Problem 1 Multiple Choice (18 credits)

Below you can see how you can answer multiple choice questions.

Mark correct answers with a cross ☒
To undo a cross, completely fill out the answer option ■
To re-mark an option, use a human-readable marking ✕■

- For all multiple choice questions any number of answers, i.e. either zero (!), one or multiple answers can be correct.
- For each question, you'll receive 2 points if all boxes are answered correctly (i.e. correct answers are checked, wrong answers are not checked) and 0 otherwise.

1.1 Which of the following models are unsupervised learning methods?

- [x] Auto-Encoder
- [ ] Maximum Likelihood Estimate
- [x] K-means Clustering
- [ ] Linear regression

1.2 In which cases would you usually reduce the learning rate when training a neural network?

- [x] When the training loss stops decreasing
- [ ] To reduce memory consumption
- [ ] After increasing the mini-batch size
- [x] After reducing the mini-batch size

1.3 Which techniques will typically decrease your **training** loss?

- [x] Add additional training data
- [x] Remove data augmentation
- [x] Add batch normalization
- [ ] Add dropout

1.4 Which techniques will typically decrease your **validation** loss?

- [x] Add dropout
- [x] Add additional training data
- [ ] Remove data augmentation
- [ ] Use *ReLU* activations instead of *LeakyReLU*

DO NOT SCAN/UPLOAD

$\theta = \theta - \alpha \cdot \nabla L$

1.5 Which of the following are affected by multiplying the loss function by a constant positive value when using SGD?

- [x] Memory consumption during training
- [x] Magnitude of the gradient step
- [ ] Location of minima
- [x] Number of mini-batches per epoch

1.6 Which of the following functions are not suitable as activation functions to add non-linearity to a network?

- [ ] $\sin(x)$
- [x] $ReLU(x) - ReLU(-x)$
- [ ] $\log(ReLU(x) + 1)$
- [x] $\log(ReLU(x + 1))$

1.7 Which of the following introduce non-linearity in the neural network?

- [x] LeakyReLU with $\alpha = 0$
- [ ] Convolution
- [x] Batch Norm
- [ ] Skip connection

1.8 Compared to the L1 loss, the L2 loss...

- [x] is robust to outliers
- [x] is costly to compute
- [x] has a different optimum
- [ ] will lead to sparser solutions

1.9 Which of the following datasets are NOT i.i.d. (independent and identically distributed)?

- [ ] A sequence (toss number, result) of 10,000 coin flips using biased coins with p(toss result = 1) = 0.7
- [x] A set of (image, label) pairs where each image is a frame in a video and each label indicates whether that frame contains humans.
- [x] A monthly sample of Munich's population over the past 100 years
- [ ] A set of (image, number) pairs where each image is a chest X-ray of a different human and each number represents the volume of their lungs.

DO NOT SCAN/UPLOAD

Problem 2 Short Questions (29 credits)

0 1 2 3

2.1 Explain the idea of data augmentation (1p). Specify 4 different data augmentation techniques you can apply on a dataset of RGB images (2p).

DO NOT SCAN/UPLOAD

0 1 2

2.2 You are training a deep neural network for the task of binary classification using the Binary Cross Entropy loss. What is the expected loss value for the first mini-batch with batch size $N = 64$ for an untrained, randomly initialized network? Hint: $BCE = -\frac{1}{N}\sum y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)$

DO NOT SCAN/UPLOAD

0 1 2

2.3 Explain the differences between *ReLU*, *LeakyReLU* and *Parametric ReLU*.

DO NOT SCAN/UPLOAD

0 1 2

2.4 How will weights be initialized by Xavier initialization? Which mean and variance will the weights have? Which mean and variance will the output data have?

DO NOT SCAN/UPLOAD

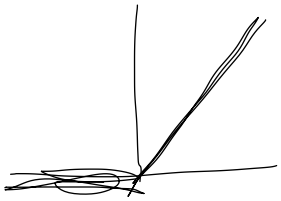
DO NOT SCAN/UPLOAD

2.1 Use different method to process the train data and add this processed data to the origin data

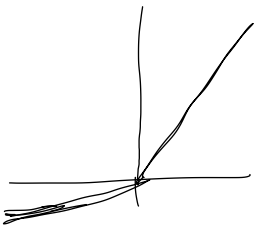
Filping / Cropping / Gaussian blur / Sharpen / Rotation /

2.2

2.3 ReLU Leaky ReLU Parameter ReLU



$\max(0, x)$



$\max(0.1x, x)$

$\max(\alpha x, x)$

2.4 $var(w) = \frac{1}{n}$ $mean = 0$

Xavier Initialization ensure the variance of outputs are same as input

2.5 Why do we often refer to L2-regularization as “weight decay”? Derive a mathematical expression that includes the weights W, the learning rate η, and the L2 regularization hyperparameter λ to explain your point.

☐ 0
☐ 1
☐ 2
☐ 3

2.6 Given a Convolution Layer in a network with 6 filters, kernel size 5, a stride of 3, and a padding of 2. For an input feature map of shape $28 \times 28 \times 28$, what are the dimensions/shape of the output tensor after applying the Convolution Layer to the input?

☐ 0
☐ 1
☐ 2

2.7 You are given a Convolutional Layer with: number of input channels 3, number of filters 5, kernel size 4, stride 2, padding 1. What is the total number of trainable parameters for this layer? Don’t forget to consider the bias.

☐ 0
☐ 1
☐ 2

2.8 You are given a fully-connected network with 2 hidden layers, the first of has 10 neurons, and the second hidden layer contains 5 neurons. Both layers use dropout with probability 0.5. The network classifies gray-scale images of size 8×8 pixels as one of 3 different classes. All neurons include a bias. Calculate the total number of trainable parameters in this network.

☐ 0
☐ 1
☐ 2

DO NOT SCAN/UPLOAD

2.5 Wrong name, but weights are pushed to be zero

L2 Regularization with (0.1)

$$w^+ = w - \eta \nabla (L(w) + \frac{1}{5}\lambda \|w\|_2^2)$$

$$w^+ = w - \eta \nabla L - \eta\lambda w$$

$$w^+ = (1 - \eta\lambda) w - \eta \nabla L.$$

$\eta\lambda \ll 1$ ∴ Each iteration push w to 0

2.6. $\frac{F+2p-N}{S} + 1$

$= \frac{28+2\times2-5}{3} + 1$

$= 10$

$10 \times 10 \times 6$

2.7 (3, 5, 4, 4)

$(3 \times 4 \times 4 + 1) \times 5$

$= 245$

2.8 64.

8×8 [10] 0.5 [5] 0.5 [0 0 3 0]

$64 \times 10 + 10 + 10 \times 5 + 5 + 5 \times 3 + 3$

$= 650 + 55 + 18$

$= 720$ 723

0 1 2

2.9 *"Breaking the symmetry"*: Why is initializing all weights of a fully-connected layer to the same value problematic?

0 1 2

2.10 Explain the difference between *Auto-Encoders* and *Variational Auto-Encoders*.

0 1 2

2.11 Generative Adversarial Networks (GANs): What is the input to the generator network (1 pt)? What are the two inputs to the discriminator (1 pt)?

0 1 2

2.12 Explain how LSTM networks often outperform traditional RNNs. What in their architecture enables this?

0 1 2

2.13 Explain how batch normalization is applied differently between a fully connected layer and a convolutional layer (1 pt). How many learnable parameters does batch normalization contain following (a) a single fully-connected layer (1 pt), and (b) a single convolutional layer with 16 filters (1 pt)?

0 1 2 3

2.9 Because all neuron have same weight
→ cause same gradients when Back propogations
→ learn same things
→ NN update very slow or not optimal

2.10 Autoencode: encoder encode the input feature to a low dimension space. decoder draw from this space and use learned feature to generate output

VAE encode encode the input feature to a latent space with is Gaussian distributed, decode sample the vector from the gaussian latent space to generate output

---

2.11 Real world Image / Random variable from latent space and sample and sample from generator

2.12 Cell-wise for Information transform and flowing for gradient.

Also use tanh and sigmoid as Activate layer.

Also have three gates to update information or cover

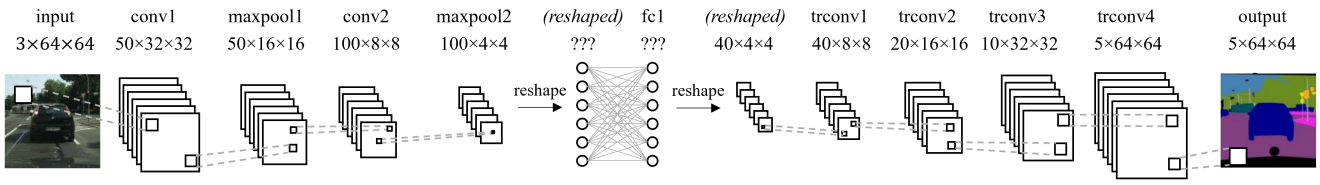
2.13 Mini-Batch is normalized for all neuron in FC

2 β and γ While Mini-Batch is normalized for all channels in conv Layer

2 for ~~conv~~ FC and 2 × 16 for conv

Problem 3 Convolutions (13 credits)

You are asked to perform **per-pixel** semantic segmentation on the Cityscapes dataset, which consists of RGB images of European city streets, and you want to segment the images into 5 classes (vehicle, road, sky, nature, other). You have designed the following network, as seen in the illustration below:
For clarification of notation: The shape **after** having applied the operation 'conv1' (the first convolutional layer in the network) is 50x32x32.
You are using 2D convolutions with: `stride` = 2 , `padding` = 1 , and `kernel_size` = 4 .
For the MaxPool operation, you are using: `stride` = 2 , `padding` = 0 , and `kernel_size` = 2 .



3.1 What is the shape of the weight matrix of the fully-connected layer 'fc1'? (Ignore the bias)

0 1 2

3.2 Explain the term 'receptive field' (1p). What is the receptive field of one pixel of the activation map. after performing the operation 'maxpool1'(1p)? What is the receptive field of a single neuron in the output of layer 'fc1' (1p)?

0 1 2 3

DO NOT SCAN/UPLOAD

100 x 4 x 4 = 1600 Input

40 x 4 x 4 = 640 Output

weight sharing 16 x 640

How many pixels in the input image can affect only one pixel on the output image

$\frac{F-N}{S}+1 = 2$

$\frac{F-N}{S}+1 = 1$, $F = 2$ 2x2

$\frac{F-2}{2} = 0$

$\frac{F+2P-N}{S}+1 = 2$

$P = +2 - 4 = 2$

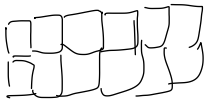
$F = 4 \times 4$

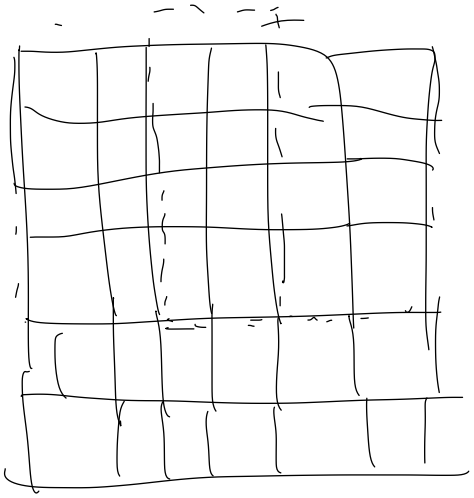
$I = 6 \times 6$

Whole Images, how all connected (64 x 64)

$\frac{4+2-4}{2}+1 = 2$

$(2+3 \times 2) \times 2$





0 ☐
1 ☐

3.3 You now want to be able to classify finer-grained labels, which comprise of 30 classes. What is the **minimal** change in network architecture needed in order to support this without adding any additional layers?

0 ☐
1 ☐
2 ☐

3.4 Luckily, you found a pre-trained version of this network, which is trained on the original 5 labels. (It outputs a tensor of shape $5 \times 64 \times 64$). How can you make use of/build upon this pre-trained network (as a black-box) to perform segmentation into 30 classes.

0 ☐
1 ☐
2 ☐
3 ☐

3.5 Luckily, you have gained access to a large dataset of city street images. Unfortunately, these images are not labelled, and you do not have the resources to annotate them. However, how can you still make use of these images to improve your network? Explain the architecture of any networks that you will use and explain how training will be performed. (Note: This question is independent of (3.3) and (3.4))

0 ☐
1 ☐
2 ☐

3.6 Instead of taking 64×64 images as input, you now want to be able to train the network to segment images of arbitrary size > 64. List, explicitly, two different approaches that would allow this. Your new network should support varying image sizes in run-time, without having to be re-trained.

3.3 Changing the last layer into FC

3.4 Use pretrained network and frozen the weight
only train the last FC Layer

3.5 Use Autoencoder to learn the feature the feature from
Unlabeled Images
And use pretrained encoder and classifier

3.6 Add a conv both ~~of~~ first layer to deal the size
problem

Problem 4 Optimization (13 credits)

4.1 Explain the idea behind the RMSProp optimizer. How does it enable faster convergence than standard SGD? How does it make use of the gradient?

☐ 0
☐ 1
☐ 2
☐ 3

4.2 What is the *bias correction* in the ADAM optimizer? Explain the problem that it fixes.

☐ 0
☐ 1
☐ 2

softwares

4.3 You read that when training deeper networks, you may suffer from the *vanishing gradients* problem. Explain what are vanishing gradients in the context of deep convolutional networks and the underlying cause of the problem.

☐ 0
☐ 1
☐ 2
☐ 3

DO NOT SCAN/UPLOAD

① use second Momentum to predict the next optimal location and jump to it then make some correction, Speed up the optimal process

RMSprop is an adaptive learning rate method It scales the learning rate enable fast convergence by skipping the saddle point Dampening the oscillation in the dim with high variance

Second Momentum is a expoentially weighted moving average of square gradients

---

When accumulating gradient in weighted moving average fasion, the first gradient is initialized to 0, this biases all the accumulating gradient toward zero.

Solution: Bias correction normalized the accumulating the gradient in early step. magnitude of

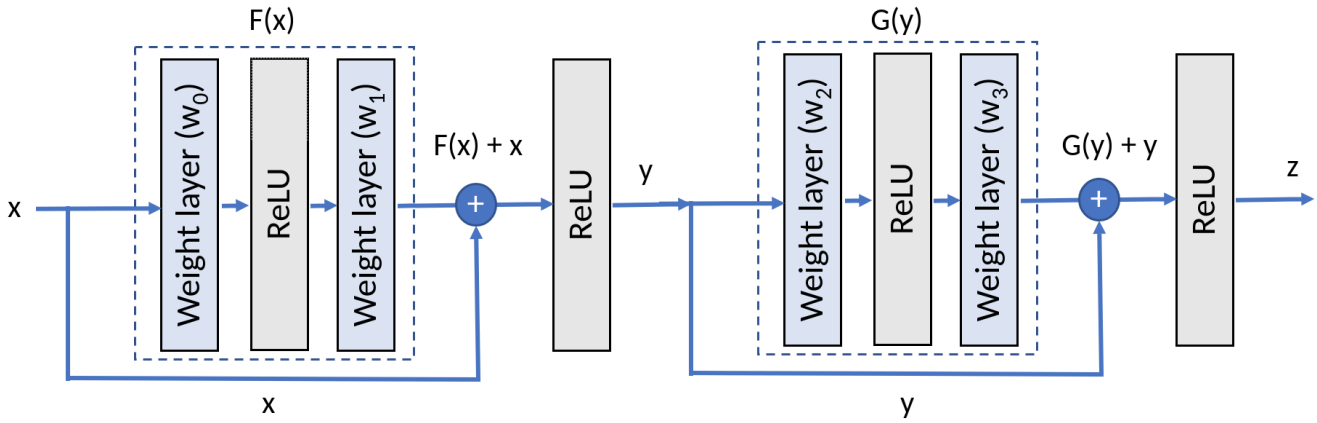
---

When the network is very deep, i.e. with alot of layers, the gradient of deep layer are almost equal to zero, this situation will speed down the optimizing process, which called gradient vanishing

- Activation layer saturates eg tanh sgn
- Chain rule, the weights from early layer depend on last layer which also equal to 0

0 1 2 3 4 5

4.4 In the following image you can see a segment of a very deep architecture that uses residual connections. How are residual connections helpful against vanishing gradients? Demonstrate this mathematically by performing a weight update for w_0. Make sure to explain how this reduces the effect of vanishing gradients. Hint: Write the mathematical expression for $\frac{\partial z}{\partial w_0}$ w.r.t all other weights.



DO NOT SCAN/UPLOAD

The Residual connection provide a highway for gradient, which can efficiently solve the gradient vanishing in deep network

Residual connection.

$z = ReLU(z')$

$z' = G(y) + y$

$G(y) = w_3 \cdot ReLU(w_2 \cdot y)$

$y = ReLU(y')$

$y' = F(x) + x$

$F(x) = w_1 \cdot ReLU(w_0 \cdot x)$

$$\frac{\partial z}{\partial w_0} = \frac{\partial ReLU(G(y)+y)}{\partial (G(y)+y)} \cdot \frac{\partial G(y)+y}{\partial y} \cdot \frac{\partial ReLU(F(x)+x)}{\partial F(x)+x} \cdot \frac{\partial F(x)+x}{\partial w_0}$$

$$= 1 \cdot \left(\frac{\partial w_3 \cdot ReLU(w_2 \cdot y)}{\partial ReLU(w_2 \cdot y)} \cdot \frac{\partial ReLU(w_2 \cdot y)}{\partial w_2 \cdot y} \cdot \frac{\partial w_2 \cdot y}{\partial y} + 1 \right) \cdot 1 \cdot \frac{\partial w_1 \cdot ReLU(w_0 \cdot x)}{\partial ReLU(w_0 \cdot x)} \cdot \frac{\partial ReLU(w_0 \cdot x)}{\partial w_0 \cdot x} \cdot \frac{\partial w_0 \cdot x}{\partial w_0}$$

$$= 1 \left(w_3 \cdot 1 \cdot w_2 + 1 \right) \left(w_1 \cdot 1 \cdot x \right)$$

$$= w_1 \cdot w_2 \cdot w_3 \cdot x + w_1 \cdot x$$

Compare to the gradient without skipping

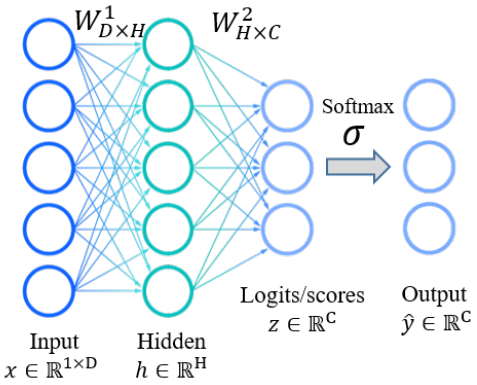
there exists addition "$w_1 x$" to prevent gradient vanish

e.g. $w_3 \to 0$

Problem 5 Multi-Class Classification (18 credits)

Note: If you cannot solve a sub-question and need its answer for a calculation in following sub-questions, mark it as such and use a symbolic placeholder (i.e., the mathematical expression you could not explicitly calculate + a note that it is missing from the previous question.)

Assume you are given a labeled dataset $\{X, y\}$, where each sample x_i belongs to one of $C = 10$ classes. We denote its corresponding label $y_i \in \{1, ..., 10\}$. In addition, you can assume each data sample is a row vector.
You are asked to train a classifier for this classification task, namely, a 2-layer fully-connected network. For a visualization of the setting, refer to the following illustration:



5.1 Why does one use a **Softmax** activation at the end of such a classification network? What property does it have that makes it a common choice for a classification task?

0
1
2

5.2 For a vector of logits $\vec{z}$, the Softmax function $\sigma : \mathbb{R}^C \to \mathbb{R}^C$, is defined:

0
1

$$\hat{y}_i = \sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}$$

where C is the number of classes and z_i is the i-th logit.
A special property of this function is that its derivative can be expressed in terms of the Softmax function itself. How could this be advantageous for training neural networks?

DO NOT SCAN/UPLOAD

Softmax activation is to calculate the probability of every output term

↖ the goal

Last Layer

Softmax can compute the probability distribution of the output over the different classes

Softmax function ensure all the out put is valid probability and sum up is equal to 1

This is eay to interepret the output as the predicted classes

---

In forward pass, you must calculate the probability distribution, in the same time you can cache it so that you can use it as the derivative when Back propagation, it sum up take from previous

0 1 2 3

5.3 Show explicitly how this can be done, by writing $\frac{\partial \hat{y}_i}{\partial z_i}$ in terms of $\hat{y}_i$.

0 1 2

5.4 Similarly, show explicitly how this can be done, by writing $\frac{\partial \hat{y}_i}{\partial z_j}$ in terms of $\hat{y}_i$ and $\hat{y}_j$, for $i \neq j$.

DO NOT SCAN/UPLOAD

$i = j$

$$\frac{\partial \hat{y}_i}{\partial z_i} = \frac{\partial \sigma(\vec{z})_i}{\partial z_i} = \frac{e^{z_i} \cdot \sum_{j=1}^{C} e^{z_j} - e^{z_i} \cdot e^{z_i}}{\left(\sum_{j=1}^{C} e^{z_j}\right)^2}$$

$$= \frac{e^{z_i}\left(\sum_{j=1}^{C} e^{z_j} - e^{z_i}\right)}{\sum_{j=1}^{C} e^{z_j} \cdot \sum_{j=1}^{C} e^{z_j}}$$

$$= \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}} \cdot \left(1 - \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}\right)$$

$$= \hat{y}_i (1 - \hat{y}_i)$$

$i \neq j$

$$\frac{\partial \hat{y}_i}{\partial z_j} = \frac{\partial \sigma(\vec{z}_i)}{\partial z_j} = \frac{0 \cdot \sum_{j=1}^{C} e^{z_j} - e^{z_i} \cdot e^{z_j}}{\left(\sum_{j=1}^{C} e^{z_j}\right)\left(\sum_{j=1}^{C} e^{z_j}\right)}$$

$$= -(\hat{y}_i)(\hat{y}_j)$$

$$= -\hat{y}_i \hat{y}_j$$

5.5 Using the Softmax activation, what loss function $\mathcal{L}(y, \hat{y})$ would you want to *minimize*, to train a network on such a multi-class classification task? Name this loss function (1 pt), and write down its formula (2 pt), for a single sample x, in terms of the network's prediction $\hat{y}$ and its true label y. Here, you can assume the label $y \in \{0,1\}^C$ is a one-hot encoded vector:

$$y_i = \begin{cases} 1, & \text{if } i == \text{true class index} \\ 0, & \text{otherwise} \end{cases}$$

☐ 0
☐ 1
☐ 2
☐ 3

5.6 Having done a forward pass with our sample x, we will back-propagate through the network. We want to perform a gradient update for the weight $w^2_{j,k}$ (the weight which is in row j, column k of the second weights' matrix W^2). First, use the chain rule to write down the derivative $\frac{\partial \mathcal{L}}{\partial w_{j,k}}$ as a product of 3 partial derivatives (no need to compute them). For convenience, you can ignore the bias and omit the ^{2} superscript.

☐ 0
☐ 1
☐ 2

DO NOT SCAN/UPLOAD

Cross Entropy

$$L = -\sum_{j=1}^{C} y_j \cdot \ln \hat{y}_j$$

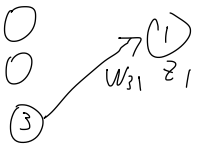
$$\frac{\partial L}{\partial w_{j,k}} = \frac{\partial L}{\partial \hat{y}_j} \cdot \frac{\partial \hat{y}_j}{\partial z_j} \cdot \frac{\partial z_j}{\partial w_{j,k}}$$

$$\frac{\partial L}{\partial w_{j,k}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial w_{j,k}}$$

0 1 2 3 4 5

5.7 Now, compute the gradient for the weight: $w^2_{3,1}$. For this, you will need to compute each of the partial derivatives you have written above, and perform the multiplication to get the final answer. You can assume the ground-truth label for the sample was `true_class` = 3 . **Hint:** The derivative of the logarithm is $(\log t)' = \frac{1}{t}$.

DO NOT SCAN/UPLOAD



$$\frac{\partial L}{\partial w_{j,k}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial w_{j,k}}$$

$$= -y_3 \cdot \frac{1}{\hat{y}_3} \cdot -\hat{y}_3 \cdot \hat{y}_1 \cdot h_3$$

$$= -1 \cdot 1 \cdot -1 \cdot \hat{y}_1 \cdot h_3$$

$$= \hat{y}_1 \cdot h_3$$

Additional space for solutions–clearly mark the (sub)problem your answers are related to and strike out invalid solutions.

DO NOT SCAN/UPLOAD

DO NOT SCAN/UPLOAD