Image & Vision Computing
Comprehensive Exam

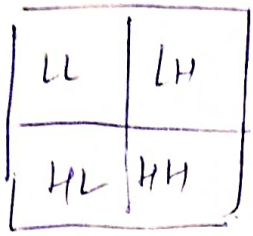
Anubhooti Jain
D20CS002.

Ans:

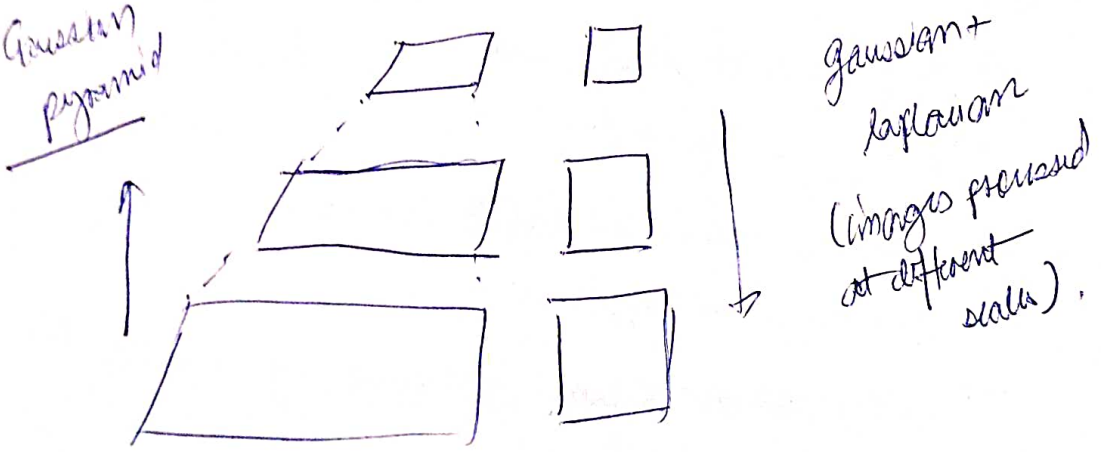
(a) Gaussian Pyramid, DCT, and DWT

Diff (1) Gaussian pyramid is used for processing images at different scales while DCT and DWT are transformation in frequency-band. DCT is discrete cosine transformation and DWT is discrete wavelet transformation. So, the transformation procedure is completely different.

for DWT



L → low band
H → High band



difference (2) The information processed

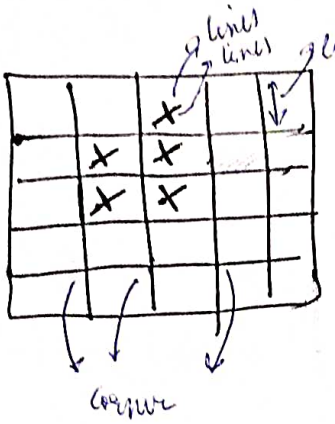
Gaussian → spatial information is more visible and effected as images are upsampled & downsampled.

DCT → Information is more visible in frequency domain but changes are visible in spatial domain as well.

DWT → Frequency-based information. We can manipulate low and

high frequency kernels to manipulate the spatial domain.

Ques 1. (b) Corners, lines, edges



In a simple checkboard-type image corners, edges and lines can be described as shown.

for edge detection

- we have filters available for edge detection. It is known that first order derivatives can help detect an edge
- an edge is noted by immediate change

detecting : line b/w background and circle.

boundary of circle is an edge

→ there will be an immediate change in the gradient information

→ we can detect it using first order differentiation

* known filters & Sobel filters
  - they are used for edge detection
  - a better known version is the 3×3 sobel filter.

For all 3 - corner, edge, line → we have filters available. Sobel filters are well known for edge detection. There is also Hough transform, though, they are better at detecting shapes.

Line detection

As said, Hough transform can be used for line detection. This transforms the parameter space [x,y → θ,r] and the entire image is changed. Intersection of several points indicate existence of lines. We can easily detect edge based on that.

Corner detection

Harris corner detection is one well-known algorithm for corner detection. It usually has a reference image (a typical chessboard) with respect to which we can extract corners. It also works on the differential gradients with direction to detect corners.

Ques 1 (a) $f = \begin{bmatrix} 130 & 11 & 67 \\ 10 & 10 & 50 \\ 80 & 90 & 100 \end{bmatrix}$ ← Mean & Median filter

the centre value changes with mean and median filter.

Mean filter

$$= \frac{1}{9} \times (130 + 11 + 67 + 10 + 10 + 50 + 80 + 90 + 100)$$

$$= \frac{1}{9}(548) = 60.8 \simeq 61.$$

Median filter

- ordered
- 10, 10, 11, 50, (67), 80, 90, 100, 130

↓

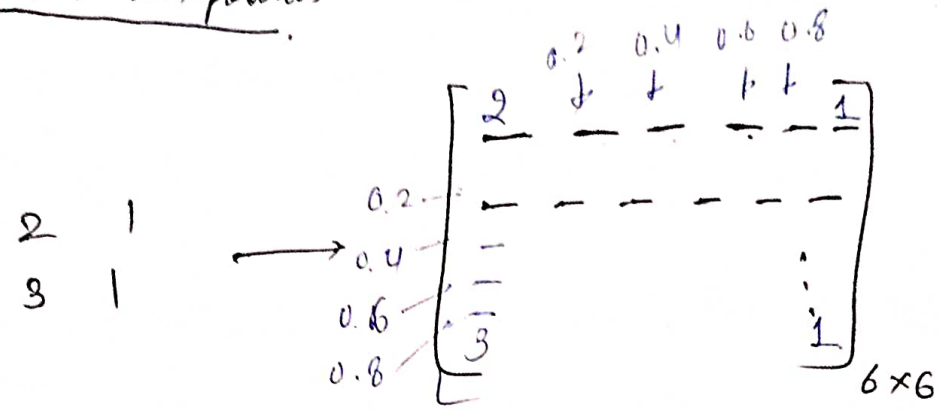
∴ 67

Mean → 61
Median → 67

Ques 1 (a) $I = \begin{bmatrix} 2 & 1 \\ 3 & 1 \end{bmatrix}_{2\times 2}$ (4)

resize by 3 times →

(i) bilinear interpolation



$2\times0.8+1\times0.2$

$$= \begin{bmatrix} 2 & 1.8 & 1.6 & 1.4 & 1.2 & 1 \\ 2.2 & 1.96 & 1.72 & 1.48 & 1.24 & 1 \\ 2.4 & 2.12 & 1.84 & 1.56 & 1.28 & 1 \\ 2.6 & 2.28 & 1.96 & 1.64 & 1.32 & 1 \\ 2.8 & 2.44 & 2.08 & 1.72 & 1.36 & 1 \\ 3 & 2.6 & 2.2 & 1.8 & 1.4 & 1 \end{bmatrix}$$

(ii) bicubic Interpolation

Ques 2(e). $f(x) = [60\ 60\ 60\ 100\ 100\ 100]$

First and second order derivative.

1st order

$$\frac{\partial f}{\partial x} = f(x+1) - f(x).$$

$$\frac{\partial f(x)}{x} = [0, 0, 40, 0, 0]$$

2nd order

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

[60, 60, 60, 100, 100, 100]

$= 60 + 60 - 2\times60 = 0$

$= 60 + 100 - 2\times60 = 160 - 120 = 40$

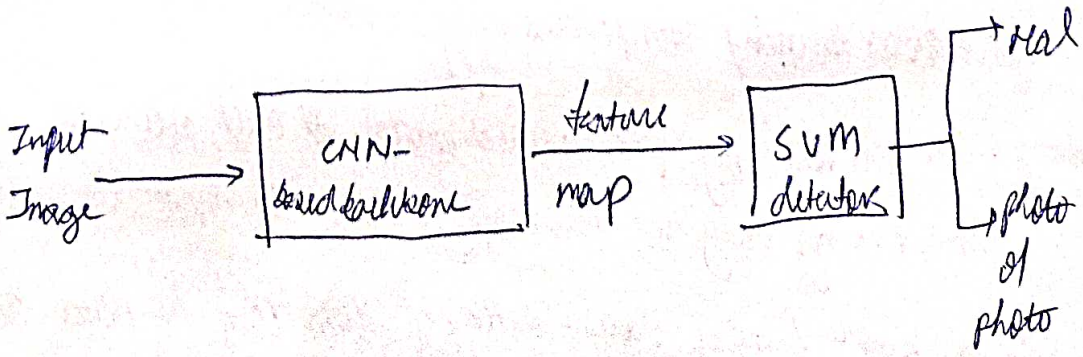
$= 60 + 100 - 2\times100 = 160 - 200 = -40$

$= 100 + 100 - 2\times100 = 0$

$$\therefore \frac{\partial^2 f(x)}{\partial x^2} = [0, 40, -40, 0]$$

Ques 2(f) Detecting photo of photo

Deep learning pipeline.



(1) The deep learning architecture has 2 components

↳ (a) CNN-based backbone — this is used for feature extraction.

(b) SVM detector — this is used to classify between whether the image is real or a photo of a photo.

↳ I/P → features extracted using CNN

O/P → 0 for fake/photo of photo
1 for real

* Reason :— SVM detectors can learn the boundary between this binary classification task with good accuracy, limited examples, and diverse kernel choice — linear, poly or so

for CNN based architecture → we can use a well-known model like VGGNet or ResNet18 or ResNet50.

Reason: — breaking the image into feature maps allows to extract important information out of a sparse input like an image. It reduces dimensions of the input without any loss in the of crucial information hence making SVM training computationally easier

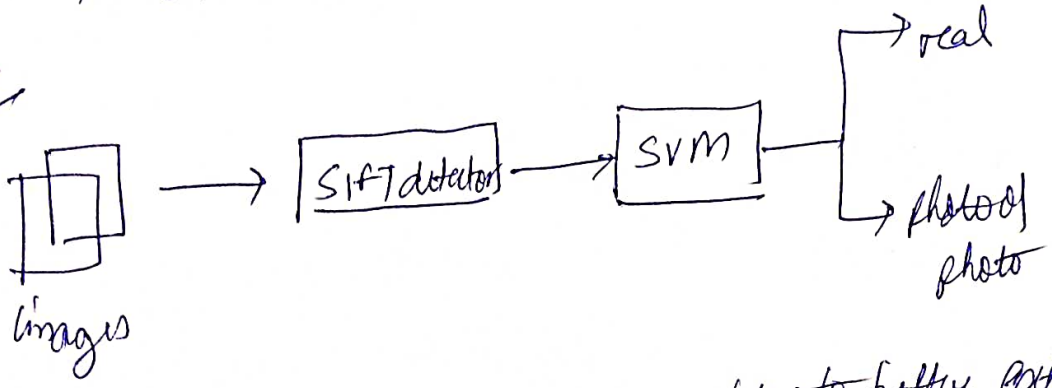
— we can use any pre-trained model. It will allow to reduce training time having only to finetune on our dataset.

Loss : a simple binary cross-entropy loss can be used or better one pre-trained backbone with SVM and its own loss.

(5) The hand-crafted algorithms

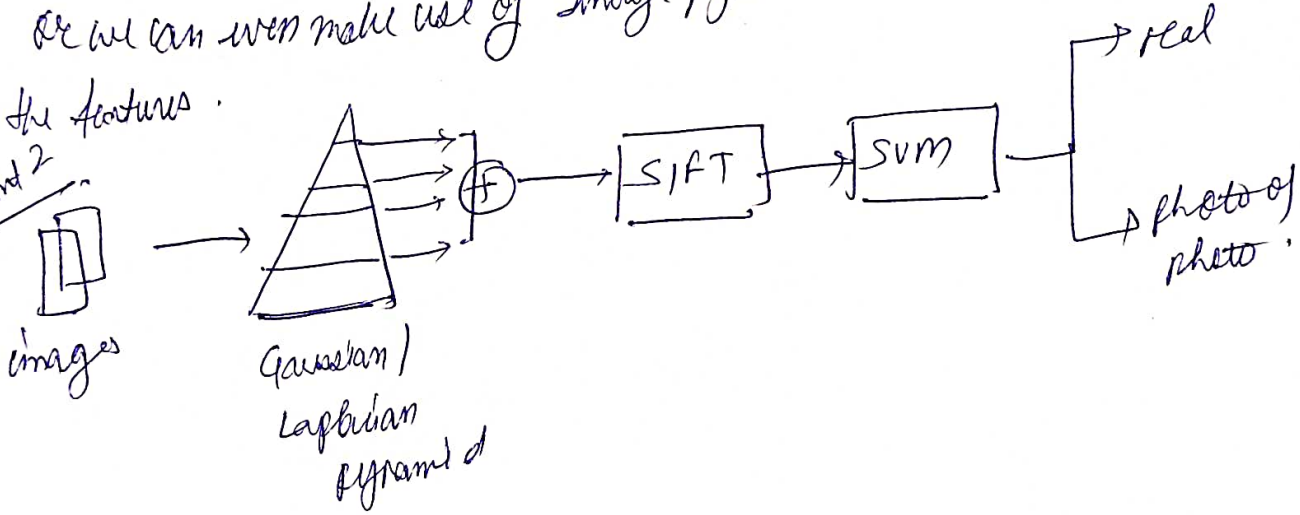
↳ we can make use of SIFT features before using SVM detector for final detection.

Variant 1



or we can even make use of Image pyramids to better extract the features.

Variant 2



Continuing with variant (1), we can extract features just like we were doing in deep learning architecture by replacing CNN-based model with SIFT features. SIFT are hand-crafted and uses keypoint detection to extract features. HoG is also a possible option with gradient & direction and considered in its calculation. The goal is to extract image feature before sending these features to an SVM model for detection training.

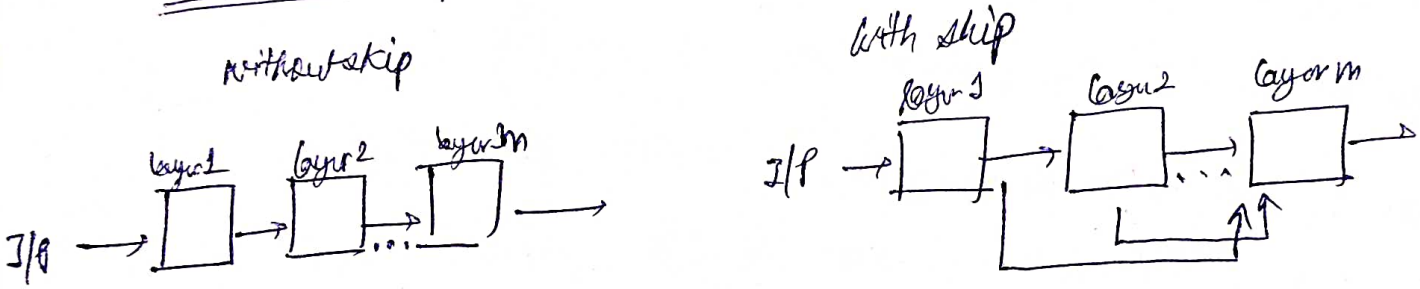
Ques 2 (a) Vanishing Gradient problem

Vanishing Gradient problem arises as depth of deep neural networks increases. The gradients (smaller values especially) cease to exist before they reach last layers thus losing information and reducing the performance of models.

In context of object detection, we have object detectors which are deep to be able to get image features, these deep models can easily face the said issue of vanishing gradient.

Solutions to vanishing gradient problem →

↳ Skip connections / Residual Connections

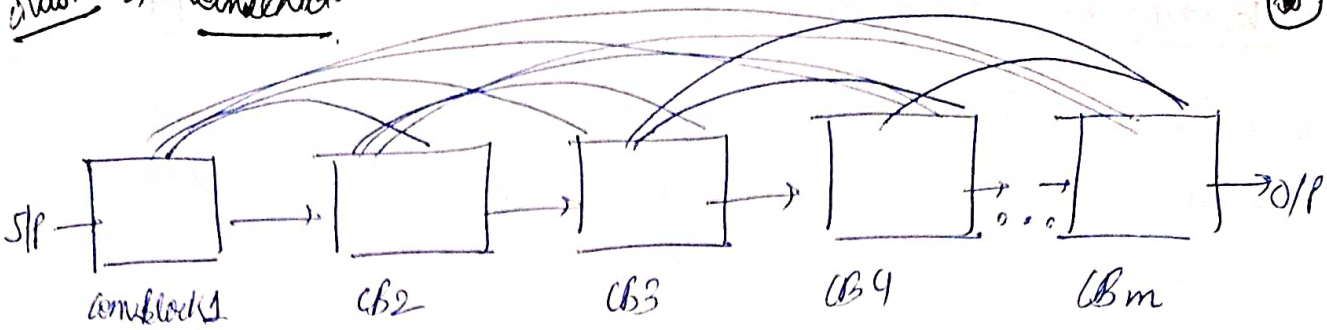


* Skip connections allow features from previous layers to reach as it is, so as to avoid vanishing gradient problem.
* We can have random layers provide their output to further layers skipping one or several layers in between.

Models like Resnet and Densenet uses skip connections. Infact, skip connections were introduced with ResNet model when performance on deep models with high depth was reducing. With ResNet, authors were able to mitigate this issue.

Further, Densenet went ahead to use all layers being connected to all the other layers. Fully connected model is also, a good example of skip connection.

Ques 2: (b) DenseNet



CB → Convolution Blocks.

With DenseNet, the skip connections were found to all the blocks in the model architecture. Each block is connected to every other block (as in the figure above). These blocks are also known as dense blocks and is used for Image classification. It can also be utilized to get feature maps by removing the classification layer at the end or pulling maps from later layers.

* There are also Bottleneck layers in the model. Bottle Neck layers have 1×1 convolutional layers (that help with dimension reduction) and also uses ReLu activation function with average pooling and Batch Normalization.

Advantages of DenseNet

(1) There is better gradient flow than the models proposed before DenseNet. With skip connections, vanishing gradient issue is mitigated, moreover, by storing it between all the dense blocks (possible with same size feature maps) it allow for better flow.

(2) With 1×1 convolutional layers in the model, the number of parameters are reduced as opposed to other equivalent deep models with similar depth.

(3) Better feature maps → as all the layers are inter-connected, the output from initial layers travel quite far, making the resultant feature maps better than those generated by previous models.

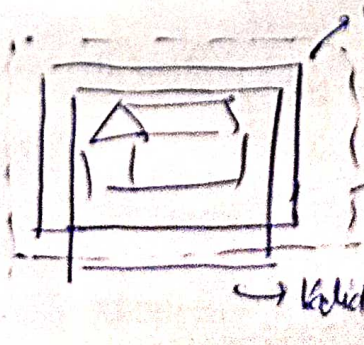
Transition layer : Transition layer has 1×1 convolutional layers followed by an average pooling layer. It is used between dense blocks so that feature maps can flow easily from one block to another.

Ques2 (c) Mean Average Precision (mAP)

mAP is used as an evaluation metric in an object detection algorithm. It allows to compute how accurately a bounding box is predicted by an object detection algorithm. Higher the mAP, better the performance of algorithm.

mAP is dependent of the IoU threshold. IoU → Intersection over union is a way to compute the overlapping of 2 boxes. How much should the two boxes overlap in order to counted as valid prediction is dependent on the IoU threshold.

For example.

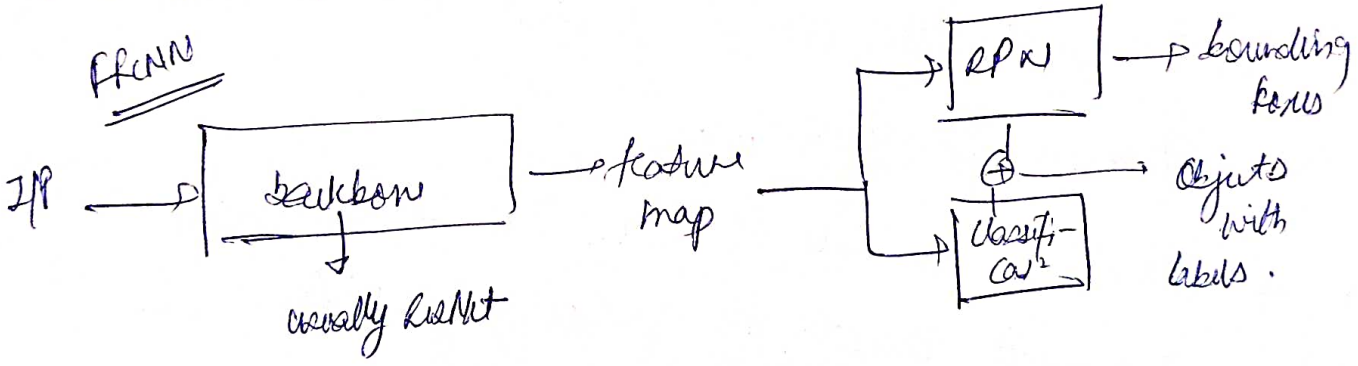


→ All 3 boxes are valid, however they do not completely overlap, they intersect at different ratios which is computed using IoU

The threshold can be set to as high as 0.95 or at an average of 0.5. mAP changes based on this threshold and is therefore usually used as an average over several threshold values when reported in an research.

Ans 2 (d). RCNN Vs faster RCNN

Both RCNN and faster RCNN are object detectors. They are used for object detection algorithm with both considering object detection as a regression problem. They are classified as 2-stage detectors. The difference between the two is their architecture. FasterRCNN proposed a region proposal Networks (RPN) to predict the bounding boxes. It used deep Neural Networks based RPN to predict the boxes which makes it faster and more accurate than the RCNN.



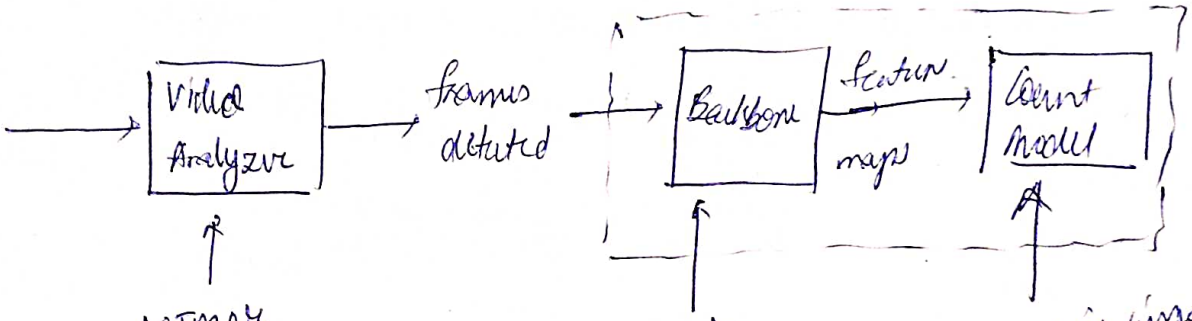
RPN is the novel module in fasterRCNN. YOLO on the other hand visualizes the object detection problem completely differently. It divides the entire image into a grid and using grid as anchors, it predicts the bounding boxes for objects. It is a 1 stage detector. That is, it is better in terms that their is end-to-end training making the model faster & robust.

Qno2 (c) Automated Video Surveillance

↳ Counting cars at toll plaza
↳ Motion detection.

* We need tracking algorithm → that is detection + prediction. That is, frame is predicted and corrected with respect to previous frames.

* A system.



~~DETR~~

Proposed algorithm

↳ Video analyzer with DETR
↳ Every frame is given as an input to DETR (DETR is a transformer-based object detector that breaks images into patches before adding positional embeddings and feeding it to transformer)

A potential challenge is a lot of frames will have to be processed, with different traffic density at different times of the day. We will need a time analyzer to detect this change / predict this change so as to change the model based on the time of the day.

Qun 3. (a) PCNN - summary

PCNN is Part-based CNN with 3 modules proposed for fine-grained visual categorization (FGVC) task. The goal is to detect discriminative object parts and learn discriminative features from global images as well as from discriminative object parts.

(1) Squeeze and Excitation block (SE) block

- uses global information
- adaptively recalibrates channel-wise feature responses
- goal is to selectively emphasize informative channels & suppress less useful ones.

(2) Part Localization Network (PLN)

- it is a deep fully convolutional Network to locate distinctive object parts.
- ~~It~~ trained under unsupervised learning by analyzing the excited feature maps due to SE blocks, with each learned part detector.
- Number of part ~~det~~ detector is upto us. In paper = 4.

(3) Part Classification Network (PCN)

- has 2 streams
  - ↳ stream 1 : classifying each individual object part into image-level categories.
  - ↳ stream 2 : concatenate part-level local features and global image feature from an image together into a joint feature with softmax function for final classification.

Loss: A new Duplex Pooled Loss (DPL) is proposed

↳ DPL: loss function is reshaped to focus on training on hard examples and thus down-weighting easy examples.

Things to note:-

(1) a set of C x 1 x 1 part detectors are learned.

(2) both PCN and PIN are trained jointly in an end-to-end fashion.

(3) Metric based learning motivates the loss but with having focused on harder examples from the easy examples.

∴ Loss = Metric learning Loss + Post classification loss + Image classification loss.

reshaped

Limitations

→ the part bounding boxes in the paper is cut at 96x96 pixels with original input as 448x448 pixels. parts which are smaller in size might not be detected in this case. we need flexibility with the choice of part bounding boxes.

→ Another limitation is the use of VGG Net as backbone. It might reduce the performance of the proposed PCNN in different domain. And with different backbone the entire setting (initialization) will have to be changed.

Ques 3 (b) Solution to PCNN limitation

Limitation → (1) Rigid size choice for part bounding boxes.

Solution → use a different architecture for PCNN on the initial PCNN. Architectures like Transformer-based detectors can work well with variable length boxes. Or for localization networks, as no bounding boxes are available, we can add a regularization term to the loss function penalizing for larger part boxes so as to force the architecture model to look at smaller part boxes.

Sol-(1) $L_{PLN} = \min \sum_{n=1}^{N} \sum_{i=1}^{|X|} (dis(m_n(X_i)) + \lambda_1 div(m_n(X_i)))$

$dis(m_n(X_i))$ ↓ distance function

$div(m_n(X_i))$ ↓ diversity function

where,

$m_n(X) = sigmoid(O_n^T U)$

For each cluster we can add a regularized term α.

$L'_{PLN} = L_{PLN} + \alpha \, \| \text{distribution variance} \|$

→ Regularizing the variance should help keep the part detector within the size limits but also intend. That is, without explicitly putting at 96×96 pixels we can use variance to limit the size.

Ans(2) Another sol^{n} could be to utilize the Image pyramids to break the post detections into smaller parts without having to explicitly explicitly deciding on 1 constant size.

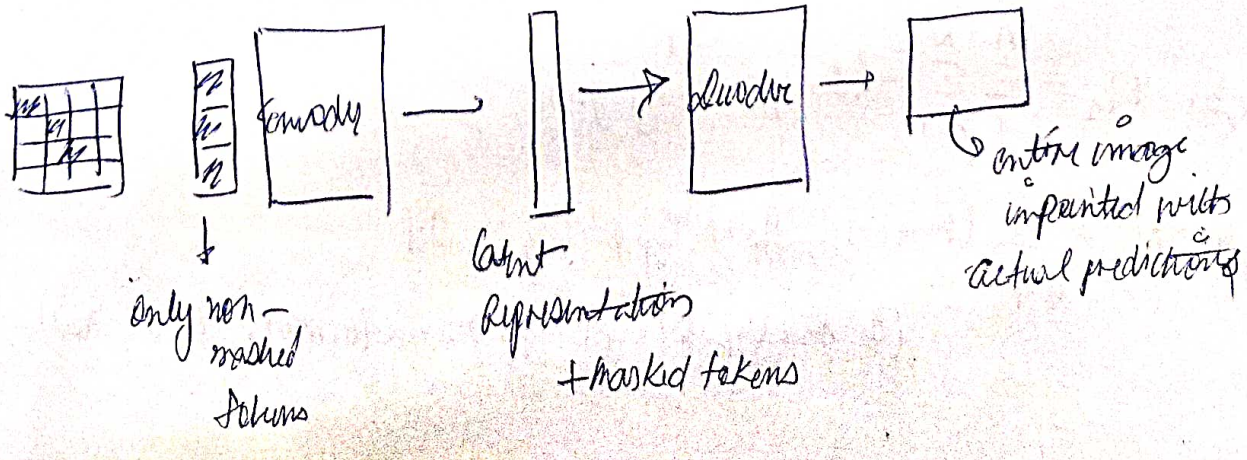
Ques 3. (a) Input image is 50% masked

↳ in patches using a 3×3 mask with all values 0

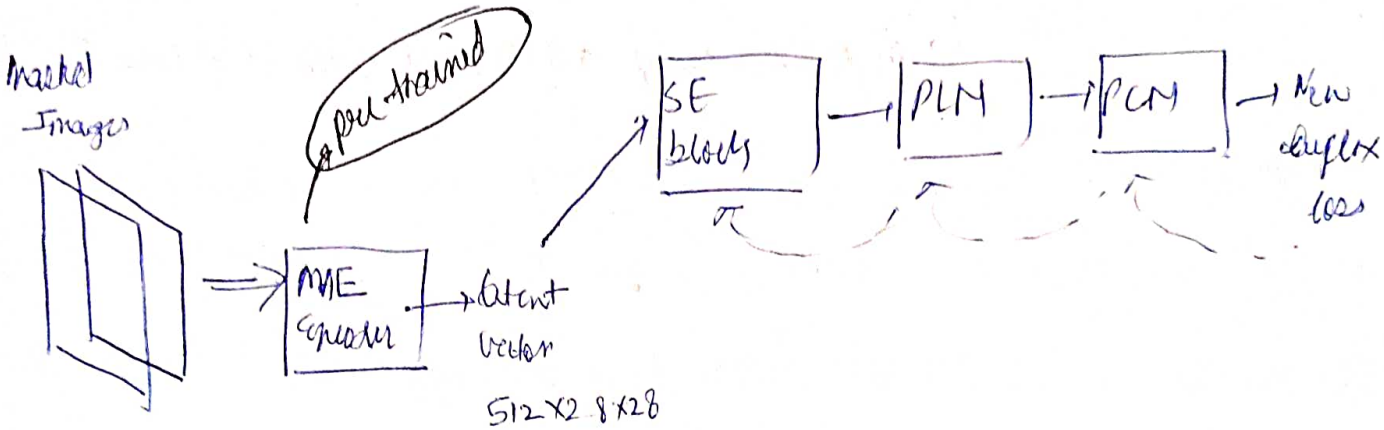
We can use Autoencoders before the PCNN networks to handle the Input image that is masked. There exists masked Autoencoders, that are able to reconstruct an image when it is upto 75% masked. These masked Autoencoders are pre-trained on heavy datasets in self-supervised fashion so we can either the reconstruct the masked images or we can use the Autoencoders encoder instead of pre-trained VGG Net to get the features. due to pre-training, the encoder learns specifically to get the important/essential features of a masked image.

Modified steps.

Masked Auto Encoders (MAEs)



MAE :- self-supervised pre-trained networks
- trained as vision transformers at backend.
- can be used instead of VGG



Mathematically, everything remains same as the only change is in the backbone choice to get the feature maps of the input for SE blocks.

Sec 3. (d) Duplex Focal Loss.

$$\text{Eq. 7}: L_{PCN} = \min \left[\underbrace{FL_1 + \lambda_2 FL_2}_{\text{Focal descriptor}^2 \text{ \& metric learning}} + L_{cls} \right]$$

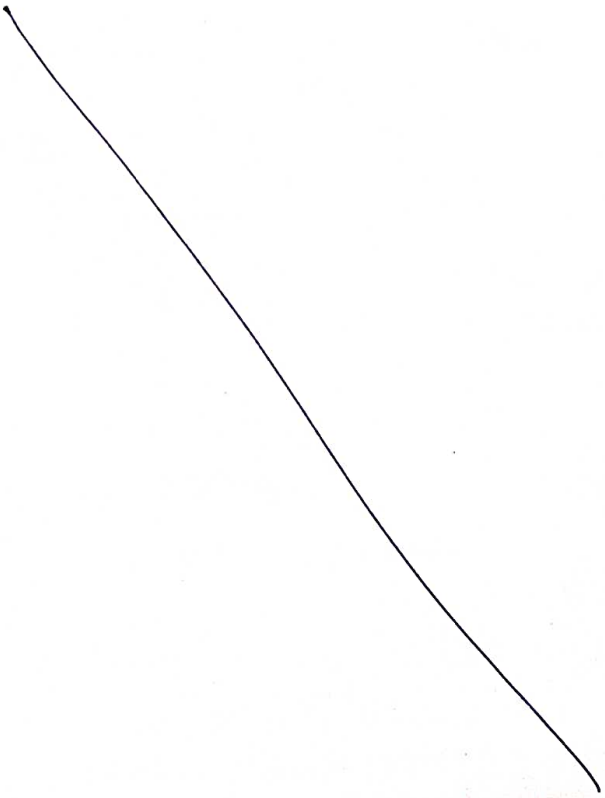
$$\text{Eq 8}: FL_1 = \sum_{i=1}^{1\times1} \sum_{n=1}^{N} -(1-y_{i,n})^{\gamma} \log y_{i,n}$$

$$\text{Eq 9}: FL_2 = \sum_{q,p} (1-\max(y_q, y_p))^{\gamma} l_{qp} D^2(x_q, x_p) + (1-\max(y_q, y_p))^{\gamma} (1-l_{qp}) \max(0, m_{rg_2} - D^2(x_q, x_p))$$

Replace Eq 8. with SVM style margin based loss.

$$FL_1 = \sum_{i=1}^{|X|} \sum_{n=1}^{N} -(1-y_{i,n})^{\delta} \log y_{i,n}$$

Ques 3 (e) a handcraft approach will be ~~to~~ get features from handcraft method like SIFT or HoG instead of VGGNet and then pass it through convolutions / filters through the 3 modules.



Ques4. (a) Gray-world assumption

Gray world algorithm assumes that the average reflectance in the scene is achromatic. Hence, the illuminant color distribution is simply estimated by averaging each channel independently. It achieves good visual performance. but performs poorly for extremely deteriorated underwater scenes. It fails to remove the color cast and looks bluish or suffers from severe artifacts.

In the paper, author uses Gray-world algorithm to compute the white-balance image but after compensating for the loss of the red channel.

That is,

[Red loss compensation] + [Gray world algorithm] → white balanced image.

↳ observations are built based on this compensation

↳ the original algorithm (with gray world assumptions) is used on the compensated image.

Ques4 (b) problem of saturation of color channel

Saturation of color channel (mostly red channel) can arise due to the Gray world step. The enhancement of red should primarily affect the pixels with small red channel values and should not change pixels that already include a significant

red component. Basically, If compensation of red or any other channel is performed in those regions that are not highly attenuated it may lead to saturation of that particular channel.

Ques 4(c) In proposed method, the red channel compensation depends on global characteristics of the image.

eq 4. $I_{rc} = I_r(n) + \alpha(\bar{I}_g - \bar{I}_r) \cdot (1 - I_r(n)) \cdot I_g(n)$

+ $\bar{I}_r$ and $\bar{I}_g$ in the equation are mean value of I_r and I_g, that is, the red and green channels of the image. Entire image is averaged in order to compensate for the red channel.

+ If mean value changes in a local neighborhood, we may be able to better compensate the attenuation.

There is one issue with making red channel compensation globally dependent, the entire image might not be attenuated consistently. Lighting and depth changes in the image, hence, some areas might get over or under compensated. It is thus better to make this compensation locally-dependent rather than globally-dependent. The neighborhood can be defined as a simple order statistic of 3×3 or 5×5 patch or we can use segmentation mask, though it will be add to the computation overload for the image and entire algorithm

---

Ques-1(d) Red channel compensation

Locally-dependent.

$$I_{rc}(x) = I_r(x) + \alpha(\bar{I}_g - \bar{I}_r)(1 - I_r(x))\, I_g(x)$$

↓

Creating a neighborhood of $K \times K$ patches with a total of m patches

$$I_{r,cc}(x) = I_r(x) + \alpha\left[\sum_{i,n=1}^{k,m}(\bar{I}_{gin} - \bar{I}_{rin})\right](1 - I_r(x))\, I_g(x)$$

↓

We take the mean over $K \times K$ neighborhood.

$\bar{I}_{rin}$, $\bar{I}_{gin}$ → "n^{th} patch with i value of neighborhood"

i is k

Pseudo-code

I/P → Image

→ divide the image into m patches of $K \times K$ size

→ compute mean values for red & green channel

$$\bar{I}_{rn}, \bar{I}_{gn} \quad \forall\ n = 1, \ldots m$$

→ compute compensated $I_{rc}(x)$ with this list of means computed on local features of images.

→ Perform Gray world & get white balancing image.

Ques 4 Yes Unequal channel attenuation at different regions of the image. We should apply locally dependent channel compensation strategy. Locally dependent strategy will adjust the local means of channel that are attenuated differently and will therefore compensate differently at the specified region.

Algorithm described in previous answer can be used here or we can take regions as an Input with the underwater image, defining specifically marked regions and using a local mean on these areas.

Ques 4. (f) Gamma Correction

Gamma correction aims at correcting the global contrast. The correction increases the difference between darker and lighter regions at the cost of a loss of details in the under or over exposed regions. The gamma correction trick is helpful when we have too bright or too dull images.

With the proposed method, it helps because in general, underwater images appear to be too bright. Using gamma correction, we can make the image less bright making it more evenly spread.

The issue as stated above with this is the loss of details in under or over-exposed regions. To compensate for this, authors use a sharpened version of the white balanced image. They follow the anchor

masking principle, by blending a blurred version of the image with the image to sharpen using →

$$S = I + \beta(I - G_1 * I)$$

- S ↓ sharpened image.
- I ↓ actual image
- β ↑ constant
- $G_1 * I$ ↳ blurred version

Ques 4. g) Yes, as we ~~to~~ move upward in a Gaussian pyramid, we get more and more high ~~frequency~~ components as we keep downsampling the image and combine it with the sharpened ~~frames~~ of the Laplacian ~~frames~~.

Laplacian ~~pyramid~~ pyramid is a bandpass pyramid as it allows low pass frequency. We can upscale an Image using the pyramid. It is quite opposite →