# יישומים ברשתות נוירונים עמוקות

## תרגיל בית 5

קוד זה מגדיר מודל מקודד אוטומטי לשחזור תמונה, מיושם עם PyTorch. המקודד האוטומטי מורכב מקודד ומפענח, שניהם מיושמים כסדרה של שכבות Conv2d ו-ReLU, ושכבת ConvTranspose2d עבור המפענח. המקודד והמפענח משולבים בשיטה קדימה של מחלקת Autoencoder. המודל מאומן באמצעות אופטימיזציית Adam ופונקציית אובדן MSE. מערכי ההדרכה והבדיקה הם תמונות MNIST, הנטענות באמצעות מודול מערכי הנתונים של PyTorch והופכות לטנזורים באמצעות מודול ההמרה. ניתן לאמן את הדגם עם או בלי רעש נוסף לתמונות הקלט. הפסדי האימון והבדיקה של המודל מאוחסנים average_train_loss and average_test_loss respectively..

ניתן להתאים את ההיפרפרמטרים של המודל על ידי העברתם כארגומנטים לבנאי Autoencoder. ערכי ברירת המחדל הם:

noise: TRUE/False

noise factor: 0.4

epochs: 20

batch Size: 64

learning Rate: 1e-4

להלן תיאור של כל פונקציה בקוד:

__init__: זהו הבנאי למחלקה Autoencoder. הוא מאתחל את רשתות המקודד והמפענח, מגדיר את ההתקן לשימוש לאימון (או GPU או CPU), מגדיר את מעמיסי הנתונים וההיפרפרמטרים של ההדרכה והבדיקה, ומאתחל את פונקציית האופטימיזציה וההפסד.

```
def __init__(self, noise=False, level_of_noise=0.4, epochs=100,
batchSize=128, learningRate=1e-3):
    super(Autoencoder, self).__init__()
    self.encoder = nn.Sequential(
        nn.Conv2d(1, 16, 3, stride=2, padding=1),
        nn.ReLU(),
        nn.Conv2d(16, 32, 3, stride=2, padding=1),
        nn.ReLU(),
        nn.Conv2d(32, 64, 7)
    )
    self.decoder = nn.Sequential(
        nn.ConvTranspose2d(64, 32, 7),
        nn.ReLU(),
        nn.ConvTranspose2d(32, 16, 3, stride=2, padding=1,
output_padding=1),
        nn.ReLU(),
        nn.ConvTranspose2d(16, 1, 3, stride=2, padding=1,
output_padding=1),
        nn.Sigmoid()
    )
    self.device = torch.device("cuda" if torch.cuda.is_available() else
"cpu")
```

```
    self.to(self.device)
    self.epochs = epochs
    self.batchSize = batchSize
    self.learningRate = learningRate
    self.noise = noise
    self.criterion = nn.MSELoss()
    self.optimizer = torch.optim.Adam(self.parameters(),
lr=self.learningRate)  # , weight_decay=1e-5
    self.train_mnist_data = datasets.MNIST('data', train=True,
download=True, transform=transforms.ToTensor())
    self.train_loader = torch.utils.data.DataLoader(self.train_mnist_data,

batch_size=self.batchSize,
                                                    shuffle=True)
    self.test_mnist_data = datasets.MNIST('data', train=False,
download=True, transform=transforms.ToTensor())
    self.test_loader = torch.utils.data.DataLoader(self.test_mnist_data,

batch_size=self.batchSize,
                                                   shuffle=True)
    self.level_of_noise = level_of_noise
    self.average_train_loss, self.average_test_loss = list(), list()
```

forward: שיטה זו מגדירה את המעבר קדימה של המקודד האוטומטי. הוא קולט טנזור קלט ומחיל עליו את רשתות המקודד והמפענח, ומחזיר את הפלט המשוחזר.

```
def forward(self, x):
    x = self.encoder(x)
    x = self.decoder(x)
    return x
```

train: שיטה זו מאמנת את מודל המקודד האוטומטי על נתוני האימון עבור מספר מוגדר של תקופות. בכל תקופה, הוא חוזר על נתוני האימון, מחיל את המודל על תמונות הקלט, מחשב את ההפסד ומבצע הפצה לאחור ואופטימיזציה. זה גם מחשב ומאחסן את הפסדי האימונים והבדיקות הממוצעים עבור כל תקופה.

```
def train(self, **kwargs):
    torch.manual_seed(42)
    for epoch in range(self.epochs):
        train_loss_per_epoch = list()
        for Image, label in self.train_loader:
            imgIn2Autoencoder = torch.clone(Image)
            if self.noise:
                imgIn2Autoencoder = imgIn2Autoencoder + self.level_of_noise
* np.random.normal(
                    size=imgIn2Autoencoder.shape).astype(
                    "float32")
            Image = Image.to(self.device)
            imgIn2Autoencoder = imgIn2Autoencoder.to(self.device)
            OutPut = self(imgIn2Autoencoder)
            loss = self.criterion(OutPut, Image)
            train_loss_per_epoch.append(float(loss))
            loss.backward()
            self.optimizer.step()
            self.optimizer.zero_grad()

self.average_train_loss.append(np.array(train_loss_per_epoch).mean())
        self.average_test_loss.append(self.function_test())
```

```
        print('Epoch:{}, train loss:{:.4f}, test loss :{:.4f}'.format(epoch
+ 1, self.average_train_loss[epoch],

self.average_test_loss[epoch]))

    return self.average_train_loss, self.average_test_loss
```

function_test: שיטה זו בודקת את מודל המקודד האוטומטי על נתוני הבדיקה ומחזירה את אובדן הבדיקה הממוצע.

```
def function_test(self):

    test_loss = []
    for Image, label in self.test_loader:
        imgIn2Autoencoder = torch.clone(Image)
        Image = Image.to(self.device)

        if self.noise:
            imgIn2Autoencoder = imgIn2Autoencoder + self.level_of_noise *
np.random.normal(
                size=imgIn2Autoencoder.shape).astype(
                "float32")

        imgIn2Autoencoder = imgIn2Autoencoder.to(self.device)
        OutPut = self(imgIn2Autoencoder)
        loss = self.criterion(OutPut, Image)
        test_loss.append(float(loss))

    return np.array(test_loss).mean()
```

print_loss: שיטה זו משרטטת את הפסדי האימון והבדיקה על פני מספר העידנים.

```
def print_loss(self):
    fig, ax = plt.subplots()
    ax.plot(range(self.epochs), self.average_train_loss, label='Training
Loss')
    ax.plot(range(self.epochs), self.average_test_loss, label='Testing
Loss')
    ax.set_title('Training and Testing Loss')
    ax.set_xlabel('Epochs')
    ax.set_ylabel('Loss')
    ax.set_xticks(range(self.epochs))
    ax.legend(loc='best')
    plt.show()
```

printResult: שיטה זו לוקחת קול שלם ומדמיינת את התמונות המקוריות, הרועשות (אם רעש מופעל) ומשוחזרות עבור מספר מוגדר של תמונות במערך הבדיקה. זה עושה זאת על ידי טעינת תחילה אצווה של תמונות בדיקה, הוספת רעש לתמונות אם הרעש מופעל, והעברת התמונות דרך מודל המקודד האוטומטי כדי לקבל את הפלטים המשוחזרים. לאחר מכן, התמונות המקוריות, הרועשות והמשוחזרות משורטטות ברשת של עלילות משנה, עם עלילות משנה קוליות בכל שורה. מספר השורות בעלילה תלוי אם רעש מופעל: אם רעש מופעל, יהיו שלוש שורות (אחת לתמונות המקוריות, אחת לתמונות הרועשות ואחת לתמונות המשוחזרות), ואילו אם הרעש מושבת , יהיו רק שתי שורות (אחת למקור ואחת לתמונות המשוחזרות). לאחר מכן נקראת שיטת ההצגה כדי להציג את העלילה.

```
def printResult(self, col):
    dataiter = iter(self.test_loader)
    Image, label = next(dataiter)
```

```
    imgIn2Autoencoder = torch.clone(Image)
    Image = Image.numpy()
    if self.noise:
        imgIn2Autoencoder = imgIn2Autoencoder + self.level_of_noise *
np.random.normal(
            size=imgIn2Autoencoder.shape).astype(
            "float32")
    noisy_images = deepcopy(imgIn2Autoencoder)
    imgIn2Autoencoder = imgIn2Autoencoder.to(self.device)
    OutPut = self(imgIn2Autoencoder)
    OutPut = OutPut.view(64, 1, 28, 28)
    OutPut = OutPut.to('cpu')
    OutPut = OutPut.detach().numpy()
    howManyRow = 3 if self.noise else 2
    plt.figure(figsize=(20, 10))
    for i in range(col):
        ax = plt.subplot(howManyRow, col, i + 1)
        plt.imshow(Image[i].squeeze(), cmap='gray')
        ax.set_axis_off()

        if self.noise:
            ax = plt.subplot(howManyRow, col, i + 1 + (howManyRow - 2) *
col)
            plt.imshow(noisy_images[i].squeeze(), cmap='gray')
            ax.set_axis_off()

        ax = plt.subplot(howManyRow, col, i + 1 + (howManyRow - 1) * col)
        plt.imshow(OutPut[i].squeeze(), cmap='gray')
        ax.set_axis_off()

    plt.show()
```

main: קוד זה מגדיר פונקציה עיקרית שיוצרת מופע של מחלקת Autoencoder, מאמן אותו במערך הנתונים של MNIST, ומדמיינת את התמונות המקוריות, הרועשות (אם רעש מופעל), והמשוחזרות עבור מספר תמונות בערכת הבדיקה. זה גם משרטט את הפסדי האימונים והבדיקות על פני מספר העידנים.

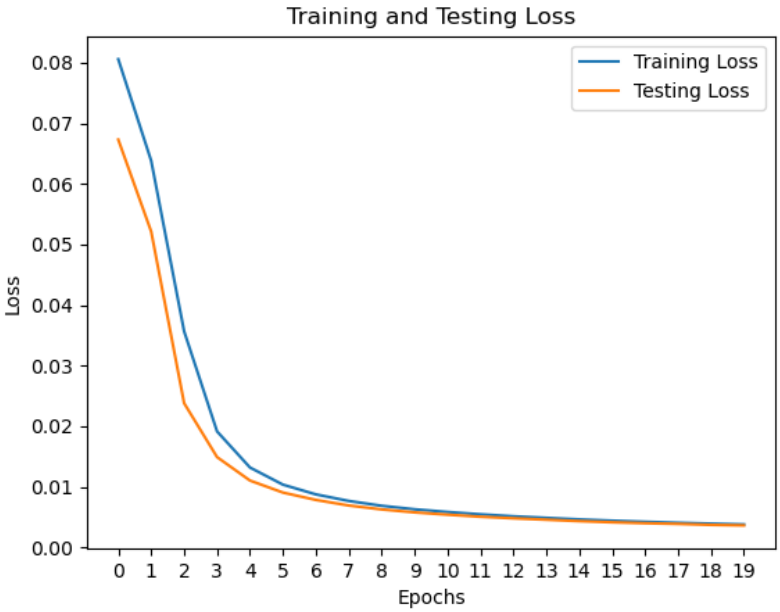
```
def main(noise=True, n_image2print=12):
    _Autoencoder = Autoencoder(noise=noise, level_of_noise=0.4, epochs=20,
batchSize=64, learningRate=1e-4)
    _avg_train_loss, _avg_test_loss = _Autoencoder.train()
    _Autoencoder.printResult(n_image2print)
    _Autoencoder.print_loss()
```

# תוצאות הרצה:

1) עבור שאלה מספר 1:

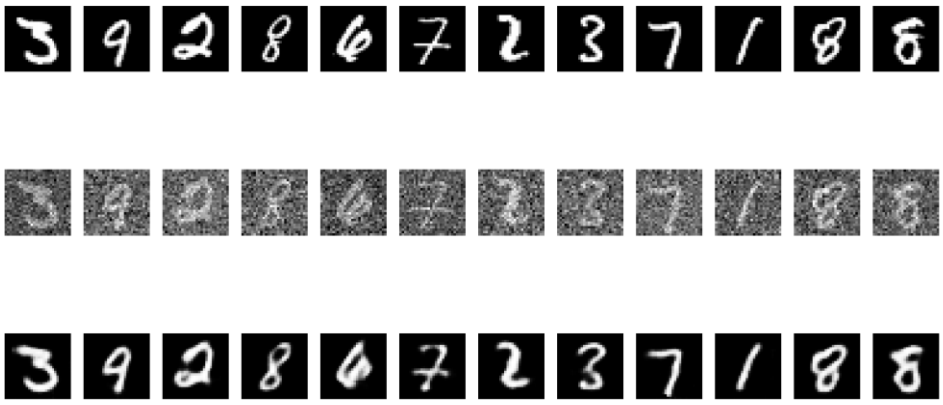
הצגת 12 תמונה (למעלה התמונות המקוריות ולמטה את תוצאות הרשת (reconstructed))

הצגת פונקציית השגיאה :

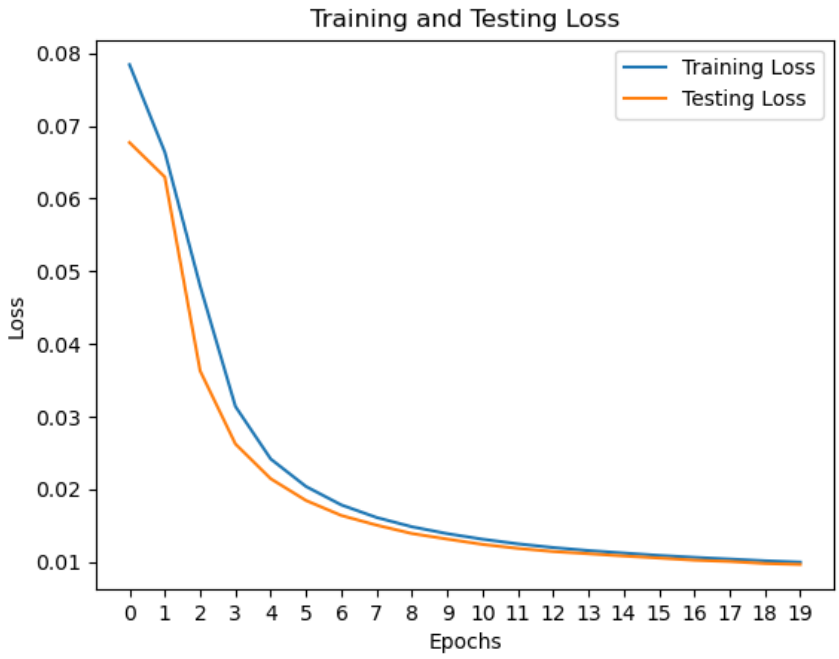



2) עבור שאלה מספר 2:

הצגת 12 תמונה (למעלה התמונות המקוריות ובאמצע את התמונות הרועשות ולמטה את תוצאות הרשת (reconstructed))



הצגת פונקציית השגיאה :




קטע הקוד שהוספתי לכל תמונה מקורית כך שתכיל רעש :

```
1) if self.noise:
2)     imgIn2Autoencoder = imgIn2Autoencoder + self.level_of_noise *
np.random.normal(
        size=imgIn2Autoencoder.shape).astype(
        "float32")
```