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. (Back and front handwritten)
- 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 visited the Deep Learning exercise in the following semester ***:	
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.

*** This addresses in particular students who did the exercise in a previous semester. We want to ensure bonus points are transferred correctly from previous semesters.

Question	1	2	3	4	5	6	7	Exercise Bonus	Total
Points	8	10	7	12	9.5	6	7.5	(6)	60
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. To get the points for one question, all ticks must be set correctly.

Question 1.1 1 P.

What is the correct order of steps to train a neural network using the backpropagation algorithm and an optimizer like Stochastic Gradient Descent (SGD)?

(A) compute the difference between the estimation and prediction

(B) iterate until the weights are optimized

(C) forward the input into the network, get the output

(D) randomly initiate the weights and biases

(E) tune the weights and biases of each neuron to minimize the loss

☐ D → B → C → A → E

☐ D → C → A → E → B

☐ D → A → C → E → B

☐ D → E → A → B → C

☐ D → B → E → C → A

Solution 1.1

2

Question 1.2 1 P.

Which of the following statements are **true** regarding the Batch Normalization layer?

☐ It normalizes the distribution of the input for the layer which follows the Batch normalization layer

☐ It normalizes the weights of each layer

☐ It is an effective way of back-propagation

☐ It can normalize the complete training dataset by computing the global mean and variance

☐ It has trainable parameters

Solution 1.2
1, 5

Question 1.3 1 P.
Suppose you have a model trained on the ImageNet data set for a classification task. Then you feed the model with a blank image where every pixel is the same white color. For this input, the network will output the same score for each class. This statement is

- ☐ True
- ☐ Cannot be answered
- ☐ False

Solution 1.3
3

Question 1.4 1 P.
Which of the following functions do/does **not** fulfill the requirements for an activation function to train a Deep Learning model?

- ☐ $tanh(x)$
- ☐ $\frac{1}{x}$
- ☐ $2x$
- ☐ $max(x, 0)$
- ☐ $\frac{1}{1+e^{-x}}$

Solution 1.4
2,3

Question 1.5 1 P.
Which of the following statement(s) is/are **wrong**?

- ☐ The goal of a good policy is to maximize the future return
- ☐ The Bellman equations form a system of linear equations which can be solved for small problems
- ☐ Q Learning is an Off-policy method that does not need information about dynamics of the environment
- ☐ Temporal Difference Learning is an On-policy method that needs information about dynamics of the environment

☐ Greedy action selection policy is not always the best policy

Solution 1.5
4

Question 1.6 1 P.
What statement(s) about the model capacity is/are **true**?

☐ The model capacity is determined by the number of training samples the network was optimized with

☐ The model capacity is linked to the variety of functions which can be approximated

☐ The model capacity is influenced by the depth of a network

☐ The tradeoff between bias and variance is closely related to the model capacity

☐ The bias in prediction (bias-variance tradeoff) necessarily increases when the model capacity increases as well

Solution 1.6
2,3,4

Question 1.7 1 P.
Which statement(s) is/are **true** about initializing your weights and biases

☐ If the bias is initialized with 0, the gradient of the loss with respect to the bias is 0 as well

☐ Initializing your bias with 0 helps with the dying ReLU problem

☐ It is important to calibrate the variance of your weights

☐ Initialization does matter because the optimization of a Deep Learning model is a convex problem

Solution 1.7
3

Question 1.8 1 P.

Which of the following statement is/are **true**?

- ☐ The goal of segmentation is to draw a bounding box which contains the desired object
- ☐ Semantic segmentation can be considered as a pixel-wise classification
- ☐ A pixel-wise segmentation of an object can be converted into a bounding box
- ☐ Instance segmentation can only find one instance of a class per image

Solution 1.8

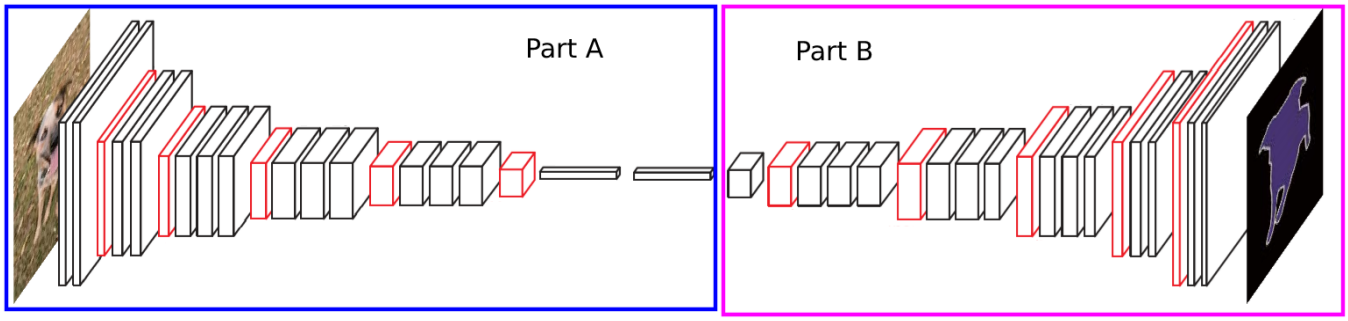
2,3

2 Short Answers (10P)

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

Question 2.1 1 P.

Given the following network



What is the name for part A and part B?

Solution 2.1

part A: Encoder part B: Decoder

Question 2.2 1 P.

U-Net, the most popular networks for segmentation, and Autoencoders typically consist of part A and part B. What is the main difference between Autoencoders and U-Nets?

Solution 2.2

U-Net has skip-connections from Encoder to Decoder part / from lower to higher layers.

Question 2.3 2 P.

Given is an Elmann RNN layer with a fully connected layer as hidden gate and another fully connected layer as output gate. Why is it incorrect to update the weights of the hidden gate, say using any optimizer, **while** you iterate backwards through the time series using the TBPTT algorithm (1P)? Which solution was introduced in the lecture/exercise to overcome this problem (1P)?

Solution 2.3

Problem: the Elman cell shares the same weights for all times steps. If weights change during TBPTT, forward and backward would not match anymore, thus the derivatives would not match anymore. Basically you change the function while you derive it (1P). The remedy is to sum up the gradient for the weights and update once the backward pass is done (1p).

Question 2.4 2 P.

There are multiple ways to evaluate the performance of an image segmentation with k classes. Here, p_{ij} is the number of pixels with ground truth label i which are predicted as class j. Given are the following formulas

$$\frac{\sum_{i=0}^{k} p_{ii}}{\sum_{i=0}^{k}\sum_{j=0}^{k} p_{ij}} \tag{1}$$

$$\frac{1}{\sum_{i=0}^{k}\sum_{j=0}^{k} p_{ij}} \sum_{i=0}^{k} \frac{\sum_{j=0}^{k} p_{ij} p_{ii}}{\sum_{j=0}^{k} p_{ij} + \sum_{j=0}^{k} p_{ji} - p_{ii}} \tag{2}$$

$$\frac{1}{k+1} \sum_{i=0}^{k} \frac{p_{ii}}{\sum_{j=0}^{k} p_{ij}} \tag{3}$$

$$\frac{1}{k+1} \sum_{i=0}^{k} \frac{p_{ii}}{\sum_{j=0}^{k} p_{ij} + \sum_{j=0}^{k} p_{ji} - p_{ii}} \tag{4}$$

A: Mean Class-wise Pixel Accuracy B: Mean Intersection over Union
C: Pixel Accuracy D: Frequency Weighted Intersection over Union

Which are the correct names of the following evaluation metrics?

(1) _____ (2) _____ (3) _____ (4) _____

Solution 2.4

1: C, 2:D, 3:A,, 4:B

Question 2.5 1 P.

A convolutional layer with stride (larger than 1) can be implemented using a convolutional layer of stride 1 followed by subsampling. Describe the special actions required in the backward pass to backpropagate through the subsampling process. (You can use the term ErrorTensor as replacement for the partial derivative of the loss with respect to the strided convolutional layer's output)

Solution 2.5

Error tensor must be padded with 0s (1P) at the columns and rows which were thrown out in the forward pass due to striding.

Question 2.6 2 P.

Why is YOLO called a "Single-Shot-Detector" (SSD)?

Solution 2.6

SSDs combine the prediction of the bounding box and the classification of each box into one model and one forward pass call.

Question 2.7 1 P.

You train a Conditional Generative Adversarial Network (Conditional GAN) to generate images of cats and dogs. You train the Conditional GAN using mini-batch gradient descent. The classes are balanced. During training, the images are ordered in a way such that all the dog images are trained first followed by all cat images. Your colleague suggests that "you absolutely need to shuffle your training set before the training procedure." Is your colleague right? Explain.

Solution 2.7

Yes, there is a problem. The optimization is much harder with minibatch gradient descent because the loss function moves by a lot and hence the generator's output when going from the one type of image to another.

3 Basics (7P)

Alice and Bob work on a mask-wearing classification project. They want to classify whether a person is wearing a face-covering mask (e.g. FFP2 to prevent COVID spread) or not, given an image of that person. Bob wants to train a deep convolutional neural network (CNN) f on a dataset with 1000 samples. Given an input image showing the person's face $\mathbf{x} \in \mathbb{R}^{HxW}$ with a height H and a width W, his model predicts $\hat{y} = f(\mathbf{x})$, with $\hat{y} \in \mathbb{R}$ being between 0 and 1. $y \in \{0, 1\}$ defines the label where 1 means "wears mask" and 0 means "wears no mask".

Question 3.1 2 P.

Which last activation function should be used in f to provide the prediction according to the definition of $\hat{y}$? Explain why.

Solution 3.1

It must be sigmoid as the output must represent values between 0 and 1 and as the output is required to be a scalar value. Softmax does not work here as y is a scalar.

Question 3.2 2 P.

Write down the formula of a suitable loss function for this task assuming a batch size of 1. Express the loss using y and $f(\mathbf{x})$

Solution 3.2

Binary Cross Entropy OR Cross Entropy on two classes

$$L = -y \ln(f(\mathbf{x})) - (1 - y) \ln(1 - f(\mathbf{x})) \tag{5}$$

Question 3.3 1 P.

With the data Bob and Alice acquired in a shopping mall, they achieve the following confusion matrix. Compute the accuracy.

	$y = 1$	$y = 0$
$\hat{y} = 1$	980	18
$\hat{y} = 0$	1	1

Solution 3.3

$\text{ACC} = \frac{980+1}{980+18+1+1} = \frac{981}{1000} = 98.1\%$

Question 3.4 2 P.

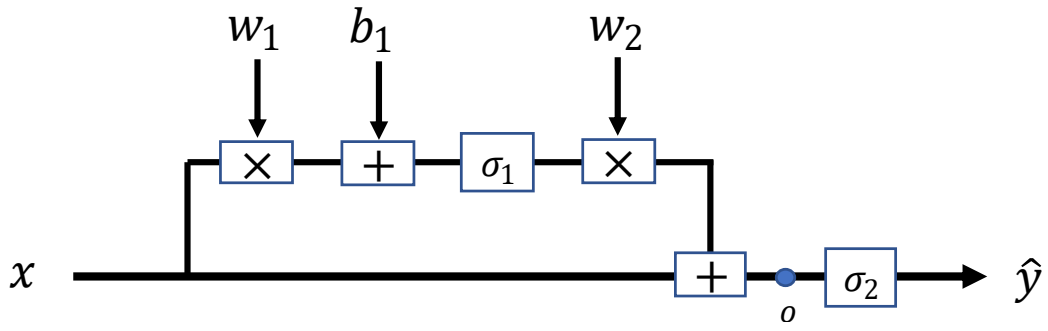
Bob claims they solved the problem given the performance achieved in Question 3.3. Alice doesn't seem to agree with Bob. Explain what doubts Alice might have and propose two solutions what can be done to solve the issue.

Solution 3.4

Highly imbalanced classes (1P). A remedy might be oversampling/undersampling (0.5P), loss weighting (0.5P) and not acquiring data in a shopping mall (somewhere else).

4 Backpropagation (12 points)

Given is the following network $f(x) = \hat{y}$ receiving $x \in \mathbb{R}$ as input to compute a prediction $\hat{y} \in \mathbb{R}$. It uses two weights $w_1, w_2 \in \mathbb{R}$, one bias $b_1 \in \mathbb{R}$ and two sigmoid activations denoted as functions $\sigma_1(\cdot)$ and $\sigma_2(\cdot)$ shown in the figure below. The network is trained using the L_2 norm, defined as $L(y, \hat{y}) = \|\hat{y} - y\|_2^2$ with labels $y \in \{0, 1\}$. The boxes are mathematical operations, while $\times$ represents the multiplication, $+$ the addition and σ the sigmoid activation. $o \in \mathbb{R}$ marks an intermediate result as indicated in the figure.



Question 4.1 1 P.

Name the architecture presented in the lecture, to which the upper network has similarities to?

Solution 4.1

This sequence has similarities to the ResNet module and hence to the ResNet architecture.

Question 4.2 1 P.

Express the network f, as function of $x, w_1, w_2, b_1, \sigma_1, \sigma_2$.

$$f(x) = \hat{y} =$$

Solution 4.2

$$f(x) = \hat{y} = \sigma_2(x + w_2\sigma_1(b_1 + w_1 x))$$

Question 4.3 1 P.

Assume you have an input sample $x = 1.0$ with associated label $y = 1$, the weights $w_1 = 0.5$, $w_2 = -2$ and the bias $b_1 = -0.5$. Compute the loss for these numbers.

Solution 4.3

$$f(1.0) = \sigma_2(1.0 + -2\sigma_1(-0.5 + 0.5 \cdot 1.0)) = 0.5$$
$$L(1, 0.5) = \|0.5 - 1\|_2^2 = 0.25$$

Question 4.4 7 P.

Derive the partial derivatives for the network f listed below. Note: In this question you should **not** insert the values of Question 4.3, but instead derive the general formulas. If necessary, define auxiliary components, which may simplify your answers.

$$\frac{\partial L}{\partial \hat{y}} =$$

$$\frac{\partial L}{\partial o} =$$

$$\frac{\partial L}{\partial w_2} =$$

$$\frac{\partial L}{\partial b_1} =$$

$$\frac{\partial L}{\partial w_1} =$$

$$\frac{\partial L}{\partial x} =$$

Solution 4.4

Let $\sigma'(x)$ be the derivative of $\sigma(x)$, which are defined as

$$\sigma(x) = \frac{1}{1+e^{-x}}$$
$$\sigma'(x) = \sigma(x)(1-\sigma(x))$$

The solution is

$$\begin{aligned}
\frac{\partial L}{\partial \hat{y}} &= 2(\hat{y}-y) \\
\frac{\partial L}{\partial o} &= \frac{\partial L}{\partial \hat{y}}\sigma_2'(x + w_2\sigma_1(w_1 x + b_1)) \\
\frac{\partial L}{\partial w_2} &= \frac{\partial L}{\partial o}\sigma_1(w_1 x + b_1) \\
\frac{\partial L}{\partial b_1} &= \frac{\partial L}{\partial o}w_2\sigma_1'(w_1 x + b_1) \\
\frac{\partial L}{\partial w_1} &= \frac{\partial L}{\partial o}w_2\sigma_1'(w_1 x + b_1)x \\
\frac{\partial L}{\partial x} &= \frac{\partial L}{\partial o} + \frac{\partial L}{\partial o}w_2\sigma_1'(w_1 x + b_1)w_1
\end{aligned}$$

Question 4.5 2 P.

Assume you have trained your network f and now want to compute an adversarial example of x given the label $y = 0$. Write down a loss function called L_{adv} for this task, and explicitly state the variable which is optimized. You are not allowed to use any images other than x for this optimization.

$$L_{adv} =$$

Solution 4.5

$$L_{adv} = \|f(x) - y\|_2^2 = \|f(x)\|_2^2$$

and x is being optimized. Regularization terms as in DeepDream can be optionally added as well.

5 Regularization and Visualization (up to 12 Points)

You are participating in the student Deep Learning challenge of your university. The task is to classify whether a person is smiling in portrait images, but you have only little training data available. You choose a network with 3 convolutional layers together with ReLU activation and one fully connected layer at the end. You observe that you have good performance on your training data but poor performance on your validation data, so your tutor recommends using regularization techniques.

Question 5.1 1 P.

What is the general purpose of regularization?

Solution 5.1

Solution: Bias/variance trade-off, counter overfitting, generalizing on unseen data, enforcing prior

Question 5.2 1.5 P.

First, you want to try regularizing your loss function by enforcing a norm on your weights. You experiment with two different norms in separate training runs. When visualizing a **histogram of your weights** for the first layer you can see the effect of the norm. Name which norm was used in which training and shortly explain your decision.

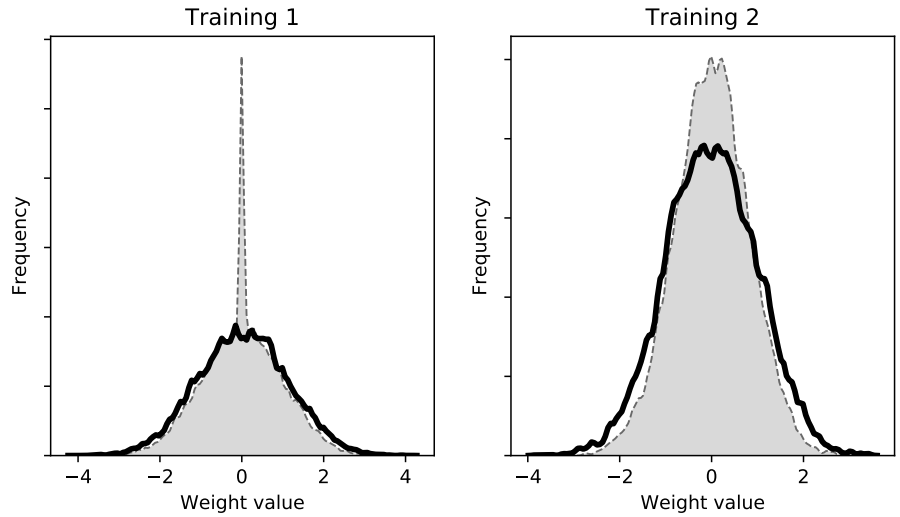


Figure 1: Weight distribution. The thick black line shows the original distribution (without regularization), the shaded grey area is the distribution with the norm regularization.

Solution 5.2
0.5P L1 left
0.5P L2 right
0.5P explanation: peak for L1 at 0 or smaller variance for L2

Question 5.3 4 P.

You continue to visualize parts of your network. When looking at the **histogram of your activation values** of one batch, you realize that the mean and the variance deviate further from 0 and 1 after each layer. Since you already normalized the data in the beginning you wonder about the problem. Name the problem and which layer-type you could add to counter this effect. Explain what this type of layer does and how it needs to behave differently during training and testing.

Solution 5.3

1P Problem: Internal covariate shift OR not zero centered RELUs
1P Layer: Batch Normalization (maybe also self normalzing SELU)
1P. explanation center mean and standardize variance (formula is fine)
0.5 P moving average for training, 0.5P use training values for testing

Question 5.4 2 P.

To make sure that your convolutional neural network for smiling detection really focuses on meaningful image parts (e.g. the mouth) you want to use saliency maps to check that. Explain the process of generating saliency maps and especially which value can be used as a pseudo loss.

Solution 5.4

1 P Classification score as pseudo loss
1 P Backpropagate to beginning of network an visualize gradient

Question 5.5 1 P.

When analyzing your saliency maps you see that the network is not really focusing on the face. You think that training your network additionally on similar classification tasks could help the main smiling classification task. How do you call this method and what would be a suitable additional task to help the

training?

Solution 5.5

0.5 P Auxiliary tasks / Multi-task learning
0.5 P Additional task: example: gender classification, pose estimation, detect glasses, head detection, open/closed eyes...

6 Coding (6P)

Question 6.1 6 P.

You are given the framework of the exercises 1, 2 and 3. Implement the ELU layer presented in the lecture using only numpy and basic python operations. ELU is defined as

$$\mathrm{ELU}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{else} \end{cases}$$

Your layer must contain a constructor **__init__**, receiving the parameter **alpha** defining the factor 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 instead.

```
import numpy as np

class ELU:
    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)

```
# forward tested with nn.ELU
# gradient tested numerically with framework
import numpy as np
class ELU:
    def __init__(self, alpha):
        self.alpha = alpha # +0.5

    def forward(self, input_tensor):
        # +1 for treating neg and pos side differently
        # +1 applying correct formula
        # +1 storing information required in backward
        self.input_tensor = input_tensor
        self.pos = (input_tensor > 0)
        self.neg = (input_tensor <= 0)
        return input_tensor * self.pos + (np.exp(input_tensor) - 1) *
           self.neg * self.alpha

    def backward(self, error_tensor):
        # +0.5 for treating neg and pos side differently
        # +1 applying correct derivative
        # +1 considered error_tensor right
        return error_tensor * self.pos + \
               error_tensor * self.neg * self.alpha * np.exp(self.
                  input_tensor)
```

7 PyTorch (7.5P)

You want to implement the following architecture in PyTorch. It consists of multiple convolutional layers with different strides and padding (specified in the Model constructor below), each followed by a ReLU activation.
Remark: All convolutional layers in PyTorch are per default in "valid" mode, that means no padding at the borders is applied (padding value of 0). If the padding value is larger than 0, padding is applied before the convolution in height and width dimension on both sides of the image. The padding value (in the constructor) defines the padding width of one side of the image.

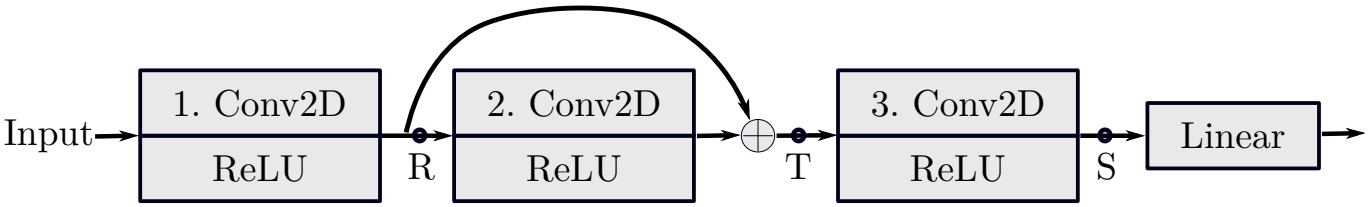


Figure 2: CNN Architecture

```
import torch
from torch import nn
from torch.nn.functional import relu

class Model(nn.Module):
    def __init__(self, input_channels, hidden_channels, num_classes):
        super(Model, self).__init__()
        self.convA = nn.Conv2d(in_channels=hidden_channels,
                               out_channels=2*hidden_channels,
                               kernel_size=5,padding=2,stride=3).cuda()
        self.convB = nn.Conv2d(in_channels=hidden_channels,
                               out_channels=hidden_channels,
                               kernel_size=3, padding=1).cuda()
        self.convC = nn.Conv2d(in_channels=input_channels,
                               out_channels=hidden_channels,
                               kernel_size=3).cuda()
        self.fc = nn.Linear(in_features=800,
                            out_features=num_classes).cuda()
```

Question 7.1 4.5 P.

To check the correct handling of the image sizes in your network, fill in the following tables. First, find the correct order of the convolution objects defined in the constructor. Then, enter the shape of the weights for the above defined convolutions (bias can be neglected) and the size of your images at the locations **R**, **T**, **S** for the defined input size. The weights should be defined as **(number of kernels, channel-dimension, x-dimension, y-dimension)** while the image sizes should have the shape **(batch-size, number of channels, x-dimension, y-dimension)**.
`in_channels` is set to 3 and `hidden_channels` is set to 4.

convolution operation	variable name	weight shape
1. Conv2D		
2. Conv2D		
3. Conv2D		

result at position	image shape
Input	(10, 3, 32, 32)
R	
T	
S	

Solution 7.1

convolution operation	variable name	weight shape
1. Conv2D	convC	(4, 3, 3, 3)
2. Conv2D	convB	(4, 4, 3, 3)
3. Conv2D	convA	(8, 4, 5, 5)

result at position	image shape
Input	(10, 3, 32, 32)
R	(10, 4, 30, 30)
T	(10, 4, 30, 30)
S	(10, 8, 10, 10)

Question 7.2 3 P.

Implement the forward pass for the given architecture in python using the PyTorch library. The input x is located on the CPU.

```
    def forward(self, x):
        print(x.shape)  # >> (10, 3, 32, 32)









        return y
```

Solution:

```
import torch
from torch import nn
from torch.nn.functional import relu

class Model(nn.Module):
    def __init__(self, input_channels, hidden_channels, num_classes):
        super(Model, self).__init__()
        self.convA = nn.Conv2d(in_channels=hidden_channels,
                               out_channels=2*hidden_channels,
                               kernel_size=5,padding=2,stride=3).cuda()
        self.convB = nn.Conv2d(in_channels=hidden_channels,
                               out_channels=hidden_channels,
                               kernel_size=3, padding=1).cuda()
        self.convC = nn.Conv2d(in_channels=input_channels,
                               out_channels=hidden_channels,
                               kernel_size=3).cuda()
        self.fc = nn.Linear(in_features=800,
                            out_features=num_classes).cuda()

    def forward(self, x):
        print(x.shape)  # >> (10, 3, 32, 32)
        x = x.cuda()
        x = relu(self.convC(x))
        y = relu(self.convB(x))
        y = y + x
        y = relu(self.convA(y))
        y = torch.flatten(y, 1)  # y.reshape(x.shape[0], -1)
        return self.fc(y)
```

Additional space for solutions and calculations