Yifei Yuan

Question SET 1

My answer is: [C]

Analysis:

A: Even if $G_t = 0$, the Kalman filter can still estimate the state using the prediction step. It just won't update the estimate based on measurements, since the kalman gain becomes 0. So it's not true that estimate is impossible, it just lacks correction from observations.

B: The kalman gain K_t is calculated from parameters like $\hat{\Sigma}_t$, C_t, and Q_t, z_t doesn't depend on the actual measurements $z_{1:t}$. So, it's wrong

Therefore, the correct answer is C.

Question Set 2

2A. There are two goal states marked H, one gives +100, the other gives -100, and each probability is 0.5.

∴ the expected reward is :

$$E[R] = 0.5 \times 100 + 0.5 \times (-100) = 0$$

The expected reward is 0.

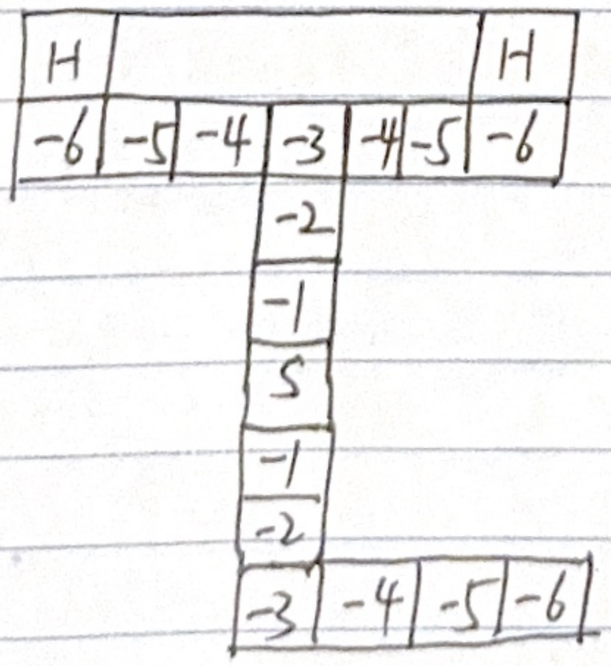
2B. Since the expected reward from moving to either H state is 0, and the reward of moving is -1 per step, so the best strategy is at the starting point.

∴ The optimal policy is to stay at state S.

2C. Now, if the agent must move, and must choose a path to one of the Hs. There should have two symmetric paths to either H, one is choosing to go left path to the ~~left~~ left H, another way is going to right path to the right H.

For both paths, the expected reward is $0.5 \times 100 + 0.5 \times (-100) - 7 = -7$, so, both are equally optimal.

Therefore, the optimal policy is, the agent move upward, then choose either left or right, then keep moving until reaching H.



Draw the white colored cell in the image and write the state values.

2D. The number of all possible deterministic policies for this grid-world is : $4^{16} = 4,294,967,296$

2 cells marked H: Terminal states where the game ends, no actions
1 cell marked S : Starting state, require actions
15 empty cells ; all require actions.

For each state, can choose from 4 actions : North, South, East, West.

So the total deterministic policies $= 4^{16} = 4,294,967,296$

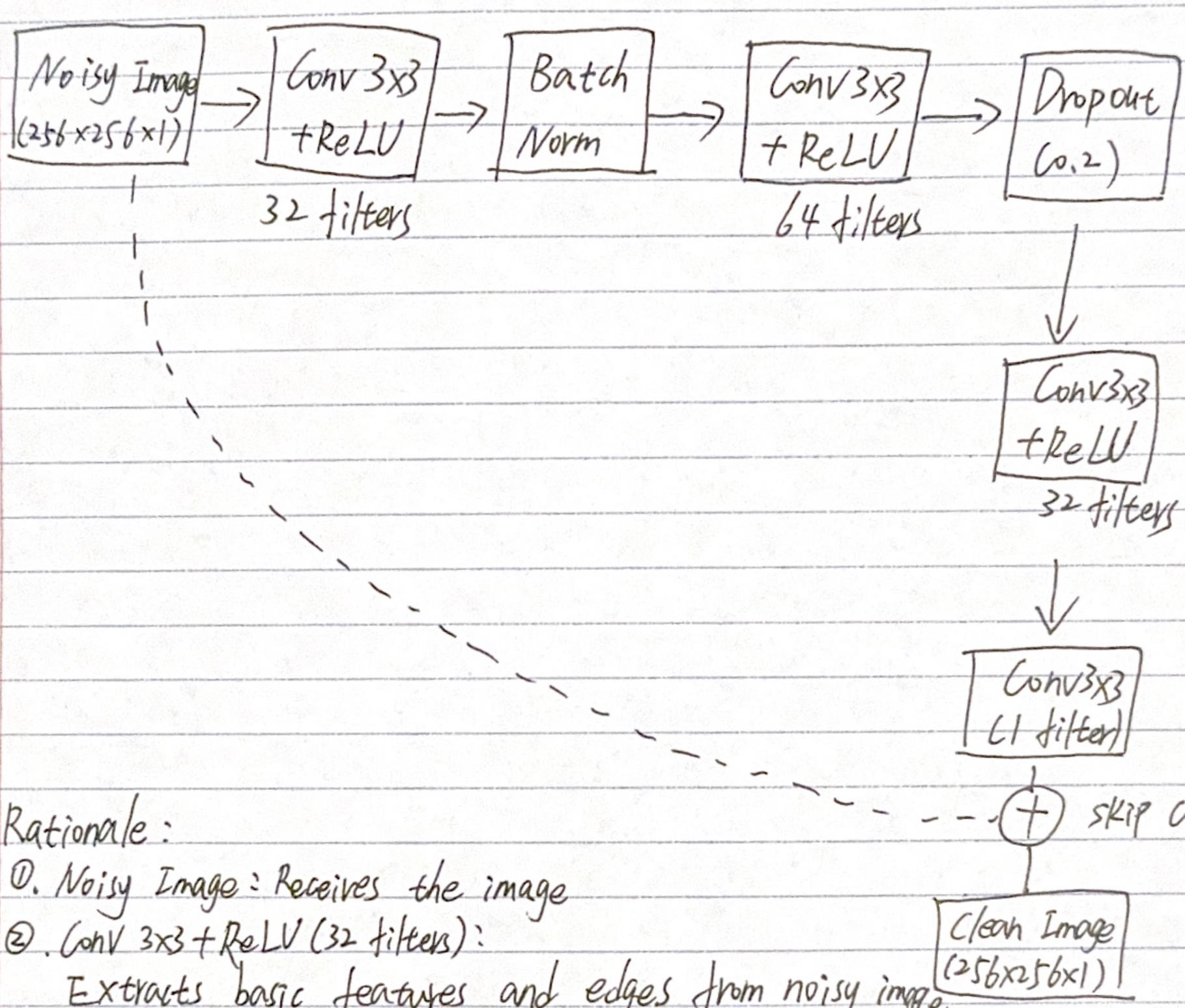
Question SET 3

Skip connections in V-Net are important because they help the model keep useful details that might get lost during the encoding process. When the image goes through the encoder, it gets downsampled to capture high level features, but in doing so, a lot of fine details, like edges or the exact shape of objects are lost. So, if the decoder had to recover all ~~ete~~ of that from the compressed features, it would be really hard and probably not very accurate.

And after adding skip connections, we let the decoder access earlier feature maps from the encoder, which still contain a lot of spatial information. This can make it much easier to rebuild a detailed segmentation map, especially for things like object boundaries. It also helps the model mix low level ~~ff~~ visual features with high level ones, this can improve both the accuracy and the sharpness of the output.

I think these are what skip connections help the network achieve in the semantic segmentation task in my view.

Question SET 4



Rationale:

①. Noisy Image: Receives the image

②. Conv 3×3 +ReLU (32 filters):
Extracts basic features and edges from noisy image

③. Batch Normalization:
Stabilizes training by normalizing layer inputs

④. Conv 3×3 +ReLU (64 filters):
Learns more complex noise patterns

⑤. Dropout layer:
Prevents overfitting during training

⑥. Conv 3×3 +ReLU (32 filters)
Refines noise detection features

⑦. Conv 3×3 (1 filter)
Produce the noise estimate

⑧. Skip connection:
Adds original image to noise estimate, Output = Input - Learned Noise

⑨. Clean Image Output

The loss function I used is MSE:

$$L = \frac{1}{N}\sum (I_{clean} - I_{pred})^2$$

L: loss value

N: the total number of pixels

I_{clean}: Pixel value from the clean image

I_{pred}: pixel value from the predicted image.