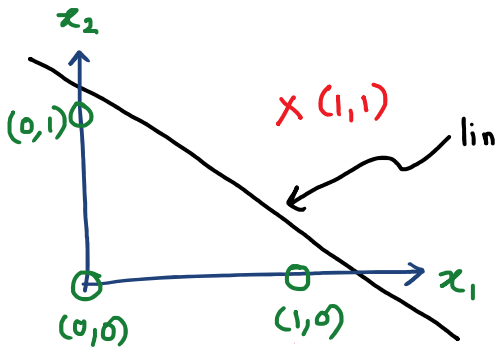
QUESTION 1 [2pts]: Can a NAND gate (truth table below) be separated by a linear classifier? Explain.

x_1	x_2	y
0	0	1
0	1	1
1	0	1
1	1	0

Soln:



∴ NAND is separable

Marking scheme: 1 pt for separable Yes/No

1 pt for explaining with diagram

QUESTION 2 [2pts]: A **validation set** is used to tune hyperparameters of a neural network model.

(a) However, if we tuned the hyperparameters using the training set, then what could have gone wrong with such a model?

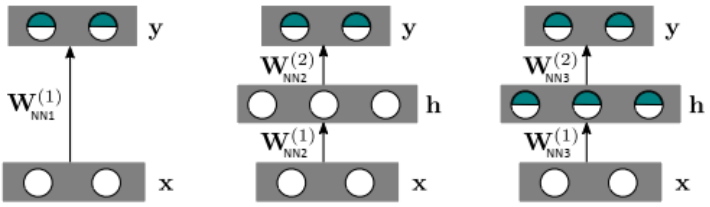
(b) Why should we not train the hyperparameters on a test set?

Soln

(a) It will select hyperparameters that will give the best results for the training set → would lead to overfitting (1)

(b) It will not give a realistic measure of generalization, as the reason for creating the test was to get gauge generalization error on unseen data. (1)

QUESTION 3 [2pts]: Consider the following three fully connected feedforward neural networks:



(a) NN1 (b) NN2 (c) NN3

- **NN1**: No hidden layer; directly maps the inputs to outputs
- **NN2**: Has a linear hidden layer (meaning $\mathbf{h} = \mathbf{W}^{(1)}_{\text{NN2}}\,\mathbf{x}$ at hidden layer)
- **NN3**: Has a non-linear hidden layer (meaning $\mathbf{h} = \sigma\left(\mathbf{W}^{(1)}_{\text{NN3}}\,\mathbf{x}\right)$ at hidden layer)

You may assume same nonlinear activation functions at the outputs and zero bias terms.
Choose the correct relation for the expressive power of the three neural networks. Explain in a few words.

(A) NN1 > NN2 > NN3

(B) NN3 > NN2 > NN1

✓(C) NN3 > NN2 = NN1

(D) NN3 = NN2 > NN1

Soln: Correct answer (C) (1)

Explanation: Adding linear hidden layers doesnot add complexity or capacity of a neural network (1)

NN1 → $y = \sigma(W_{NN1}\, x)$ ← Similar

NN2 → $y = \sigma(W^{(2)}_{NN2}\, W^{(1)}_{NN2}\, x) = \sigma(W_{NN2}\, x)$

NN3 → $y = \sigma(W^{(2)}_{NN3}\, \sigma(W^{(1)}_{NN3}\, x))$

QUESTION 4 [2pts]: A difficulty with logistic sigmoid activation function is that of saturated units.

(a) Comment on if switching from logistic sigmoid to `tanh` fixes the problem of saturated units or not.

(b) Explain in 1-2 sentences on how switching to `tanh` activation helps in optimization?

Soln: (a) No, because for both sigmoid & tanh, the tail ends saturate (1)

(b) tanh is zero-centered (unlike sigmoid), which naturally provide some normalization and hence is helpful in optimization (½) (½)

QUESTION 5 [2pts]: In gradient descent optimization of neural networks, the learning rate is an important parameter. Briefly describe a problem encountered in optimization if we choose the learning rate too be (a) too high, and (b) too low for full batch gradient descent.

Soln: (a) Too high : Unstable (loss blows up), weights diverge, or the weight values oscillate a lot. (V2)

(b) Too low : Weight values change very slowly (1)

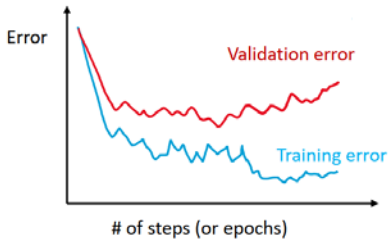
QUESTION 6 [1pt]: While training a neural net using stochastic gradient descent (SGD), you notice that after an SGD update, the training loss function has increased.

Claim: The reason must be that the learning rate is too high.

Is the claim TRUE or FALSE? Justify your answer

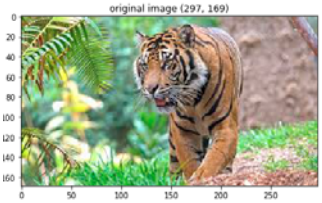
Soln: False! (V2)

While the training loss can increase when the learning rate is too high, in SGD, it can also increase due to fluctuations which happen due to noisy gradients (V2)



QUESTION 7 [1pt]: Consider that we use data augmentation to add different transformations of the original training set data to the training set. In this context, give an example of transformation that would be useful for classifying tigers vs. lions but not for classifying handwritten digits? Briefly explain your answer in 1-2 lines.

Soln: Horizontal flipping (V2)



still a tiger

2 → totally different (V2)

QUESTION 8 [3pts]: Consider a CNN with following architecture:

- **input**: RGB image of size 256×256
- **conv layer**: 64 convolution kernels of size 5×5 with equal stride $S = 2$, equal padding $P = 1$
- **maxpool layer**: 3×3 window with equal stride $S = 2$ with no padding

(a) Determine the output size and number of parameters of the conv layer

(b) Determine the output size and number of parameters of the maxpool layer

(c) What is the size of the receptive field for a single unit in the pooling layer (i.e., determine the width × height of the region in the input RGB image which influences the activation of a single unit in the pooling layer)?

Solu: (a) conv layer $\left\lfloor \frac{256-5+2(1)}{2} + 1 \right\rfloor = \left\lfloor \frac{253}{2} + 1 \right\rfloor = \lfloor 127.5 \rfloor = 127$

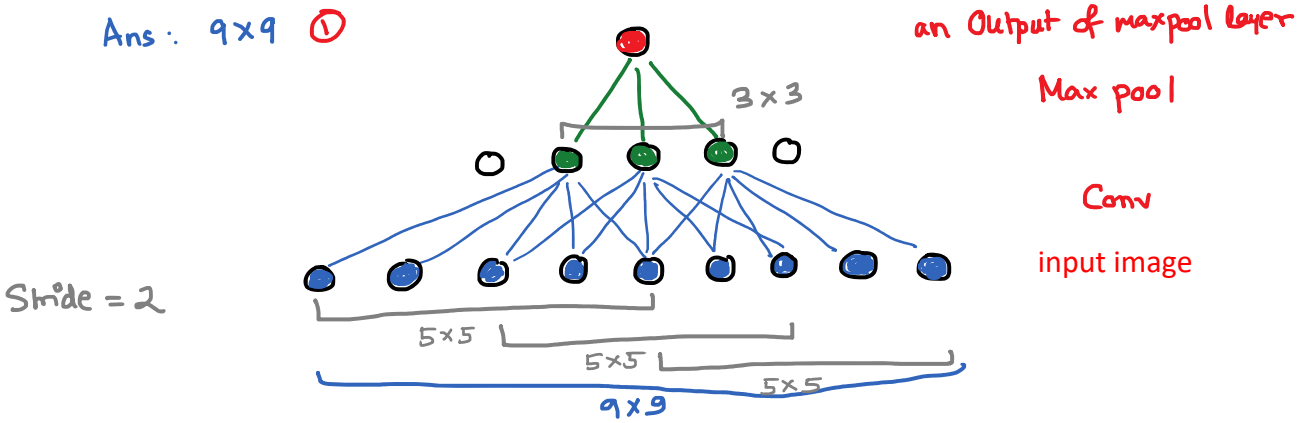
$127 \times 127 \times 64$ (1/2) # parameters $= 3 \times 5 \times 5 \times 64$ (1/2)

(b) maxpool layer $\left\lfloor \frac{127-3+2(0)}{2} + 1 \right\rfloor = \left\lfloor \frac{124}{2} + 1 \right\rfloor = \lfloor 63 \rfloor = 63$

$63 \times 63 \times 64$ (1/2) # parameters $= 0$ (1/2)

(c) Since the width and height are symmetric, we can look at just one dimension

Ans: 9×9 (1)



QUESTION 9 [2pts]: Which of the following statements is/are **False**? (Multiple answers may be correct)

(A) Larger batch-sizes reduces the fluctuations in mini-batch gradient descent

✓(B) Batch normalization ensures that the weights of each of the hidden layer of a deep neural network is normalized

(C) CNNs are less sensitive to spatial translation of objects within an image than fully connected NNs

✓(D) CNNs are prone to overfitting because of less number of parameters

(E) Weight sharing in CNN means a kernel is used throughout multiple locations of the whole input image

(F) 1×1 convolutions used in Inception modules of GoogLeNet decreases the feature depth but preserves width and height

(G) Residual blocks are used in ResNets to build deeper neural networks and avoid vanishing/exploding gradient problem

(G) Residual blocks are used in ResNets to build deeper neural networks and avoid vanishing/exploding gradient problem

Soln: (B) (1) and (D) (1) [0.5 mark will be deducted for additional choices on top of (B) & (D)]

(B) Batch normalization does not normalize the weights, it normalizes the activations of the hidden layer $h = \sigma(Wg)$ — this is not normalized; this is normalized

See lecture notes

↳ Another trick: **Batch normalization**

- It centers each hidden activation and can speed up training by 1.5-2x
- It normalizes the activations of each layer to unit-variance Gaussians
- Is applied immediately *after* fully connected/conv layers and *before* non-linear activations

(D) CNN has less parameters → less capacity → lower chances of overfitting

QUESTION 10 [3pts]: Consider a hidden layer with ReLU activation functions:

$$z_i = \sum_j w_{ij} x_j + b_i$$
$$h_i = \text{ReLU}(z_i)$$

where,

$$\text{ReLU}(a) = \begin{cases} a & a > 0 \\ 0 & a \leq 0 \end{cases}$$

(a) Write out the backpropagation rules for computing the gradients of $\bar{z}_i$, $\bar{x}_j$, and $\bar{w}_{ij}$, in terms of $\bar{h}_i$ Note, the notation $\bar{v} = \frac{\partial \mathcal{L}}{\partial v}$, where $\mathcal{L}$ is some loss function

(b) Based on your answer to part (a), for what values of x_j and z_i are we guaranteed that $\bar{w}_{ij} = 0$?

Soln: (a) $\bar{h}_i = \frac{\partial \mathcal{L}}{\partial h_i}$

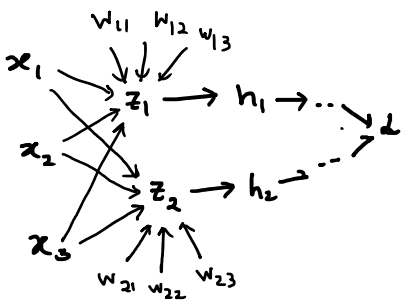
$$\bar{z}_i = \frac{\partial \mathcal{L}}{\partial z_i} = \frac{\partial \mathcal{L}}{\partial h_i} \times \frac{\partial h_i}{\partial z_i} = \bar{h}_i \frac{\partial h_i}{\partial z_i} = \bar{h}_i \frac{\partial \text{ReLU}(z_i)}{\partial z_i}$$

$$\Rightarrow \bar{z}_i = \begin{cases} \bar{h}_i & z_i > 0 \\ 0 & z_i \leq 0 \end{cases} \quad (1/2)$$

$$\bar{x}_j = \frac{\partial \mathcal{L}}{\partial x_j} = \sum_i \frac{\partial \mathcal{L}}{\partial z_i} \times \frac{\partial z_i}{\partial x_j} = \sum_i \bar{z}_i w_{ij} \quad (1/2 \text{ for } \bar{z}_i, \ 1/2 \text{ for } w_{ij})$$

$$\bar{w}_{ij} = \frac{\partial \mathcal{L}}{\partial w_{ij}} = \frac{\partial \mathcal{L}}{\partial z_i} \frac{\partial z_i}{\partial w_{ij}} = \bar{z}_i x_j \quad (1/2)$$

Example



(b) (1/2) If $x_j = 0 \rightarrow \bar{w}_{ij} = 0$

(1/2) If $z_i \leq 0 \rightarrow \bar{z}_i = 0$

$\Big\} \quad \bar{w}_{ij} = 0$