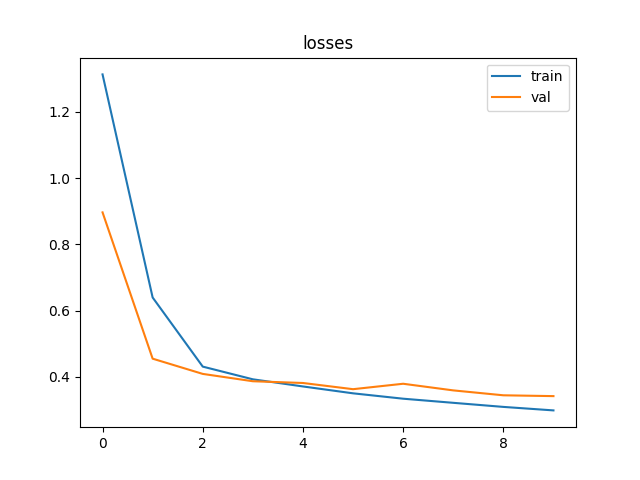
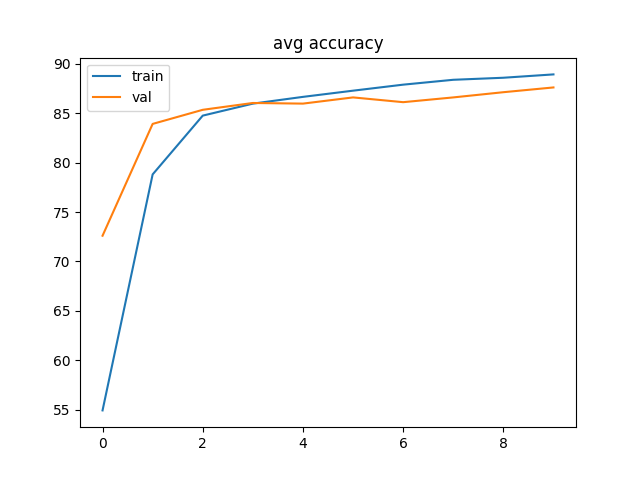
**דו"ח תרגיל 4**

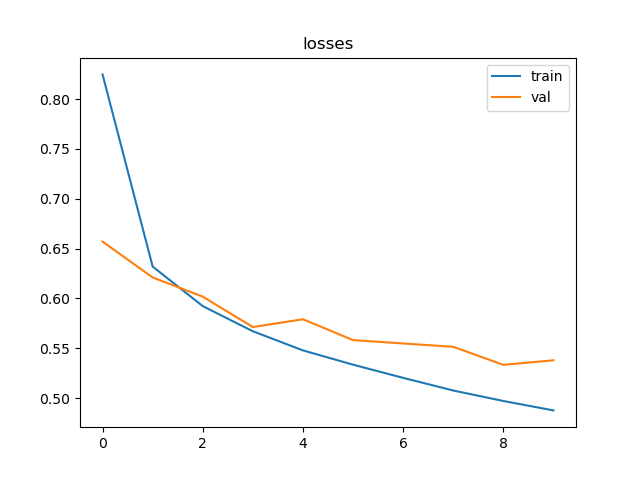
**מודל A**

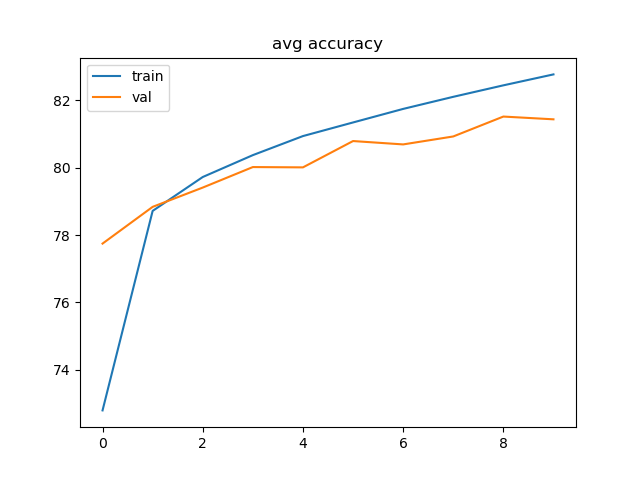


1. 
2. דיוק על test set של Fashion Mnist- 86.34%
3. היפר פרמטרים:

* Learning rate – 0.01
* Momentum (sgd) – 0.9
* Batch size – 64

**מודל B:**

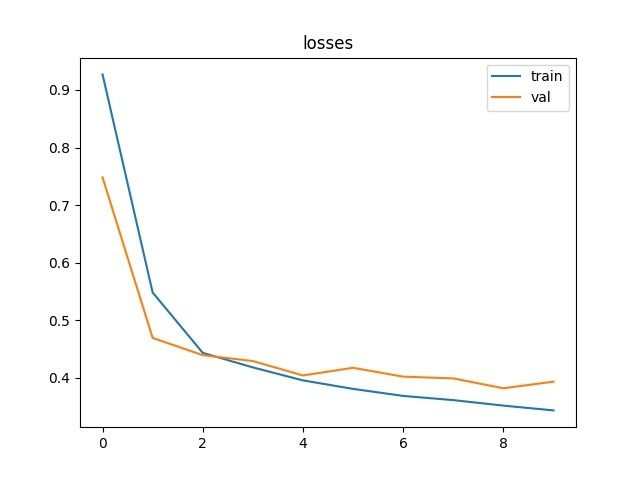


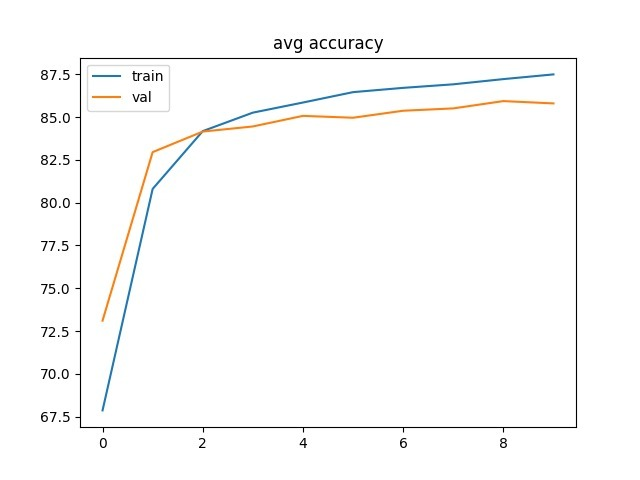


1. דיוק על test set של Fashion Mnist- 80.86%
2. היפר פרמטרים:

* Learning rate – 0.001
* Batch size – 64

**מודל C:**

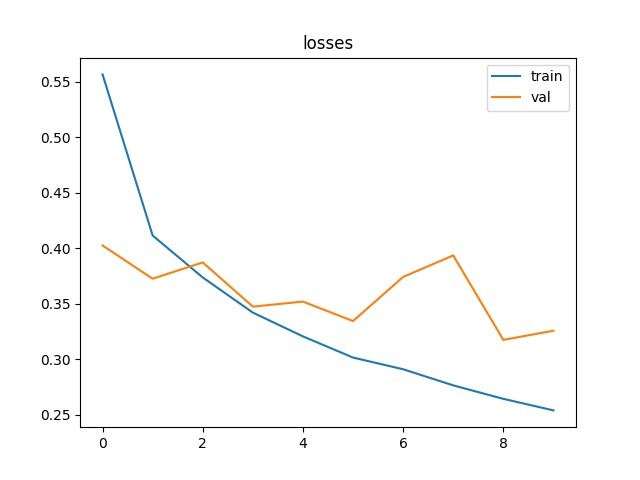
1.

2.

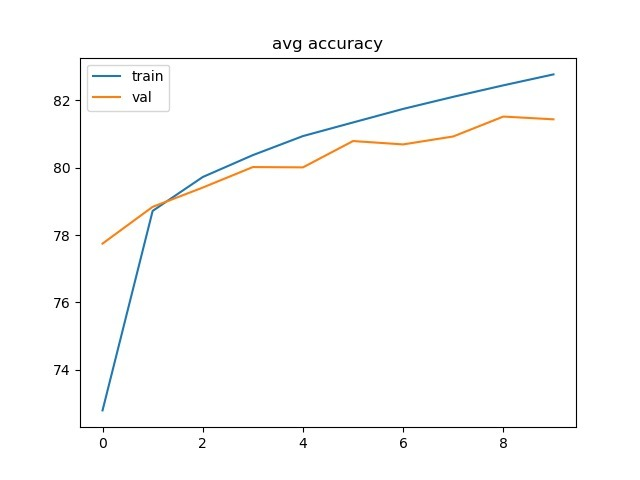
1. דיוק על test set של Fashion Mnist- 85%
2. היפר פרמטרים:

* Learning rate – 0.001
* Dropout – [0.3,0.1]
* Batch size – 64

**מודל D1 (batch norm מופעל לפני פונקציית האקטיבציה):**



1.

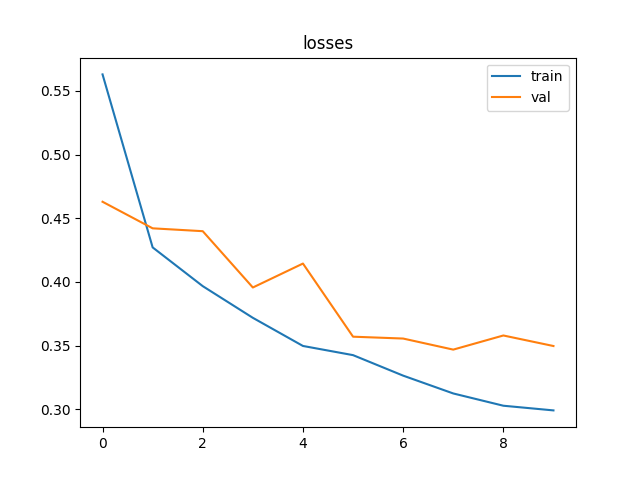


2.

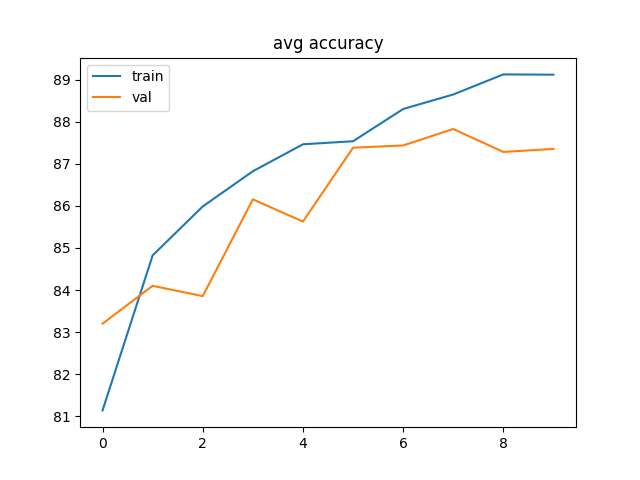
1. דיוק על test set של Fashion Mnist- 87.5
2. היפר פרמטרים:

* Learning rate – 0.01
* Batch size – 64

**מודל D2 (batch norm מופעל אחרי פונקציית האקטיבציה):**

 1.

2.

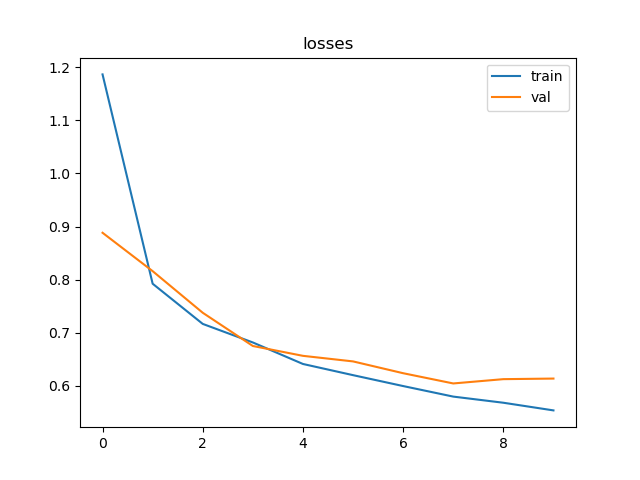


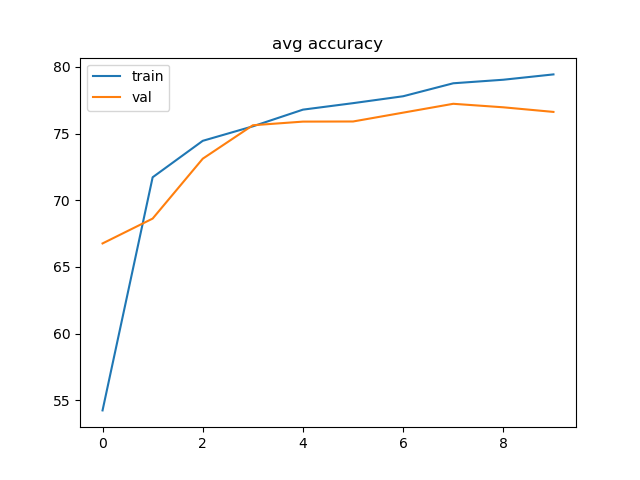
1. דיוק על test set של Fashion Mnist- 86.82%
2. היפר פרמטרים:

* Learning rate – 0.01
* Batch size – 64

**מודל E:**

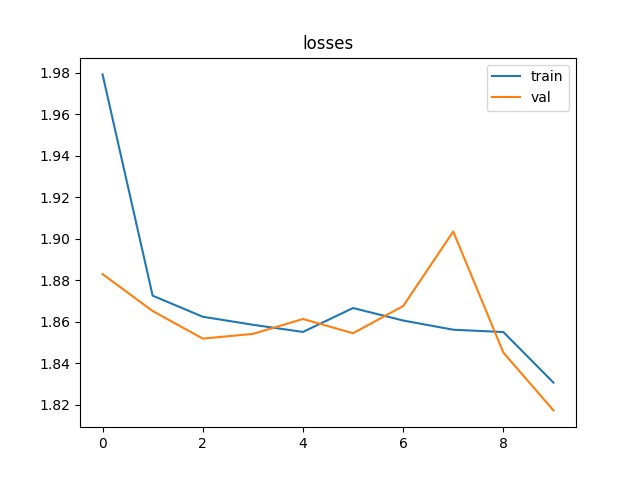
1.

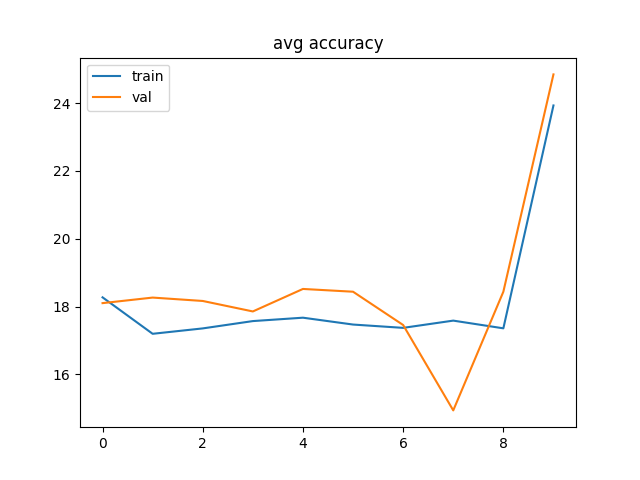


1. 
2. דיוק על test set של Fashion Mnist- 76.61%
3. היפר פרמטרים:

* Learning rate – 0.001
* Batch size – 64

**מודל F:**

1 .



2.

1. דיוק על test set של Fashion Mnist- 24.18%
2. היפר פרמטרים:

* Learning rate – 0.01
* Batch size – 64

**הארכיטקטורה של המודל הטוב ביותר:**

יצרנו 3 שכבות fully-conected בגדלים: 256,100,50

השתמשנו בbatchnorm בין כל שכבה, והשתמשנו בrelu כפונקציית אקטיבציה.

את השכבה האחרונה העברנו בsoftmax

היפר פרמטרים שהשתמשנו בהם:

Learning rate-0.001

Batch size – 64

Epochs - 0

השתמשנו בAdam כאופטימייזר

self.image\_size = image\_size  
self.fc0 = nn.Linear(image\_size, 256)  
self.fc0\_bn = nn.BatchNorm1d(256)  
self.fc1 = nn.Linear(256, 100)  
self.fc1\_bn = nn.BatchNorm1d(100)  
self.fc2 = nn.Linear(100, 50)  
self.fc2\_bn = nn.BatchNorm1d(50)  
self.fc3= nn.Linear(50, num\_cls)

**הארכיטקטורה של המודל הטוב ביותר בשימוש בקונבולוציה:**

יצרנו שתי שכבות קונבולוציה. בשכבה אחת יש 6 קרנלים, כל קרנל בגודל 5\*5, ובשכבה השנייה יש 12 קרנלים באותו גודל.

בנוסף יצרנו שתי שכבות fully connected בגדלים 120 ו60.

את התוצאה העברנו בsoftmax.

השתמשנו באופטימייזר Adam

היפר פרמטרים:

Learning rate – 0.001

Batch size – 64

Epochs - 40

def \_\_init\_\_(self,image\_size, num\_cls=10):  
 super(BestCnn, self).\_\_init\_\_()  
  
 self.conv1 = nn.Conv2d(in\_channels=1, out\_channels=6, kernel\_size=5)  
 self.conv2 = nn.Conv2d(in\_channels=6, out\_channels=12, kernel\_size=5)  
  
 self.fc1 = nn.Linear(in\_features=12 \* 4 \* 4, out\_features=120)  
 self.fc2 = nn.Linear(in\_features=120, out\_features=60)  
 self.out = nn.Linear(in\_features=60, out\_features=10)  
  
  
  
def save\_model(self, path):  
 torch.save(self.state\_dict(), path)  
  
def load\_model(self, path):  
 self.load\_state\_dict(torch.load(path))  
  
def forward(self, x):  
 x = x.view((x.size()[0],1,28,28))  
 # conv 1  
 t = self.conv1(x)  
 t = F.relu(t)  
 t = F.max\_pool2d(t, kernel\_size=2, stride=2)  
  
 # conv 2  
 t = self.conv2(t)  
 t = F.relu(t)  
 t = F.max\_pool2d(t, kernel\_size=2, stride=2)  
  
 # fc1  
 t = t.reshape(-1, 12 \* 4 \* 4)  
 t = self.fc1(t)  
 t = F.relu(t)  
  
 # fc2  
 t = self.fc2(t)  
 t = F.relu(t)  
  
 # output  
 t = self.out(t)  
 # don't need softmax here since we'll use cross-entropy as activation.