**МІНІСТЕРСТВО ОСВІТИ І НАУКИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

**Інститут комп'ютерних наук та інформаційних технологій**

**Кафедра систем штучного інтелекту**



**Звіт до лабораторної роботи №3**

**з дисципліни “ Машинне навчання ”**

**На тему: “ Згорткова нейронна мережа ”**

**Виконав**:

ст. гр. КН-311

Ткачук Орест

**Викладач:**

Бойчук А. Р.

Львів – 2021

**Мета:** засвоїти основні відомості про роботу з згортковими нейронними мережами. Навчитись використовувати їх для класифікації зображень.

**Завдання:** реалізувати згорткову нейронну мережу використовуючи PyTorch(рекомендовано) чи Tensorflow та натренувати її на заданому наборі даних, а саме:

1. Розділити дані на тренувальні(60 000) і тестові(10 000).
2. Реалізувати згорткову нейронну мережу на PyTorch.
3. Реалізувати тренувальний та валідаційні цикли.
4. Реалізувати тестувальний цикл.
5. Натренувати задану мережу до максимально можливої точності. Вивести графіки тренувальної та валідаційної точностей та функції втрат відносно ітерацій/епох.
6. Звізуалізувати найгірші 10 передбачень(ті, в яких мережа найбільше помилилась відносно правильних відповідей), порівняти їх з вдалими передбаченнями цього ж класу. Спробувати пояснити причини поганих передбачень.
7. Покращити архітектуру мережі відомими вам методами. При цьому внести мінімум змін в її архітектуру. Провести 2-3 ітерації покращення.
8. Натренувати мережу до заданої точності. Вивести графіки тренувальної та валідаційної точностей та функції втрат відносно ітерацій/епох.
9. Звізуалізувати найгірші 10 передбачень(ті, в яких мережа найбільше помилилась відносно правильних відповідей), порівняти їх з вдалими передбаченнями цього ж класу. Спробувати пояснити причини поганих передбачень.
10. Натреновану мережу в пункті 5 та найкращу мережу в пункті 8 зберегти в форматі onnx та вкласти при здачі завдання.

**Хід роботи**

**Варіант 5**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Варіант # | Датасет | Accuracy | Activation function | Модель |
| 5 | [MNIST](https://pytorch.org/docs/stable/torchvision/datasets.html#mnist) | 95% | [Sigmoid](https://pytorch.org/docs/stable/nn.html#sigmoid) | Lenet 5 |

1. Розділити дані на тренувальні(60 000) і тестові(10 000).

def\_trans = transforms.Compose([transforms.ToTensor()])

train\_dataset = datasets.MNIST(root=root, train=True, transform=def\_trans, download=True)

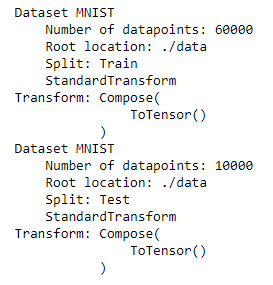
test\_dataset = datasets.MNIST(root=root, train=False, transform=def\_trans, download=True)

train\_loader = DataLoader(dataset=train\_dataset, batch\_size=batch\_size, shuffle=True)

test\_loader = DataLoader(dataset=test\_dataset, batch\_size=batch\_size, shuffle=False)

print(train\_dataset)

print(test\_dataset)



1. Реалізувати згорткову нейронну мережу на PyTorch.

class Model(nn.Module):

def \_\_init\_\_(self, n\_classes):

super(Model, self).\_\_init\_\_()

self.feature\_extractor = nn.Sequential(

nn.Conv2d(in\_channels=1, out\_channels=6, kernel\_size=5, stride=1, padding = 2),

#nn.BatchNorm2d(num\_features=6),

nn.Sigmoid(),

nn.AvgPool2d(kernel\_size=2, stride=2),

nn.Conv2d(in\_channels=6, out\_channels=16, kernel\_size=5, stride=1),

#nn.BatchNorm2d(num\_features=16),

nn.Sigmoid(),

nn.AvgPool2d(kernel\_size=2, stride=2),

#nn.Softmax(dim = 2)

)

self.classifier = nn.Sequential(

nn.Linear(in\_features=400, out\_features=120),

nn.Sigmoid(),

nn.Linear(in\_features=120, out\_features=84),

nn.Sigmoid(),

nn.Linear(in\_features=84, out\_features=n\_classes),

nn.Softmax(dim = 1),

)

def forward(self, x):

x = self.feature\_extractor(x)

x = torch.flatten(x, 1)

x = self.classifier(x)

return x

1. Реалізувати тренувальний та валідаційні цикли.

def train(train\_data, get\_model, get\_criterion, get\_optimizer, size, device):

training\_loss = 0

get\_model.train()

for inputs, labels in train\_data:

inputs = inputs.to(device)

labels = labels.to(device)

get\_optimizer.zero\_grad()

outputs = get\_model(inputs)

loss = get\_criterion(outputs, labels)

loss.backward()

get\_optimizer.step()

training\_loss += loss.data \* inputs.shape[0]

epoch\_loss = training\_loss / size["train"]

return get\_model, get\_optimizer, epoch\_loss

def valid(test\_data, get\_model, get\_criterion, get\_optimizer, size, device):

valid\_loss = 0

get\_model.eval()

for inputs, labels in test\_data:

inputs = inputs.to(device)

labels = labels.to(device)

outputs = get\_model(inputs)

get\_optimizer.zero\_grad()

loss = get\_criterion(outputs, labels)

valid\_loss += loss.data \* inputs.shape[0]

epoch\_valid\_loss = valid\_loss / size["test"]

return get\_model, get\_optimizer, epoch\_valid\_loss

1. Реалізувати тестувальний цикл.

def test\_accuracy(loader, get\_model, train\_state):

num\_correct = num\_samples = 0

get\_model.eval()

with torch.no\_grad():

for data in loader:

inputs, labels = data

inputs = inputs.to(device)

labels = labels.to(device)

outputs = get\_model(inputs)

\_, preds = outputs.max(1)

num\_correct += (preds == labels).sum()

num\_samples += preds.size(0)

accuracy = (num\_correct.item() / num\_samples) \* 100

if train\_state:

print("Model Predicted {} correctly out of {} from training dataset, "

"Accuracy: {:.2f}".format(num\_correct.item(), num\_samples, accuracy))

else:

print("Model Predicted {} correctly out of {} from testing dataset, Accuracy: {:.2f}".format(num\_correct.item(), num\_samples, accuracy))

get\_model.train()

1. Натренувати задану мережу до максимально можливої точності. Вивести графіки тренувальної та валідаційної точностей та функції втрат відносно ітерацій/епох.

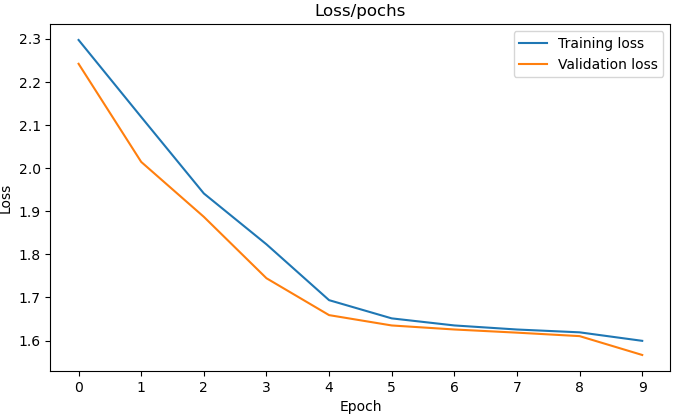
epoch 10

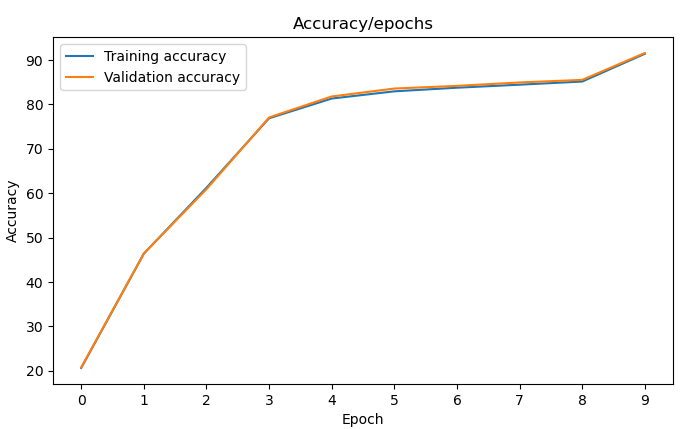
Training loss:1.5993

Training accuracy:91.39

Validation loss:1.5665

Validation accuracy:91.57





1. Звізуалізувати найгірші 10 передбачень(ті, в яких мережа найбільше помилилась відносно правильних відповідей).

def top\_errors(loader, get\_model):

model.eval()

inp = 0

errors = []

for data in test\_loader:

inputs, labels = data

inputs = inputs.to(device)

labels = labels.to(device)

outputs = model(inputs)

\_, preds = outputs.max(1)

values = (preds == labels).data.cpu().numpy()

values = np.where(values == False)[0]

if len(values) > 0:

error = []

pred\_value = outputs[values].data.cpu().numpy().flatten()[preds[values].data.cpu().numpy()]

lable\_value = outputs[values].data.cpu().numpy().flatten()[labels[values].data.cpu().numpy()]

error\_value = abs(pred\_value - lable\_value)

for i in range(len(values)):

error.append(preds[values[i]].data.cpu().numpy())

error.append(labels[values[i]].data.cpu().numpy())

error.append(error\_value[i])

img = inputs[values[i]].data.cpu().resize\_(28,28)

error.append(img)

errors.append(error)

for error in sorted(errors,key=lambda x: x[2], reverse=True)[:10]:

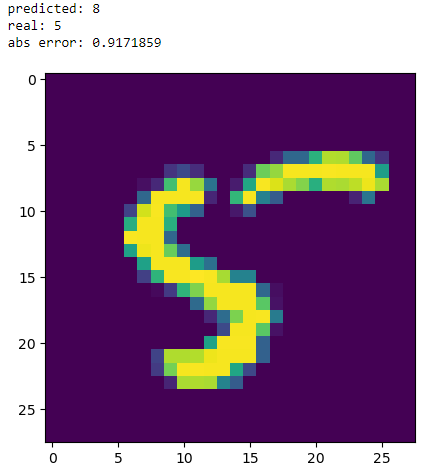
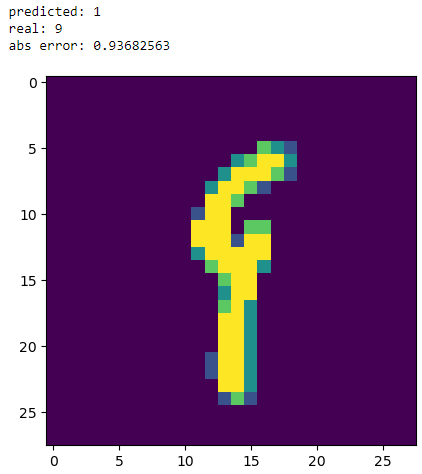
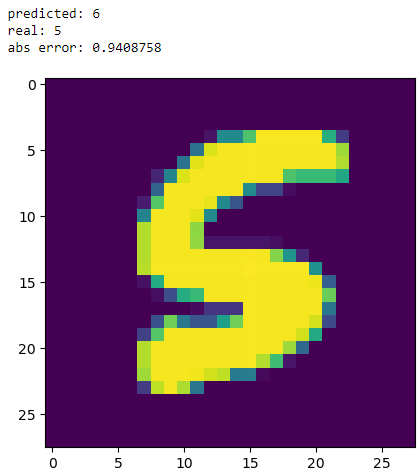
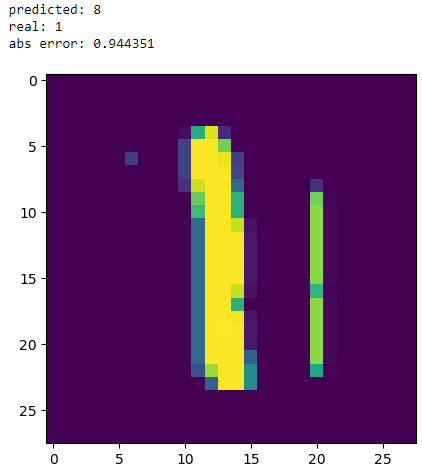
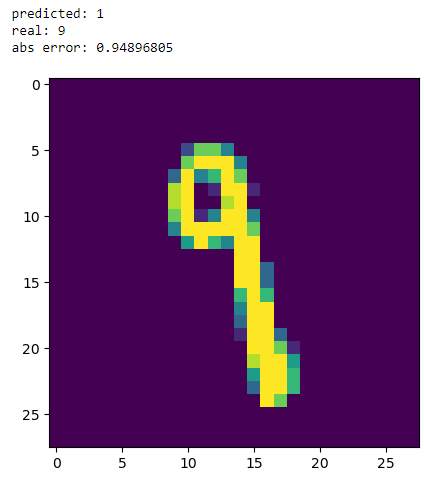
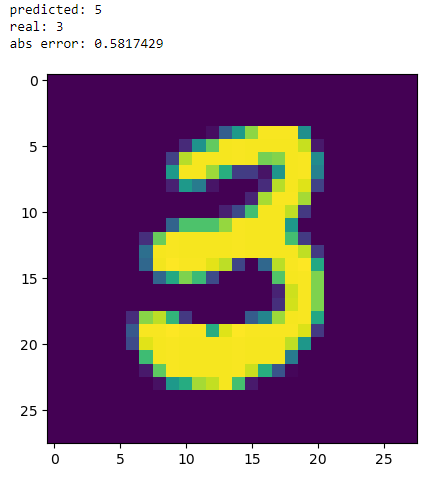
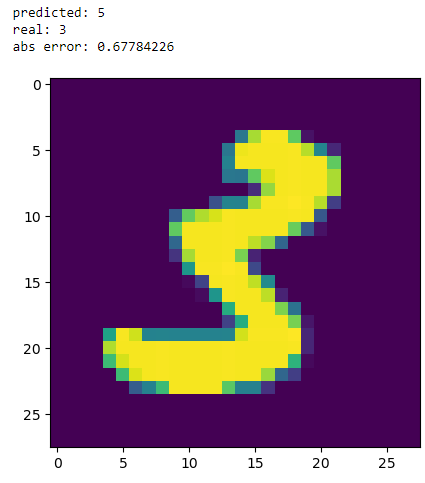
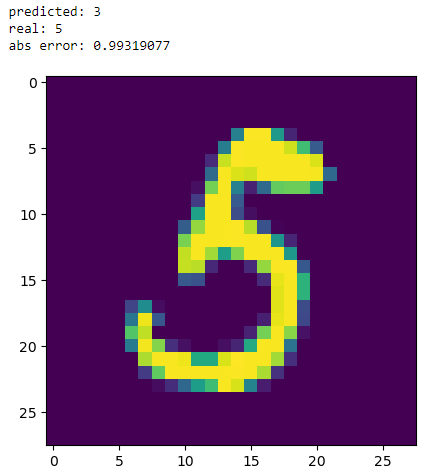
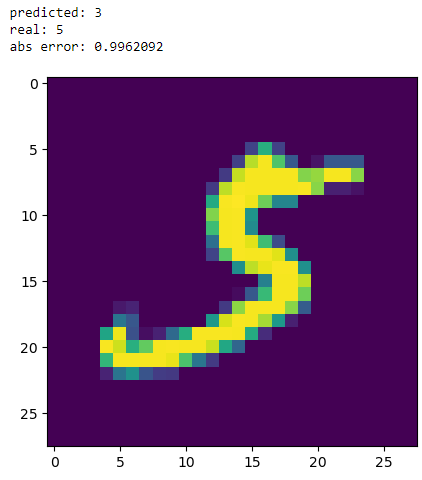
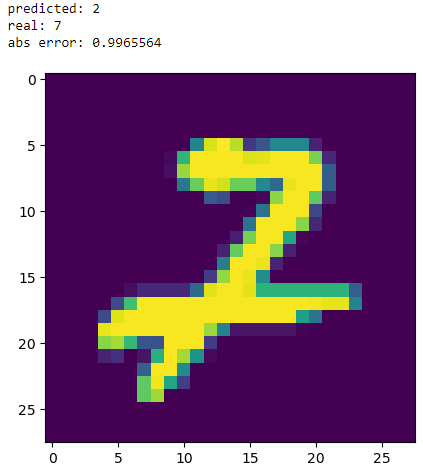
print(error[0])

print(error[1])

print(error[2])

plt.imshow(error[3])

plt.show()



1. Покращити архітектуру мережі відомими вам методами. При цьому внести мінімум змін в її архітектуру.

Додаємо nn.BatchNorm2d()

Зменшуємо learning\_rate з 0.001 до 0.002

Зменшуємо batch\_size з 500 до 50

1. Натренувати мережу до заданої точності. Вивести графіки тренувальної та валідаційної точностей та функції втрат відносно ітерацій/епох.

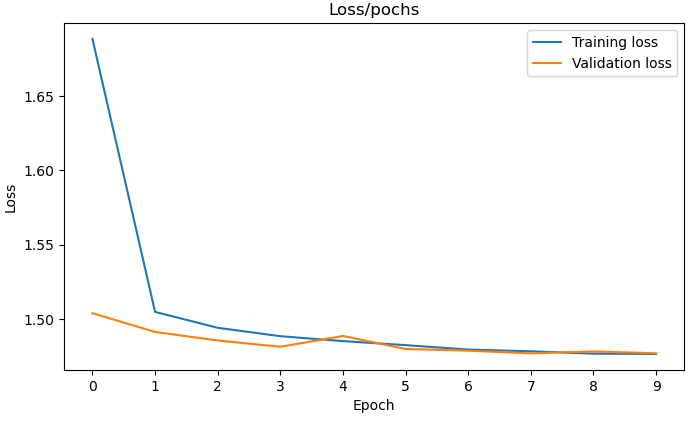
epoch 10

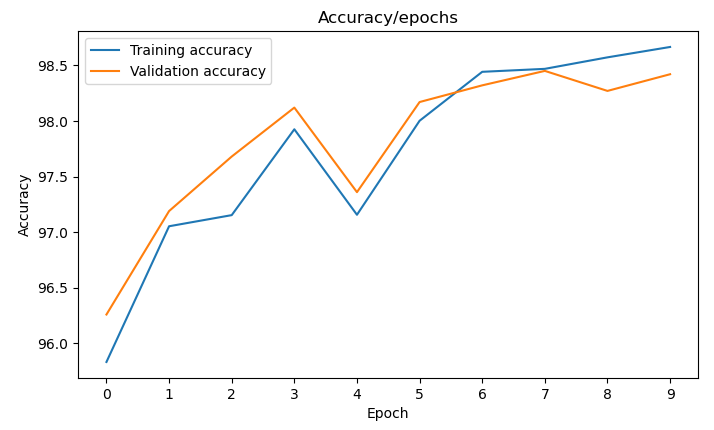
Training loss:1.4765

Training accuracy:98.66

Validation loss:1.4770

Validation accuracy:98.42





1. Звізуалізувати найгірші 10 передбачень(ті, в яких мережа найбільше помилилась відносно правильних відповідей).

