DL Exam

Full Name*	
Matriculation Number*	
Course of Studies*	

- You have 90 minutes to finish the exam.
- You are not allowed to use any electronic auxiliaries including calculators. If you have complex mathematical expressions, you may leave the fractions, logarithms, exponentials, etc. as is without having to calculate the exact numerical value.
- You are allowed to use exactly one DinA4 sheet of notes.
- The space below each question should be sufficient to write down your answer (more paper is available on demand).
- Please keep your handwriting legible and stick to the number of answers asked for. Illegible, ambiguous and multiple answers will be not graded.
  Use a permanent marker!
- Students who registered with "MeinCampus" can check their results after grading there. All others will be notified by the e-mail address linked with the StudOn course access.

☐ You can send me e-mails for upcoming events and open positions to the following e-mail address**:

I have read all the information above and entered required data truthfully:

Signature	

*This data is required to identify you for the grading process.
**This entry is optional and has no effect on the exam whatsoever. Only fill it in if you want to be put on a mailing list from our lab.

Question	1	2	3	4	5	6	7	Total
Points	8	16	12	9	13	4	4	60 + 6
Achieved								

1 Multiple Choice Questions (8P)

For each of the following questions, mark **all choices** that apply. Each question has AT LEAST one correct option unless otherwise stated. No explanation is required.

Question 1.1 1 P.

One technique that is commonly used in machine learning is data normalization. Which of the following statements is/are true?

- ☐ Data normalization is not helpful when using neural networks since they adapt to the data during training.
- ☐ Normalization should always be performed for each sample independently to ensure that the features are appropriately scaled.
- ☐ The test data should be normalized with statistics computed on the test data since the test distribution might be different from the training distribution.
- ☐ One common strategy for normalization is to normalize the data to have zero mean and unit variance.
- ☐ None of the above.

Solution 1.1

(iv)

Question 1.2 1 P.

You design the following fully connected network: All but the last activation function are ReLU activations.

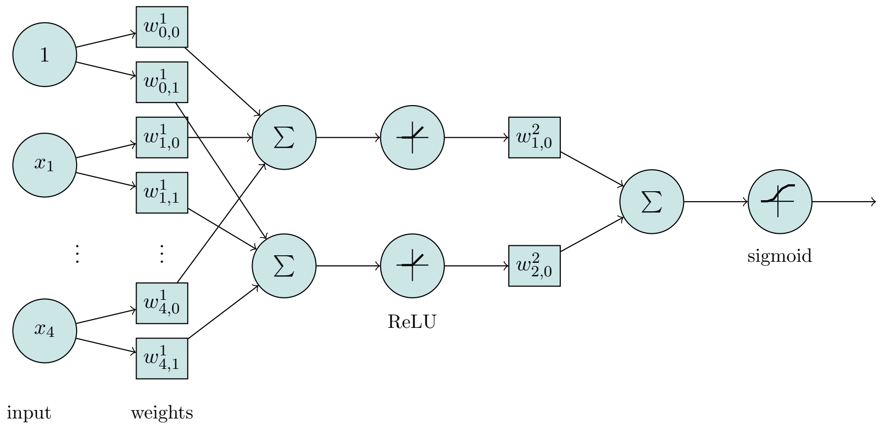


Figure 1: Related FC network

The last activation is a sigmoid function. You initialize all weights and biases to zero and forward an input $\mathbf{x} = (1, 1, 1, 1)$ though the network. What is the output of the network $\hat{y}$?

- ☐ -1
- ☐ 4
- ☐ 0.5

☐ 0

☐ None of the above.

Solution 1.2
(iii)

Question 1.3 1 P.
You receive a network implementation that has an unknown activation function X as the last layer of the network. To test the implementation, you feed an input to a neuron and you get the output -0.0001. Which of the following activation functions could X be?

☐ ReLU

☐ Tanh

☐ Leaky ReLU

☐ Sigmoid

☐ SoftMax

Solution 1.3
(ii), (iii)

Question 1.4 1 P.
Which of the following statements about batch normalization is/are true?

☐ Batch normalization layers are only relevant during training and can be skipped during inference.

☐ It is an effective way of backpropagation.

☐ It normalizes the weights of each layer.

☐ It smooths the loss landscape during training.

☐ None of the above.

Solution 1.4
(iv)

Question 1.5 1 P.
Which of the following features is/are NOT used in LeNet?

☐ Pooling layers

☐ Convolutional layers to extract spatial features

☐ Non-linearity using ReLU

☐ Use of skip connections

☐ All of the above are used in LeNet.

Solution 1.5
(iii), (iv)

Question 1.6 1 P.
Which of the following statements is/are TRUE about the vanishing gradient problem?

- ☐ Dropout can be used to combat the vanishing gradient problem.
- ☐ Recurrent neural networks are not affected by the vanishing gradient problem.
- ☐ Because of vanishing gradients, later layers (closer to the output) learn slower compared to earlier layers.
- ☐ Tanh functions are preferred compared to sigmoid functions because they do not suffer from vanishing gradients.
- ☐ None of the above.

Solution 1.6
(v)

Question 1.7 1 P.
Which of the following statements is/are TRUE about a convolutional layer?

- ☐ The total number of weights depends on the number of channels of the input.
- ☐ The total number of weights depends on the stride.
- ☐ It computes rotation invariant features.
- ☐ It can process inputs with arbitrary spatial size.
- ☐ None of the above.

Solution 1.7
(i), (iv)

Question 1.8 1 P.
You train your deep model multiple times and observe that different training runs provide very different predictions; however, the average performance across the different runs is quite high. This means that your model has

- ☐ high bias, high variance
- ☐ low bias, high variance
- ☐ high bias, low variance
- ☐ low bias, low variance
- ☐ None of the above.

Solution 1.8
(ii)

2 Short Answers (16P)

For each of the following questions, answer briefly in 1-2 sentences.

Question 2.1 2 P.

You develop a network for binary classification where the last layer is a ReLU activation function. To obtain predictions between 0 and 1, you add a sigmoid activation function and classify all samples ≥ 0.5 as "positive". Is this a good design choice? Explain your answer.

Solution 2.1

No, because the output of the ReLU is always ≥ 0 which means that the output of the sigmoid function is always ≥ 0.5, which means all samples are classified as "positive".

Question 2.2 2 P.

In reinforcement learning, what is the difference between exploitation and exploration?

Solution 2.2

Improvement (exploitation) of current (locally optimal) reward vs. testing of potentially suboptimal actions (exploration) that provide higher returns in the following.

Question 2.3 2 P.

You are tasked with developing a system that supports checking whether the number of people in a room is below the maximum capacity. Is this a task you would address with semantic segmentation or object detection? Explain your answer.

Solution 2.3
Object detection, detects person instances which allows for counting the number of people.

Question 2.4 2 P.
Assume that you have a dataset that consists of images of size (10, 10, 3) where the last dimension refers to the number of channels. Your colleagues propose two options:

- Flatten the images to a vector of length 300, followed by a fully connected layer with five output neurons.
- Use a convolutional layer with five filter kernels with spatial size of 10×10 and valid padding.

Which approach would you use? Explain why.

Solution 2.4
Both approaches are equivalent, optional: approach two avoids flattening; matrix-multiplication might be implemented more efficiently.

Question 2.5 3 P.
Assume that you have created a neural network with two fully connected layers, the last function being a sigmoid function. Upon closer inspection of your code, you see that no intermediate activation function is used. First, write down the function that this network represents, where $\mathbf{x}$ represents the input, $\hat{\mathbf{y}}$ represents the output, $\mathbf{W}_l$ and $\mathbf{b}_l$ are the weights and the bias for layer l, and $\sigma(\cdot)$ represents a piecewise sigmoid function. Next, show that this is equivalent to having a single-layer network with weights $\tilde{\mathbf{W}}$ and bias $\tilde{\mathbf{b}}$ by expressing them in terms of the original network parameters.

Solution 2.5
Expression:

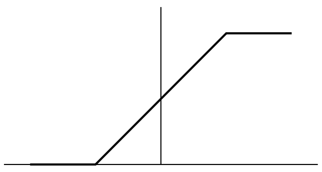
$$\hat{\mathbf{y}} = \sigma(\mathbf{W}_2(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2) \tag{1}$$

Single layer:

$$\begin{align} \hat{\mathbf{y}} &= \sigma(\mathbf{W}_2(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2) \tag{2} \\ \hat{\mathbf{y}} &= \sigma(\mathbf{W}_2\mathbf{W}_1\mathbf{x} + \mathbf{W}_2\mathbf{b}_1 + \mathbf{b}_2) \tag{3} \\ \tilde{\mathbf{W}} &= \mathbf{W}_2\mathbf{W}_1 \tag{4} \\ \tilde{\mathbf{b}} &= \mathbf{W}_2\mathbf{b}_1 + \mathbf{b}_2 \tag{5} \end{align}$$

Question 2.6 2 P.
A friend of yours suggests a new non-linear activation function that approximates the sigmoid function:

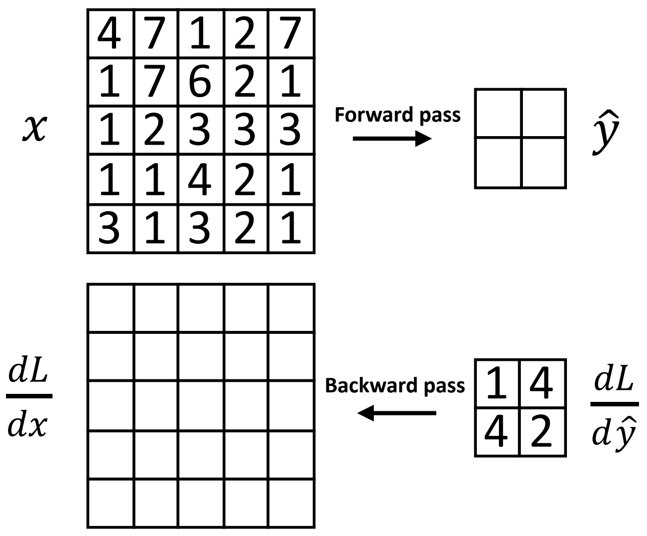
$$f(x) = \begin{cases} 0 & x < -0.5, \\ x & x \in [-0.5, 0.5], \\ 1 & x > 0.5 \end{cases} \tag{6}$$



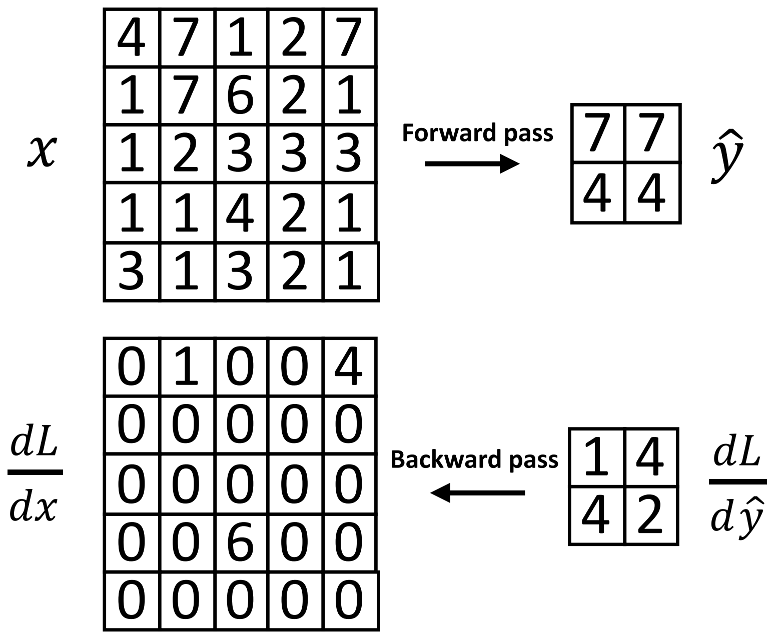
They explain that networks that use this activation function can be efficiently trained with stochastic gradient descent (SGD). Do you believe them? Explain why or why not.

Solution 2.6
No, gradient vanishes almost everywhere.

Question 2.7 3 P.
Given is a max pooling layer of kernel size 3 and stride 2 in "valid" mode. This layer is used to process some input $\mathbf{X}$ producing the output $\hat{\mathbf{y}}$. We expect that the pooling layer here behaves exactly as the one from the exercises. Furthermore, the gradient with respect to the input need to be computed. Fill in the corresponding values into the boxes below.



Solution 2.7
Alternative solutions are possible with the 7 in the top left kernel slide. Alternatively one can guide the Error 1 to the other 7 in that slide.



3 Perceptron and Common Practices (12P)

Question 3.1 4 P.

We are in the middle of a zombie apocalypse. We need to build a classifier to determine whether a patient is about to transform into a zombie. For this purpose, we have taken some blood samples containing the following concentrations of some interesting enzymes:

Sample	Braineatase(μl)	Prettysmelliase(μl)	Transformed into Zombie
Sample 1	42	0	yes
Sample 2	62	12	no
Sample 3	102	4	no
Sample 4	122	16	yes

In order to pre-process our data we want the distribution of each feature to have **zero mean** and a **maximum value of 1**. Please **normalize** the data and **enter** the normalized values here:

Sample	Braineatase	Prettysmelliase
Sample 1		
Sample 2		
Sample 3		
Sample 4		

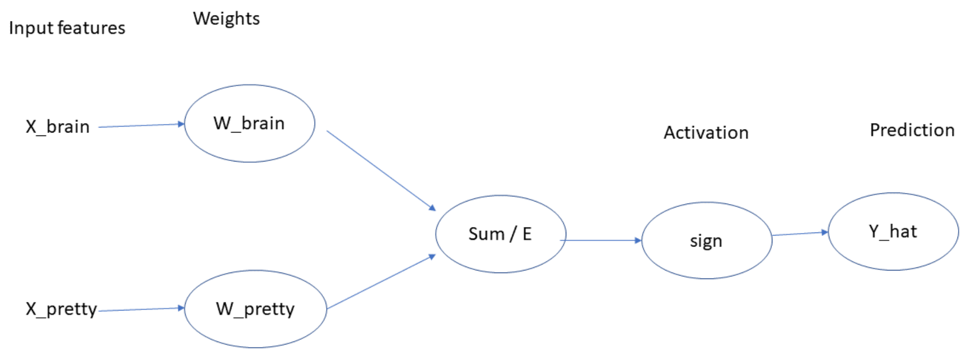
Solution 3.1

Sample	Braineatase	Prettysmelliase
Sample 1	-1	-1
Sample 2	-0.5	0.5
Sample 3	0.5	-0.5
Sample 4	1	1

Question 3.2 3 P.

We decide to use a Rosenblatt Perceptron as a classifier. You have trained your model using the normalized training data and obtained weights $w_{Brain} = 1$ and $w_{Pretty} = 3$ for the respective enzyme. Please **sketch** this kind of perceptron with given weights and **name** the components.

Solution 3.2



- 0.5 p inputs/output + naming
- 1p topology
- 1p activation function

Question 3.3 3 P.

You have taken a blood sample of yourself. You measure the following enzyme values: Braineatase - 22μl Prettysmelliase - 8 μl. According to the trained classifier defined above: will you transform into a zombie? Include calculations in your reasoning.

Solution 3.3

- Normalize: Braineatase = (22-82)/40 = -1.5; Prettysmelliase = (8-8)/8 = 0
- Calculate output: -1.5+0 = -1.5
- Apply signum function: -1 $\rightarrow$ You are not transforming into a zombie.

Question 3.4 2 P.

You have another look at the sample distribution. Was the choice of a single layer perceptron smart for the underlying data distribution? Briefly **explain**.

Hint: It may help to plot your training samples in a graph.

Solution 3.4

No it was not a good idea since the data distribution hints towards a XOR problem which cannot be solved by a single layer perceptron.

4 Backpropagation & Recurrent Neural Networks (9P)

Backpropagation is an important concept that enables training neural networks and it is also used in recurrent neural networks. Given is the recurrent cell visualized in Figure 2. In the following, the subscript (e.g., x_t) will denote the time step t.

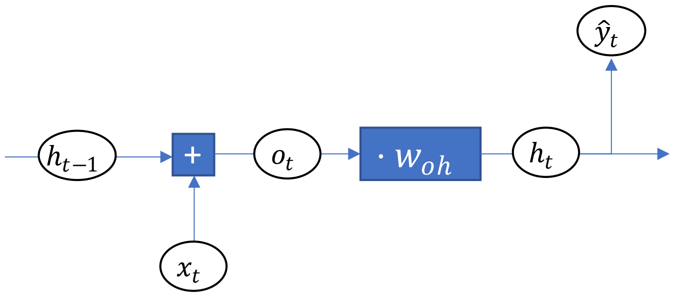


Figure 2: Schematic of a novel recurrent cell.

Question 4.1 2 P.

The network receives inputs $x_1 = 1$, $x_2 = 0.5$. The weights are initialized with $w_{oh} = 0.5$ and the hidden state at time point $t = 0$ is $h_0 = 0$. Calculate the intermediate states and the output for time points $t = 1$ and $t = 2$:

$$\begin{aligned} h_1 &= \\ h_2 &= \\ \hat{y}_1 &= \\ \hat{y}_2 &= \end{aligned}$$

Solution 4.1

$$\begin{aligned} h_1 &= 0.5 \\ h_2 &= 0.5 \\ \hat{y}_1 &= 0.5 \\ \hat{y}_2 &= 0.5 \end{aligned}$$

Question 4.2 5 P.

After the forward pass from these two time steps, you want to update the weights accordingly. The network is trained with the binary cross entropy function $L(y_t, \hat{y}_t) = -y_t \ln(\hat{y}_t) - (1 - y_t)\ln(1 - \hat{y}_t)$. Derive the partial derivatives in general as defined below.

$$\begin{aligned}
\frac{\partial L}{\partial \hat{y}_t} &= \\
\frac{\partial L}{\partial h_t} &= \\
\frac{\partial L}{\partial o_t} &= \\
\frac{\partial L}{\partial w_{oh_t}} &= \\
\frac{\partial L}{\partial w_{oh}} &=
\end{aligned}$$

Solution 4.2

$$\begin{aligned}
\frac{\partial L}{\partial \hat{y}_t} &= -\frac{y_t}{\hat{y}_t} + \frac{1-y_t}{1-\hat{y}_t} \\
\frac{\partial L}{\partial h_t} &= \frac{\partial L}{\partial \hat{y}_t} + \frac{\partial L}{\partial o_{t+1}} = \frac{\partial L}{\partial \hat{y}_t} + \frac{\partial L}{\partial \hat{h}_{t+1}} \cdot w_{oh} \\
\frac{\partial L}{\partial o_t} &= \frac{\partial L}{\partial \hat{h}_t} \cdot w_{oh} \\
\frac{\partial L}{\partial w_{oh_t}} &= \frac{\partial L}{\partial \hat{h}_t} \cdot (h_{t-1} + x_t) = \frac{\partial L}{\partial \hat{h}_t} \cdot o_t \\
\frac{\partial L}{\partial w_{oh}} &= \sum_t \frac{\partial L}{\partial w_{oh_t}}
\end{aligned}$$

Question 4.3 2 P.

What problem might occur with the setting presented above? What would be the remedy to solve it?

Solution 4.3

Due to the logarithm, the Cross-Entropy cannot handle values outside of [0, 1]. A sigmoid layer after the recurrent (or inside after the copy node) would squash the output values into the desired range. Furthermore, the chance of exploding or vanishing gradients for long sequences is high due to a missing non-linearity inside the cell.

5 Unsupervised Learning and Generative Adversarial Networks (11 + 2P)

Question 5.1 4 P.

Explain the general setup and training of a generative adversarial networks (GAN) consisting of a generator and a discriminator (including respective inputs and outputs).

Solution 5.1

Generator creates samples based on random noise, discriminator either receives real examples from a training set or artificial samples generated by the generator and needs to classify whether they are real or artificial. Both are trained using a min-max cost function, i.e., the Discriminator is trained to maximize the classification performance whereas the generator is trained to minimize the performance of the discriminator.

Question 5.2 3 P.

You are tasked with developing a GAN to generate images of different natural environments, specifically, forest, lake, sea side, mountains, and grassland. The goal is that users can select an environment and then a set of artificial images showing the chosen environment is presented to them. What GAN type do you have to use here and how does it work? What does your training data need to contain to be able to train this kind of model?

Solution 5.2

Conditional GAN, allows to specify specifics of the desired output. Additional input to both generator and discriminator that encodes the target. Training set has to consist of images that show each environment (+labels which image contains what).

Question 5.3 2 P.

After having trained your GAN for approx. 100 epochs, you check the losses for the generator and the discriminator. You observe that the loss values are very similar compared to the beginning of your training. Do you have to assume that the networks have not learned anything and that the generated images look very similar compared to the beginning? Briefly explain why or why not.

Solution 5.3

No, it is not unlikely that both the generator and the discriminator have improved but just equally well.

Question 5.4 2 P.

You have trained your system and now want to evaluate it with some test users. After seeing a few images, they complain that all look very similar. Explain what could have happened and name a strategy to combat this effect.

Solution 5.4

Mode collapse, Strategies: Mini-batch discrimination or unrolled GANs

Question 5.5 2 P.

Bonus question:

Instead of generating completely new images, you also want to provide a system where users can smartly in-paint certain regions of their already existing images. For that, they can erase certain parts of the image that are then replaced by a neural network. While you can opt for a GAN as well, you can also use a different approach that allows to use your existing training data very efficiently. Name that approach and briefly explain how you would train this network.

Solution 5.5
(Denoising) Autoencoder, can erase arbitrary parts of the image and then let the autoencoder repaint it to the original image. Loss function e.g., MSE or L1 loss.

6 Coding (4P)

Question 6.1 4 P.

You are given the framework of the exercises 1, 2 and 3. Implement the LeakyReLU layer using only numpy and basic python operations. Your layer must contain a constructor **__init__**, receiving a parameter **alpha** defining the slope for negative inputs, a method **forward(input_tensor)** and a method **backward(error_tensor)**. Note: The exact recall of numpy functions is not required to pass the task. You may instead "define" a suitable function, in this case document this accordingly.

```python
import numpy as np

class LeakyReLU:
    def __init__(self, alpha):
    # insert your code here





    def forward(self, input_tensor):
    # insert your code here





    def backward(self, error_tensor):
    # insert your code here




# end of file
```

Solution: (multiple solutions are possible)

```
import numpy as np
class LeakyReLU:
    def __init__(self, alpha): # +0.5
        # insert your code here
        self.alpha = alpha



    def forward(self, input_tensor):
        # insert your code here
        self.input_tensor = input_tensor
        self.pos = (input_tensor >= 0)  # +1P
        self.neg = (input_tensor < 0) * self.alpha # +0.5P

        return input_tensor * self.pos + input_tensor * self.neg

        # alternative:
        return input_tensor * self.pos + (1-self.pos) * self.alpha *
           input_tensor  # +1P

    def backward(self, error_tensor):
        # insert your code here
        return error_tensor * self.pos + error_tensor * self.neg # +1P
```

7 Bonusquestion: PyTorch (4P)

Question 7.1

4 P.

You want to execute the following network implemented with PyTorch on your graphics card. Look for the **four** errors and **draw** a circle around to mark them.

```
from torch import nn
from torch.nn import functional as F
class Model(nn.Module):
    def __init__(self):
        super(Model, self).__init__()
        self.conv1 = nn.Conv2d(in_channels=1,
                               out_channels=100,
                               kernel_size=5).cuda()
        self.conv2 = nn.Conv2d(in_channels=10,
                                out_channels=50,
                                kernel_size=5).cuda()
        self.conv3 = nn.Conv2d(in_channels=50,
                                out_channels=5,
                                kernel_size=5).cuda()
        self.fc1 = nn.Linear(in_features=5 * 256,
                             out_features=50).cuda()
    def forward(self, x):
        print(x.shape)
        # >> (10, 1, 28, 28)
        y = x.cpu()
        y = F.relu(self.conv1(y))
        y = F.relu(self.conv2(y))
        y = F.relu(self.conv3(y))
        y = F.relu(self.fc1(y.flatten(start_dim = 1)))
        y = x + y
        return self.fc2(y)
```

Solution:

```
from torch import nn
from torch.nn import functional as F
class Model(nn.Module):
    def __init__(self):
        super(Model, self).__init__()
        self.conv1 = nn.Conv2d(in_channels=1,
                               out_channels=100,
                               kernel_size=5).cuda()
        self.conv2 = nn.Conv2d(in_channels='10', # here
                                out_channels=50,
                                kernel_size=5).cuda()
        self.conv3 = nn.Conv2d(in_channels=50,
                                out_channels=5,
                                kernel_size=5).cuda()
        self.fc1 = nn.Linear(in_features=5 * 256,
                             out_features=50).cuda()
    def forward(self, x):
        print(x.shape)
        # >> (10, 1, 28, 28)
        y = x'.cpu()' # here
        y = F.relu(self.conv1(y))
        y = F.relu(self.conv2(y))
        y = F.relu(self.conv3(y))
        y = F.relu(self.fc1(y.flatten(start_dim = 1)))
        y = 'x + y' # here
        return self.'fc2(y)'  # here
```

Additional space for solutions and calculations