10460.94

BUY at step 51: Bought -6881.00 shares at -1.52, Cash: 0.00, Stock: -6881.00

SELL at step 53: Sold -6881.00 shares at -1.61, Cash: 11058.51

BUY at step 75: Bought -7254.37 shares at -1.52, Cash: 0.00, Stock: -7254.37

SELL at step 80: Sold -7254.37 shares at -1.74, Cash: 12583.76

BUY at step 81: Bought -7295.32 shares at -1.72, Cash: 0.00, Stock: -7295.32

SELL at step 85: Sold -7295.32 shares at -1.66, Cash: 12069.28

BUY at step 90: Bought -7907.92 shares at -1.52, Cash: 0.00, Stock: -7907.92

SELL at step 94: Sold -7907.92 shares at -1.48, Cash: 11716.19

BUY at step 98: Bought -8274.81 shares at -1.41, Cash: -0.00, Stock: -8274.81

BUY at step 99: Bought 0.00 shares at -1.41, Cash: 0.00, Stock: -8274.81

SELL at step 100: Sold -8274.81 shares at -1.38, Cash: 11444.60

BUY at step 105: Bought -7821.25 shares at -1.46, Cash: 0.00, Stock: -7821.25

SELL at step 106: Sold -7821.25 shares at -1.44, Cash: 11275.69

BUY at step 119: Bought -7922.33 shares at -1.42, Cash: -0.00, Stock: -7922.33

SELL at step 120: Sold -7922.33 shares at -1.48, Cash: 11720.73

BUY at step 126: Bought -8234.97 shares at -1.42, Cash: 0.00, Stock: -8234.97

SELL at step 129: Sold -8234.97 shares at -1.41, Cash: 11608.98

BUY at step 139: Bought -7576.72 shares at -1.53, Cash: 0.00, Stock: -7576.72

SELL at step 141: Sold -7576.72 shares at -1.49, Cash: 11251.18

BUY at step 169: Bought -6427.17 shares at -1.75, Cash: 0.00, Stock: -6427.17

SELL at step 173: Sold -6427.17 shares at -1.66, Cash: 10679.46

BUY at step 180: Bought -5729.19 shares at -1.86, Cash: 0.00, Stock: -5729.19

SELL at step 183: Sold -5729.19 shares at -1.86, Cash: 10648.51

BUY at step 184: Bought -5722.98 shares at -1.86, Cash: 0.00, Stock: -5722.98

BUY at step 185: Bought -0.00 shares at -1.86, Cash: -0.00, Stock: -5722.98

BUY at step 186: Bought 0.00 shares at -1.84, Cash: 0.00, Stock: -5722.98

SELL at step 190: Sold -5722.98 shares at -1.70, Cash: 9730.49

BUY at step 205: Bought -5377.16 shares at -1.81, Cash: 0.00, Stock: -5377.16

BUY at step 206: Bought -0.00 shares at -1.78, Cash: 0.00, Stock: -5377.16

SELL at step 210: Sold -5377.16 shares at -1.84, Cash: 9870.86

BUY at step 213: Bought -5493.49 shares at -1.80, Cash: 0.00, Stock: -5493.49

SELL at step 219: Sold -5493.49 shares at -1.91, Cash: 10506.55

BUY at step 220: Bought -5409.39 shares at -1.94, Cash: 0.00, Stock: -5409.39

SELL at step 222: Sold -5409.39 shares at -1.98, Cash: 10688.06

BUY at step 254: Bought -5464.38 shares at -1.95, Cash: 0.00, Stock: -5464.38

SELL at step 257: Sold -5464.38 shares at -1.83, Cash: 9986.94

BUY at step 274: Bought -5632.73 shares at -1.77, Cash: 0.00, Stock: -5632.73

SELL at step 276: Sold -5632.73 shares at -1.76, Cash: 9876.02

BUY at step 282: Bought -5783.95 shares at -1.71, Cash: 0.00, Stock: -5783.95

SELL at step 286: Sold -5783.95 shares at -1.66, Cash: 9620.53

BUY at step 291: Bought -5760.59 shares at -1.67, Cash: 0.00, Stock: -5760.59

SELL at step 293: Sold -5760.59 shares at -1.72, Cash: 9870.29

BUY at step 301: Bought -5648.30 shares at -1.75, Cash: 0.00, Stock: -5648.30

SELL at step 304: Sold -5648.30 shares at -1.72, Cash: 9714.95

BUY at step 306: Bought -5419.57 shares at -1.79, Cash: 0.00, Stock: -5419.57

SELL at step 309: Sold -5419.57 shares at -1.73, Cash: 9345.75

BUY at step 311: Bought -5691.94 shares at -1.64, Cash: -0.00, Stock: -5691.94

BUY at step 312: Bought 0.00 shares at -1.64, Cash: 0.00, Stock: -5691.94

SELL at step 316: Sold -5691.94 shares at -1.54, Cash: 8769.20

BUY at step 323: Bought -5482.74 shares at -1.60, Cash: -0.00, Stock: -5482.74

SELL at step 324: Sold -5482.74 shares at -1.58, Cash: 8653.92

BUY at step 325: Bought -5561.54 shares at -1.55, Cash: 0.00, Stock: -5561.54

SELL at step 326: Sold -5561.54 shares at -1.55, Cash: 8584.63

BUY at step 333: Bought -5544.36 shares at -1.55, Cash: 0.00, Stock: -5544.36

SELL at step 334: Sold -5544.36 shares at -1.59, Cash: 8802.89

BUY at step 336: Bought -5493.87 shares at -1.60, Cash: 0.00, Stock: -5493.87

Final Portfolio Value: 8794.10, Remaining Cash: 0.00, Remaining Stock at price 0.45: -5493.87. Initial price was -1.7300534069260411

Amount earned: -1205.904932410198

Results for DeepLOB:

8794.095067589802

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Initial cash: 10000, transaction cost: 0.001

BUY at step 7: Bought -6713.01 shares at -1.49, Cash: 0.00, Stock: -6713.01

SELL at step 9: Sold -6713.01 shares at -1.61, Cash: 10800.91

BUY at step 18: Bought -7795.27 shares at -1.38, Cash: -0.00, Stock: -7795.27

SELL at step 19: Sold -7795.27 shares at -1.20, Cash: 9381.06

BUY at step 24: Bought -9345.99 shares at -1.00, Cash: 0.00, Stock: -9345.99

BUY at step 25: Bought -0.00 shares at -1.04, Cash: 0.00, Stock: -9345.99

SELL at step 26: Sold -9345.99 shares at -1.14, Cash: 10600.42

BUY at step 27: Bought -10921.91 shares at -0.97, Cash: 0.00, Stock: -10921.91

SELL at step 29: Sold -10921.91 shares at -0.91, Cash: 9910.87

BUY at step 38: Bought -8786.26 shares at -1.13, Cash: 0.00, Stock: -8786.26

BUY at step 39: Bought -0.00 shares at -1.23, Cash: -0.00, Stock: -8786.26

BUY at step 40: Bought 0.00 shares at -1.23, Cash: 0.00, Stock: -8786.26

SELL at step 41: Sold -8786.26 shares at -1.27, Cash: 11133.35

BUY at step 43: Bought -9796.14 shares at -1.14, Cash: -0.00, Stock: -9796.14

BUY at step 44: Bought 0.00 shares at -1.32, Cash: 0.00, Stock: -9796.14

SELL at step 48: Sold -9796.14 shares at -1.47, Cash: 14380.38

BUY at step 50: Bought -10554.96 shares at -1.36, Cash: 0.00, Stock: -10554.96

SELL at step 52: Sold -10554.96 shares at -1.59, Cash: 16749.35

BUY at step 54: Bought -10919.50 shares at -1.53, Cash: 0.00, Stock: -10919.50

SELL at step 59: Sold -10919.50 shares at -1.43, Cash: 15640.14

BUY at step 110: Bought -12031.92 shares at -1.30, Cash: 0.00, Stock: -12031.92

SELL at step 112: Sold -12031.92 shares at -1.28, Cash: 15359.60

BUY at step 113: Bought -11603.41 shares at -1.32, Cash: 0.00, Stock: -11603.41

SELL at step 115: Sold -11603.41 shares at -1.26, Cash: 14657.04

BUY at step 116: Bought -10661.54 shares at -1.37, Cash: -0.00, Stock: -10661.54

BUY at step 117: Bought 0.00 shares at -1.40, Cash: -0.00, Stock: -10661.54

BUY at step 118: Bought 0.00 shares at -1.44, Cash: -0.00, Stock: -10661.54

SELL at step 119: Sold -10661.54 shares at -1.42, Cash: 15144.03

BUY at step 151: Bought -10183.60 shares at -1.49, Cash: 0.00, Stock: -10183.60

BUY at step 152: Bought -0.00 shares at -1.54, Cash: -0.00, Stock: -10183.60

BUY at step 153: Bought 0.00 shares at -1.56, Cash: 0.00, Stock: -10183.60

SELL at step 162: Sold -10183.60 shares at -1.66, Cash: 16929.87

BUY at step 163: Bought -10565.19 shares at -1.60, Cash: 0.00, Stock: -10565.19

SELL at step 165: Sold -10565.19 shares at -1.59, Cash: 16774.69

BUY at step 167: Bought -9948.14 shares at -1.68, Cash: 0.00, Stock: -9948.14

SELL at step 168: Sold -9948.14 shares at -1.76, Cash: 17472.01

BUY at step 187: Bought -9826.16 shares at -1.78, Cash: 0.00, Stock: -9826.16

BUY at step 188: Bought -0.00 shares at -1.77, Cash: 0.00, Stock: -9826.16

SELL at step 189: Sold -9826.16 shares at -1.74, Cash: 17126.63

BUY at step 190: Bought -10052.88 shares at -1.70, Cash: 0.00, Stock: -10052.88

SELL at step 206: Sold -10052.88 shares at -1.78, Cash: 17849.29

BUY at step 231: Bought -9463.42 shares at -1.88, Cash: 0.00, Stock: -9463.42

SELL at step 234: Sold -9463.42 shares at -1.89, Cash: 17838.10

BUY at step 236: Bought -9866.69 shares at -1.81, Cash: 0.00, Stock: -9866.69

SELL at step 245: Sold -9866.69 shares at -1.93, Cash: 18996.33

BUY at step 267: Bought -10227.63 shares at -1.86, Cash: 0.00, Stock: -10227.63

SELL at step 285: Sold -10227.63 shares at -1.68, Cash: 17137.60

BUY at step 286: Bought -10282.70 shares at -1.66, Cash: 0.00, Stock: -10282.70

BUY at step 287: Bought -0.00 shares at -1.63, Cash: 0.00, Stock: -10282.70

BUY at step 288: Bought -0.00 shares at -1.63, Cash: -0.00, Stock: -10282.70

BUY at step 289: Bought 0.00 shares at -1.59, Cash: 0.00, Stock: -10282.70

SELL at step 298: Sold -10282.70 shares at -1.74, Cash: 17867.47

BUY at step 300: Bought -10376.32 shares at -1.72, Cash: -0.00, Stock: -10376.32

BUY at step 301: Bought 0.00 shares at -1.75, Cash: -0.00, Stock: -10376.32

BUY at step 302: Bought 0.00 shares at -1.71, Cash: -0.00, Stock: -10376.32

BUY at step 303: Bought 0.00 shares at -1.69, Cash: 0.00, Stock: -10376.32

SELL at step 305: Sold -10376.32 shares at -1.72, Cash: 17796.60

BUY at step 306: Bought -9927.98 shares at -1.79, Cash: 0.00, Stock: -9927.98

SELL at step 310: Sold -9927.98 shares at -1.64, Cash: 16277.26

BUY at step 311: Bought -9913.52 shares at -1.64, Cash: 0.00, Stock: -9913.52

SELL at step 315: Sold -9913.52 shares at -1.57, Cash: 15550.60

BUY at step 320: Bought -9630.47 shares at -1.61, Cash: 0.00, Stock: -9630.47

SELL at step 321: Sold -9630.47 shares at -1.60, Cash: 15364.27

BUY at step 324: Bought -9714.68 shares at -1.58, Cash: 0.00, Stock: -9714.68

SELL at step 326: Sold -9714.68 shares at -1.55, Cash: 14995.29

BUY at step 328: Bought -9621.16 shares at -1.56, Cash: 0.00, Stock: -9621.16

SELL at step 329: Sold -9621.16 shares at -1.55, Cash: 14899.90

BUY at step 331: Bought -10025.26 shares at -1.48, Cash: 0.00, Stock: -10025.26

SELL at step 332: Sold -10025.26 shares at -1.58, Cash: 15810.89

BUY at step 333: Bought -10211.42 shares at -1.55, Cash: 0.00, Stock: -10211.42

SELL at step 334: Sold -10211.42 shares at -1.59, Cash: 16212.87

Final Portfolio Value: 16212.87, Remaining Cash: 16212.87, Remaining Stock at price 0.45: 0.00. Initial price was -1.7300534069260411

Amount earned: 6212.872546207593

Results for B\_TABL:

16212.872546207593

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Initial cash: 10000, transaction cost: 0.001

BUY at step 0: Bought -5774.39 shares at -1.73, Cash: -0.00, Stock: -5774.39

BUY at step 1: Bought 0.00 shares at -1.61, Cash: -0.00, Stock: -5774.39

BUY at step 2: Bought 0.00 shares at -1.56, Cash: -0.00, Stock: -5774.39

BUY at step 3: Bought 0.00 shares at -1.60, Cash: 0.00, Stock: -5774.39

SELL at step 4: Sold -5774.39 shares at -1.57, Cash: 9040.07

BUY at step 6: Bought -5870.77 shares at -1.54, Cash: 0.00, Stock: -5870.77

SELL at step 12: Sold -5870.77 shares at -1.61, Cash: 9440.44

BUY at step 13: Bought -5933.99 shares at -1.59, Cash: 0.00, Stock: -5933.99

SELL at step 15: Sold -5933.99 shares at -1.42, Cash: 8423.80

BUY at step 18: Bought -6079.66 shares at -1.38, Cash: 0.00, Stock: -6079.66

SELL at step 21: Sold -6079.66 shares at -1.03, Cash: 6281.30

BUY at step 24: Bought -6257.82 shares at -1.00, Cash: 0.00, Stock: -6257.82

SELL at step 32: Sold -6257.82 shares at -1.20, Cash: 7498.43

BUY at step 33: Bought -6129.67 shares at -1.22, Cash: 0.00, Stock: -6129.67

SELL at step 59: Sold -6129.67 shares at -1.43, Cash: 8779.60

BUY at step 60: Bought -6146.60 shares at -1.43, Cash: 0.00, Stock: -6146.60

BUY at step 61: Bought -0.00 shares at -1.34, Cash: 0.00, Stock: -6146.60

BUY at step 62: Bought -0.00 shares at -1.36, Cash: 0.00, Stock: -6146.60

BUY at step 63: Bought -0.00 shares at -1.33, Cash: 0.00, Stock: -6146.60

BUY at step 64: Bought -0.00 shares at -1.27, Cash: -0.00, Stock: -6146.60

BUY at step 65: Bought 0.00 shares at -1.26, Cash: 0.00, Stock: -6146.60

SELL at step 88: Sold -6146.60 shares at -1.55, Cash: 9539.88

BUY at step 89: Bought -6236.71 shares at -1.53, Cash: 0.00, Stock: -6236.71

SELL at step 95: Sold -6236.71 shares at -1.44, Cash: 8996.48

BUY at step 96: Bought -6492.00 shares at -1.38, Cash: 0.00, Stock: -6492.00

SELL at step 161: Sold -6492.00 shares at -1.68, Cash: 10891.86

BUY at step 163: Bought -6797.13 shares at -1.60, Cash: 0.00, Stock: -6797.13

SELL at step 195: Sold -6797.13 shares at -1.72, Cash: 11657.88

BUY at step 197: Bought -6717.45 shares at -1.73, Cash: -0.00, Stock: -6717.45

BUY at step 198: Bought 0.00 shares at -1.82, Cash: -0.00, Stock: -6717.45

BUY at step 199: Bought 0.00 shares at -1.81, Cash: 0.00, Stock: -6717.45

SELL at step 295: Sold -6717.45 shares at -1.70, Cash: 11418.50

BUY at step 296: Bought -6445.97 shares at -1.77, Cash: -0.00, Stock: -6445.97

BUY at step 297: Bought 0.00 shares at -1.80, Cash: 0.00, Stock: -6445.97

SELL at step 298: Sold -6445.97 shares at -1.74, Cash: 11200.67

BUY at step 301: Bought -6409.62 shares at -1.75, Cash: -0.00, Stock: -6409.62

BUY at step 302: Bought 0.00 shares at -1.71, Cash: -0.00, Stock: -6409.62

BUY at step 303: Bought 0.00 shares at -1.69, Cash: 0.00, Stock: -6409.62

SELL at step 304: Sold -6409.62 shares at -1.72, Cash: 11024.40

BUY at step 306: Bought -6150.05 shares at -1.79, Cash: 0.00, Stock: -6150.05

SELL at step 312: Sold -6150.05 shares at -1.64, Cash: 10051.90

BUY at step 313: Bought -6297.88 shares at -1.59, Cash: 0.00, Stock: -6297.88

SELL at step 316: Sold -6297.88 shares at -1.54, Cash: 9702.73

BUY at step 317: Bought -6414.74 shares at -1.51, Cash: 0.00, Stock: -6414.74

SELL at step 318: Sold -6414.74 shares at -1.58, Cash: 10119.45

BUY at step 319: Bought -6408.79 shares at -1.58, Cash: -0.00, Stock: -6408.79

BUY at step 320: Bought 0.00 shares at -1.61, Cash: 0.00, Stock: -6408.79

BUY at step 321: Bought -0.00 shares at -1.60, Cash: 0.00, Stock: -6408.79

SELL at step 323: Sold -6408.79 shares at -1.60, Cash: 10229.85

BUY at step 324: Bought -6468.23 shares at -1.58, Cash: 0.00, Stock: -6468.23

BUY at step 325: Bought -0.00 shares at -1.55, Cash: 0.00, Stock: -6468.23

SELL at step 326: Sold -6468.23 shares at -1.55, Cash: 9984.18

BUY at step 328: Bought -6405.97 shares at -1.56, Cash: 0.00, Stock: -6405.97

SELL at step 332: Sold -6405.97 shares at -1.58, Cash: 10102.88

BUY at step 333: Bought -6524.92 shares at -1.55, Cash: 0.00, Stock: -6524.92

BUY at step 334: Bought -0.00 shares at -1.59, Cash: 0.00, Stock: -6524.92

BUY at step 335: Bought -0.00 shares at -1.58, Cash: 0.00, Stock: -6524.92

BUY at step 336: Bought -0.00 shares at -1.60, Cash: 0.00, Stock: -6524.92

Final Portfolio Value: 10444.51, Remaining Cash: 0.00, Remaining Stock at price 0.45: -6524.92. Initial price was -1.7300534069260411

Amount earned: 444.50774818150785

Results for C\_TABL:

10444.507748181508

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Initial cash: 10000, transaction cost: 0.001

BUY at step 1: Bought -6189.73 shares at -1.61, Cash: -0.00, Stock: -6189.73

BUY at step 2: Bought 0.00 shares at -1.56, Cash: -0.00, Stock: -6189.73

BUY at step 3: Bought 0.00 shares at -1.60, Cash: 0.00, Stock: -6189.73

SELL at step 12: Sold -6189.73 shares at -1.61, Cash: 9953.35

BUY at step 21: Bought -9614.58 shares at -1.03, Cash: 0.00, Stock: -9614.58

SELL at step 30: Sold -9614.58 shares at -1.12, Cash: 10790.09

BUY at step 32: Bought -8986.90 shares at -1.20, Cash: -0.00, Stock: -8986.90

SELL at step 33: Sold -8986.90 shares at -1.22, Cash: 10971.71

BUY at step 38: Bought -9726.72 shares at -1.13, Cash: 0.00, Stock: -9726.72

SELL at step 39: Sold -9726.72 shares at -1.23, Cash: 11907.90

BUY at step 45: Bought -9306.95 shares at -1.28, Cash: -0.00, Stock: -9306.95

BUY at step 46: Bought 0.00 shares at -1.41, Cash: -0.00, Stock: -9306.95

BUY at step 47: Bought 0.00 shares at -1.41, Cash: -0.00, Stock: -9306.95

BUY at step 48: Bought 0.00 shares at -1.47, Cash: 0.00, Stock: -9306.95

SELL at step 56: Sold -9306.95 shares at -1.55, Cash: 14399.44

BUY at step 63: Bought -10801.68 shares at -1.33, Cash: 0.00, Stock: -10801.68

SELL at step 66: Sold -10801.68 shares at -1.29, Cash: 13952.47

BUY at step 70: Bought -8546.14 shares at -1.63, Cash: 0.00, Stock: -8546.14

SELL at step 72: Sold -8546.14 shares at -1.73, Cash: 14737.28

BUY at step 74: Bought -9865.04 shares at -1.49, Cash: -0.00, Stock: -9865.04

SELL at step 75: Sold -9865.04 shares at -1.52, Cash: 15008.14

BUY at step 84: Bought -8982.05 shares at -1.67, Cash: 0.00, Stock: -8982.05

SELL at step 86: Sold -8982.05 shares at -1.59, Cash: 14222.42

BUY at step 94: Bought -9580.33 shares at -1.48, Cash: 0.00, Stock: -9580.33

SELL at step 95: Sold -9580.33 shares at -1.44, Cash: 13819.68

BUY at step 96: Bought -9972.50 shares at -1.38, Cash: 0.00, Stock: -9972.50

BUY at step 97: Bought -0.00 shares at -1.38, Cash: 0.00, Stock: -9972.50

BUY at step 98: Bought -0.00 shares at -1.41, Cash: 0.00, Stock: -9972.50

SELL at step 100: Sold -9972.50 shares at -1.38, Cash: 13792.61

BUY at step 111: Bought -10675.24 shares at -1.29, Cash: 0.00, Stock: -10675.24

SELL at step 114: Sold -10675.24 shares at -1.32, Cash: 14102.74

BUY at step 149: Bought -9284.90 shares at -1.52, Cash: -0.00, Stock: -9284.90

BUY at step 150: Bought 0.00 shares at -1.49, Cash: -0.00, Stock: -9284.90

BUY at step 151: Bought 0.00 shares at -1.49, Cash: -0.00, Stock: -9284.90

SELL at step 152: Sold -9284.90 shares at -1.54, Cash: 14323.98

BUY at step 176: Bought -8065.89 shares at -1.77, Cash: 0.00, Stock: -8065.89

SELL at step 177: Sold -8065.89 shares at -1.80, Cash: 14532.64

BUY at step 180: Bought -7796.29 shares at -1.86, Cash: 0.00, Stock: -7796.29

SELL at step 181: Sold -7796.29 shares at -1.81, Cash: 14103.14

BUY at step 184: Bought -7579.65 shares at -1.86, Cash: 0.00, Stock: -7579.65

SELL at step 192: Sold -7579.65 shares at -1.66, Cash: 12603.46

BUY at step 193: Bought -7377.62 shares at -1.71, Cash: 0.00, Stock: -7377.62

SELL at step 194: Sold -7377.62 shares at -1.72, Cash: 12691.03

BUY at step 197: Bought -7312.77 shares at -1.73, Cash: 0.00, Stock: -7312.77

SELL at step 200: Sold -7312.77 shares at -1.80, Cash: 13150.84

BUY at step 202: Bought -7253.62 shares at -1.81, Cash: 0.00, Stock: -7253.62

SELL at step 217: Sold -7253.62 shares at -1.80, Cash: 13050.17

BUY at step 254: Bought -6672.03 shares at -1.95, Cash: -0.00, Stock: -6672.03

SELL at step 255: Sold -6672.03 shares at -1.96, Cash: 13092.10

BUY at step 270: Bought -7141.28 shares at -1.83, Cash: 0.00, Stock: -7141.28

SELL at step 271: Sold -7141.28 shares at -1.81, Cash: 12942.55

BUY at step 272: Bought -7215.57 shares at -1.79, Cash: -0.00, Stock: -7215.57

SELL at step 273: Sold -7215.57 shares at -1.80, Cash: 12988.32

BUY at step 274: Bought -7325.53 shares at -1.77, Cash: 0.00, Stock: -7325.53

SELL at step 275: Sold -7325.53 shares at -1.76, Cash: 12856.52

BUY at step 291: Bought -7698.24 shares at -1.67, Cash: -0.00, Stock: -7698.24

SELL at step 292: Sold -7698.24 shares at -1.69, Cash: 13006.96

BUY at step 301: Bought -7443.27 shares at -1.75, Cash: 0.00, Stock: -7443.27

SELL at step 302: Sold -7443.27 shares at -1.71, Cash: 12737.61

BUY at step 303: Bought -7512.32 shares at -1.69, Cash: 0.00, Stock: -7512.32

SELL at step 304: Sold -7512.32 shares at -1.72, Cash: 12921.01

BUY at step 305: Bought -7518.55 shares at -1.72, Cash: 0.00, Stock: -7518.55

SELL at step 306: Sold -7518.55 shares at -1.79, Cash: 13450.61

BUY at step 307: Bought -7521.31 shares at -1.79, Cash: 0.00, Stock: -7521.31

SELL at step 312: Sold -7521.31 shares at -1.64, Cash: 12293.13

BUY at step 320: Bought -7613.13 shares at -1.61, Cash: -0.00, Stock: -7613.13

BUY at step 321: Bought 0.00 shares at -1.60, Cash: 0.00, Stock: -7613.13

Final Portfolio Value: 12186.42, Remaining Cash: 0.00, Remaining Stock at price 0.45: -7613.13. Initial price was -1.7300534069260411

Amount earned: 2186.416162214331

Results for DeepLOBSeq2Seq:

12186.416162214331

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Initial cash: 10000, transaction cost: 0.001

BUY at step 0: Bought -5774.39 shares at -1.73, Cash: -0.00, Stock: -5774.39

BUY at step 1: Bought 0.00 shares at -1.61, Cash: -0.00, Stock: -5774.39

BUY at step 2: Bought 0.00 shares at -1.56, Cash: -0.00, Stock: -5774.39

SELL at step 3: Sold -5774.39 shares at -1.60, Cash: 9217.15

BUY at step 5: Bought -5863.61 shares at -1.57, Cash: 0.00, Stock: -5863.61

SELL at step 13: Sold -5863.61 shares at -1.59, Cash: 9309.83

BUY at step 83: Bought -5372.06 shares at -1.73, Cash: 0.00, Stock: -5372.06

SELL at step 84: Sold -5372.06 shares at -1.67, Cash: 8958.26

BUY at step 151: Bought -6023.98 shares at -1.49, Cash: 0.00, Stock: -6023.98

BUY at step 152: Bought -0.00 shares at -1.54, Cash: -0.00, Stock: -6023.98

SELL at step 153: Sold -6023.98 shares at -1.56, Cash: 9385.43

BUY at step 156: Bought -5904.92 shares at -1.59, Cash: 0.00, Stock: -5904.92

SELL at step 159: Sold -5904.92 shares at -1.64, Cash: 9676.30

BUY at step 180: Bought -5191.02 shares at -1.86, Cash: -0.00, Stock: -5191.02

SELL at step 181: Sold -5191.02 shares at -1.81, Cash: 9390.32

BUY at step 185: Bought -5044.46 shares at -1.86, Cash: -0.00, Stock: -5044.46

BUY at step 186: Bought 0.00 shares at -1.84, Cash: 0.00, Stock: -5044.46

SELL at step 188: Sold -5044.46 shares at -1.77, Cash: 8911.00

BUY at step 190: Bought -5230.52 shares at -1.70, Cash: 0.00, Stock: -5230.52

SELL at step 217: Sold -5230.52 shares at -1.80, Cash: 9410.36

BUY at step 220: Bought -4845.01 shares at -1.94, Cash: 0.00, Stock: -4845.01

SELL at step 221: Sold -4845.01 shares at -1.95, Cash: 9428.90

BUY at step 229: Bought -4682.10 shares at -2.01, Cash: 0.00, Stock: -4682.10

SELL at step 234: Sold -4682.10 shares at -1.89, Cash: 8825.53

BUY at step 238: Bought -4916.36 shares at -1.79, Cash: 0.00, Stock: -4916.36

SELL at step 243: Sold -4916.36 shares at -1.97, Cash: 9663.77

BUY at step 274: Bought -5450.46 shares at -1.77, Cash: 0.00, Stock: -5450.46

SELL at step 277: Sold -5450.46 shares at -1.73, Cash: 9443.04

BUY at step 280: Bought -5605.80 shares at -1.68, Cash: 0.00, Stock: -5605.80

SELL at step 281: Sold -5605.80 shares at -1.66, Cash: 9315.05

BUY at step 285: Bought -5548.06 shares at -1.68, Cash: 0.00, Stock: -5548.06

SELL at step 286: Sold -5548.06 shares at -1.66, Cash: 9228.17

BUY at step 290: Bought -5514.47 shares at -1.67, Cash: 0.00, Stock: -5514.47

SELL at step 293: Sold -5514.47 shares at -1.72, Cash: 9448.59

BUY at step 294: Bought -5551.62 shares at -1.70, Cash: 0.00, Stock: -5551.62

SELL at step 298: Sold -5551.62 shares at -1.74, Cash: 9646.63

BUY at step 302: Bought -5625.79 shares at -1.71, Cash: 0.00, Stock: -5625.79

SELL at step 312: Sold -5625.79 shares at -1.64, Cash: 9195.01

BUY at step 315: Bought -5850.12 shares at -1.57, Cash: 0.00, Stock: -5850.12

Final Portfolio Value: 9364.34, Remaining Cash: 0.00, Remaining Stock at price 0.45: -5850.12. Initial price was -1.7300534069260411

Amount earned: -635.6574019080945

Results for DeepLOBAttention:

9364.342598091906

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Click to add a cell.