28/3/22

PAMI Comprehensive

Anubhav Jain
D20CS002

PART-A

Ques 1. $A = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}$ $B = \begin{bmatrix} 1 & -2 \\ 0 & 1 \end{bmatrix}$

Let's say B is the inverse of A matrix, then,

$$B = A^{-1}$$

Also,

$$A A^{-1} = I$$

$$\therefore \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & -2 \\ 0 & 1 \end{bmatrix} = I$$

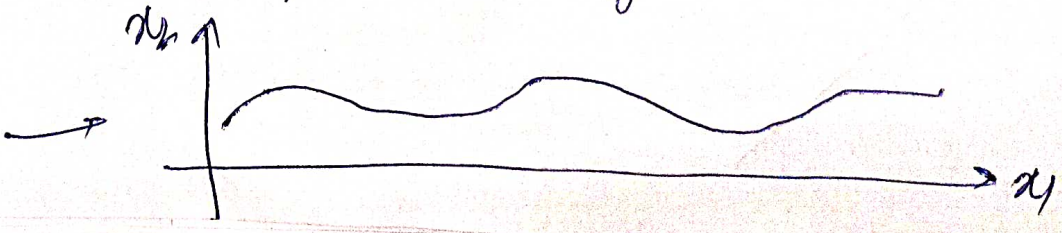
$$= \begin{bmatrix} 1+0 & -2+2 \\ 0+0 & 0+1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \text{Identity Matrix.}$$

∴ True, given Matrices are inverse of each other.

Ques 2. objective function on R^2 where gradient descent will be exceedingly slow.

→ a wavy function with not much of a change could be one function where gradient descent could be slow



gradient step update →

$$\theta := \theta - \alpha \frac{\delta}{\delta\theta} J(\theta)$$

$\alpha \rightarrow$ learning Rate

$\theta \rightarrow$ parameters to be updated

$J(\theta) \rightarrow$ cost function.

* A cost function that doesn't have a big rate of change is one (property of the functions) that will be slow to converge.

* $\alpha \rightarrow$ learning rate can also be reduced.
Cost a function like $\rightarrow x^{1/2} + y^{1/2}$ → can be slow to converge as x & y input features in $\mathbb{R}^2$ domain.

* Any plateau-based function are approaching plateau-like slopes will be exceedingly slow to converge.

Ques3. We can't remove bias parameter from a convolutional layer or a fully-connected layer before applying batch norm is a false statement.

Batchnormalization includes normalizing the data sent in the particular iteration pass. The normalization occurs and allows to reduce the vanishing and exploding gradient problem. This normalization is formulated with a bias term involved; thus, bias before batch normalization layer can be considered quite redundant. There is no need for an extra bias term, and thus we can't only remove the term altogether, having it is quite redundant.

Ques4. Autoencoder with linear activation and single hidden layer is same as PCA. is a true statement. PCA and Autoencoder are differentiated by the non-linearity component possible in the autoencoder. Linear activation and only a single layer will have the same subspace and will reduce dimensionality just as in the PCA. The dimensionality will have to be the same though for this to work out. They will be equivalent in the subspace spanned.

Ques5: In the basic formulation of GAN, both generator and discriminator perform a binary logistic regression with the cross-entropy loss. Is a true statement.

→ $\min_E \max_D G(E,D)$ we have a minmax GAN loss.

↳ G → GAN

E → Generator

D → Discriminator.

Both Generator and discriminator seems to works of binary logistic regression with Generator trying to minimize

$1 - \log E(z)$ $z \rightarrow$ input image

$R \rightarrow$ Representation / Reconstructed Image by E

where,

→ $E(z) = R$

→ $D(R) = x'$

→ $D(E(z)) = z'$

similarly, Discriminator D is trying to maximize. The final formulation looks as follows →

at generator side.

$$G(E,D) = \underset{P_{data}}{E}[\log D(z)] + \underset{P_z}{E}[\log(1 - D(E(z))]$$

for GAN we go in step-by-step training, that is, freeze Generator, update D and freeze discriminator to update Generator. This in its simple formulation can be done using binary logistic regression with cross-entropy loss.

Ques 6. False, Interpretability and Explainability of a neural network are not the same. Interpretability of a model deals with how we interpret the decisions of the model while explainability is more about how we explain a specific decision made by the model.

for example, say we have a simple Neural Network with activation function predicting the rate of houses. Interpretability of the model will include questions like how do we interpret the particular model used. It will also include questions like how do we justify or interpret a choice of activation function used, like using sigmoid squashes the values between 0 to 1 usually leading to normalized values.

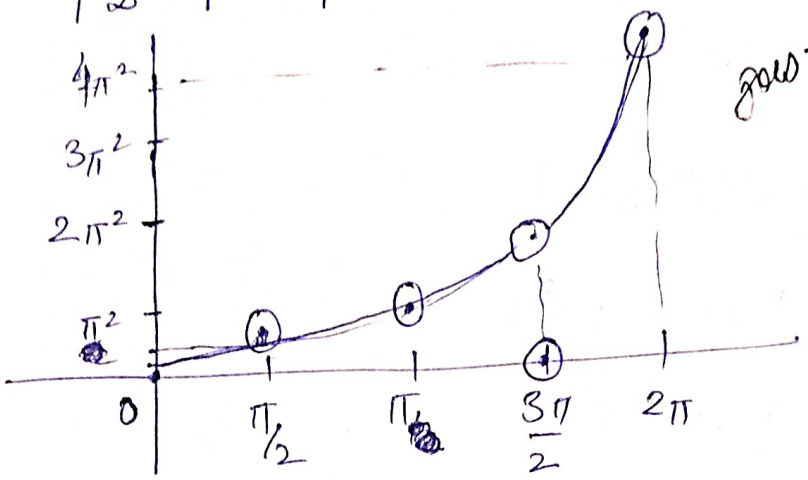
Whereas, explainability will deal with questions like give a particular set of features X, why do the model gives Y as the output. what makes the model take this decision.

Ques 7. AdaGrad

AdaGrad is an adaptive learning rate algorithm which allows to aggressively decay the learning rate as training progresses. It is used to modify the Gradient descent algorithm.

Ques 8. (a) $f(x) = x^2(1+\sin(x))$

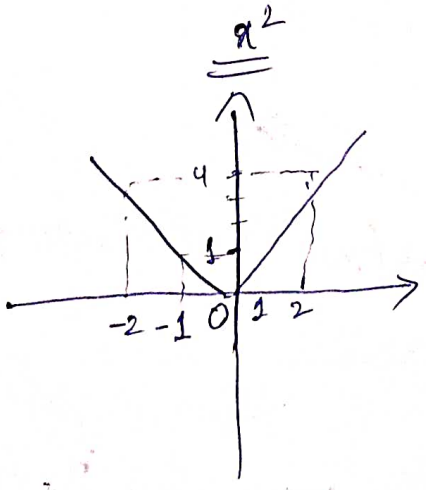
x	0	$\pi/2$	π	$3\pi/2$	2π
$f(x)$	0	$2\times\pi^2/4$ $\frac{\pi^2}{2}$	π^2	$(\frac{3\pi}{2})(1-1)$ 0	$(2\pi)^2$ $4\pi^2$

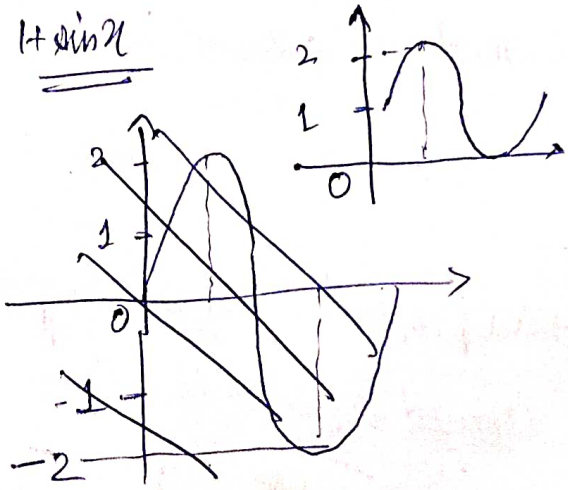


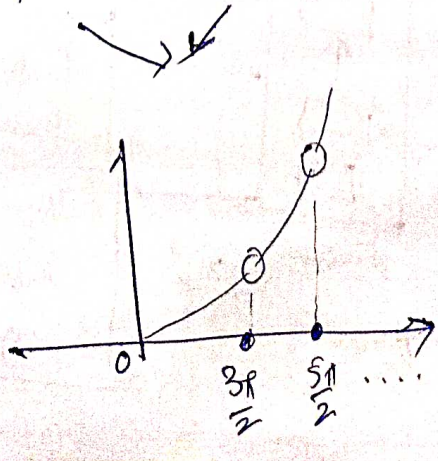
goes to 0 for every $\sin x = -1$.
i.e $\forall x = (2n+1)\frac{\pi}{2}$.

(b) Computation Graph.

$f(x) = x^2(1+\sin x)$







(c) $f'(0)$

$$f(x) = x^2(1+\sin(x))$$

$$= x^2 \frac{d}{dx}[1+\sin(x)] + \frac{d}{dx}x^2 * (1+\sin(x))$$

$$= x^2(0+\cos x) + 2x \cdot (1+\sin x)$$

$$= x^2\cos x + 2x(1+\sin x)$$

$$= x^2\cos x + 2x\sin x + 2x$$

$$= x[x\cos x + 2\sin x + 2]$$

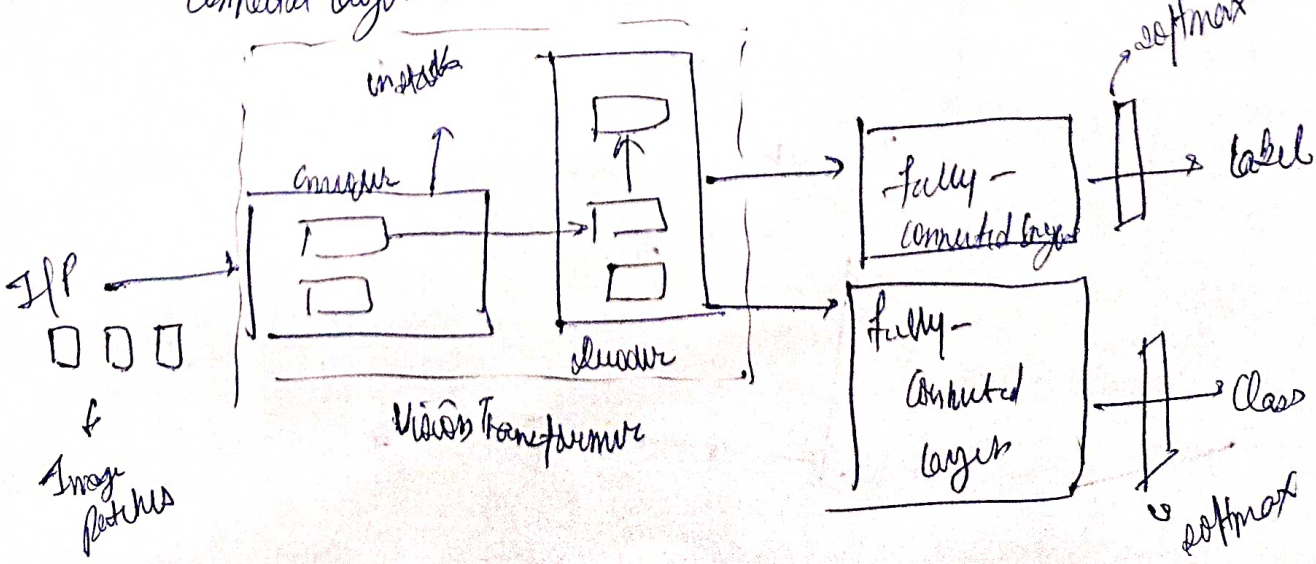
$$f'(0) = 0[0+0+2] = 0.$$

Ques 9. Imbalanced Domain Learning

* "Multilabel - Multiclass problem" (ML-MC)

* n-classes
  m-labels

* Architecture proposed : Vision transformer (using attention) with fully-connected layers at the end.



The input is given in form of patches. Vision transformer can be replaced by simple transformer in case input is textual. Further, attention is used and positional embeddings are required. The processed input is then fed to 2 fully-connected layers, one for n-classes and one for m-labels. Both the layers have softmax layer to choose from the labels and classes based on a probability distribution.

- For imbalanced domain learning we use regularizer in the training process. Here, for training, we give more weights to scarce class or domain not present in abundance. This will allow the model to focus on scarce examples and give them more attention.
- Moreover, using transformers we won't be invariant to textual features which can help with imbalanced domain learning.
- Finally, CNNs can be used but they are not localized, that is, we won't be able to use positional embeddings, neither we will be able to use the attention module.

Ques 10- Distributed DL Learning

1(a) Model & Data parallelism

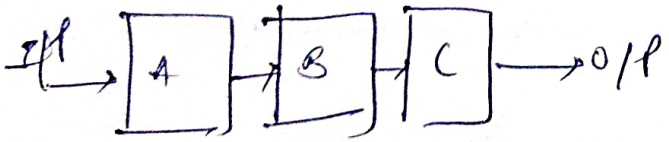
Model Parallelism

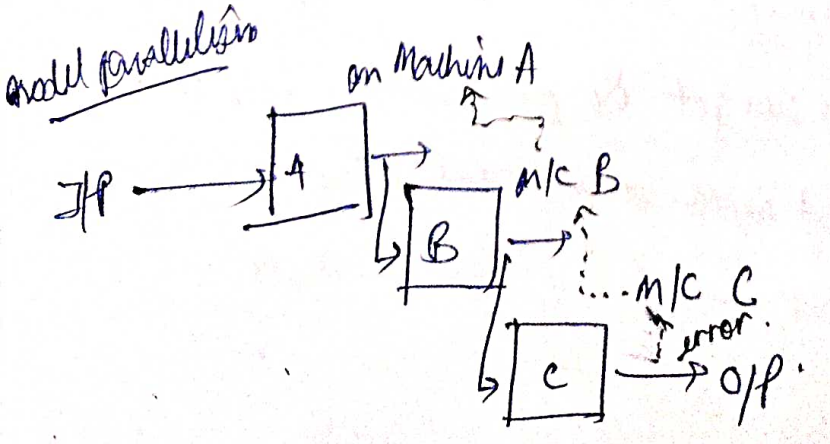
↳ Here data remains the same.

→ Here the model is trained in parallel. One can imagine it training as greedy layer by layer training. However, instead of freezing the layers, we train the layers in parallel over the same dataset.

→ gradients flow back once we reach to the last layer of the model.

Say model has been divided into 3 parts



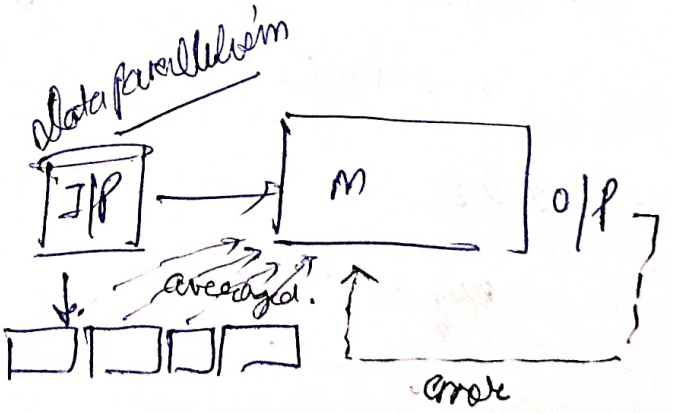


Data parallelism

↳ The same model instance is used. SGD uses certain samples from the entire dataset, Thus we know, the complete data is not required in one go

→ gradients for data parallelism can be backpropagated by averaging them over the entire dataset.

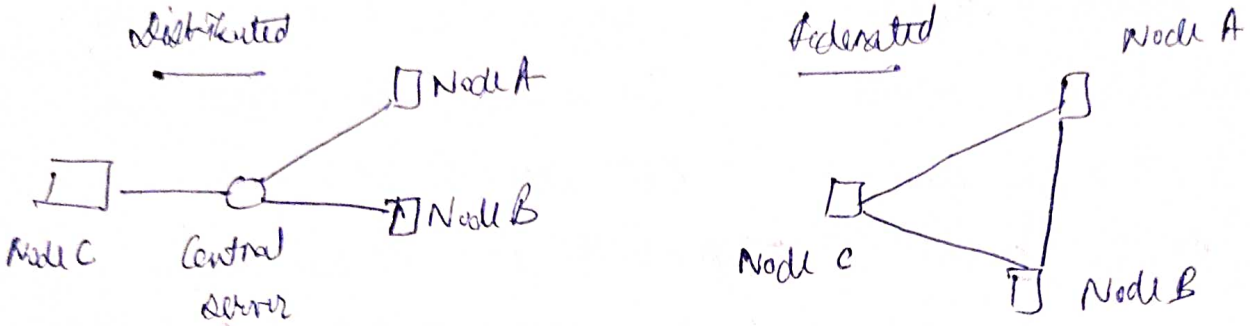
→ Data can be fed in parallel in mini batches, after computing gradients we average & backpropagate them



Weights are updated once the gradients are (avg.) backpropagated

(b) Distributed DL and Federated DL (10)

In Distributed dly learning, we share the model and data among the resources while the updates are made at a central server. On the other hand, federated learning is about having no central server - decentralized learning.



In federated learning, it is upto us as to what to share with the nodes, that is all the nodes only gets the updated weights without having to share their data. They only share the model weights after training the model locally. This way we can ensure data security & privacy.

For Federated DL,

say → m nodes

↳ training happens at m nodes with data at the particular node X_i $\{i \in m\}$

say → after updation we get θ_i parameters $\{i \in m\}$ that parameters are shared with other nodes who combine or update their own model by fusing these updated parameters.

1 Iteration

Parameters $\theta_1, \theta_2 \ldots \theta_m$

data $x_1, x_2 \ldots x_m$

After training ↓ (training at node 1, assume)

$\theta^*_1, \theta_2, \ldots \theta_m$

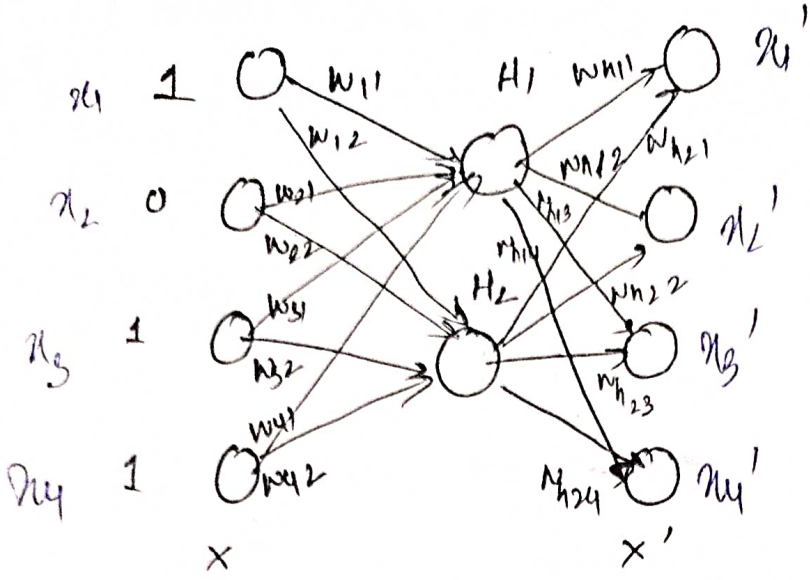
new parameters → θ_1^* $\theta_1^* \theta_2$ $\ldots$ $\theta_1^* \theta_m$

$\theta_1^* \theta_2 \rightarrow \theta_2^*$

$\theta_1^* \theta_m \rightarrow \theta_m^*$

P.T.O

Qussil. $g(x) = x/4$, $\eta = 0.5$



$x = 1\ 0\ 1\ 1$

weights = initialized to 1.

$g(wx)$

$h_1 = w_{11}x_1 + w_{21}x_2 + w_{31}x_3 + w_{41}x_4$

$= 1\times1 + 1\times0 + 1\times1 + 1\times1$

$= 3$

$g(h_1) = \frac{3}{4} = h_1'$

similarly h_2,

$h_2 = 3$

$g(h_2) = 3/4 = h_2'$

$h_1' w_{h11} + h_2' w_{h21} = x_1' = \frac{3}{4} + \frac{3}{4} = \frac{3}{2}$

$h_1' w_{h12} + h_2' w_{h22} = x_2' = 3/2$

$h_1' w_{h13} + h_2' w_{h23} = x_3' = 3/2$

$h_1' w_{h14} + h_2' w_{h24} = x_4' = 3/2$

loss calculation

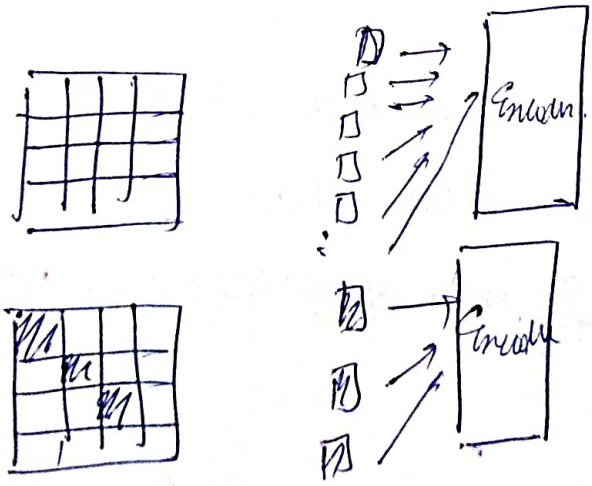
$$\left|\left|\frac{3}{2} - 1\right|\right|^2 + \left|\left|\frac{3}{2} - 0\right|\right|^2 + \left|\left|\frac{3}{2} - 1\right|\right|^2 + \left|\left|\frac{3}{2} - 1\right|\right|^2$$

error $= \frac{3\sqrt{5}}{2} + \frac{3}{2} = \frac{3}{2}(1+\sqrt{5})$

PART-B 1. (Masked Autoencoders)

Ques (a) The specialized sparse operations can include dimensionality reduction and dense feature representation. Images introduces sparsity, which means we can reduce dimensions of the inputs without actually affecting the or losing any important information that can lead to model's final decision. Such operations include using algorithms like PCA or providing entire input to the encoder. Here, in the paper, only non-masked regions were provided to the encoder. Certain models, even have specialized sparse matrix representation to handle such sparse inputs. Dictionary Learning also takes advantage of this particular sparse property.

(b) Why are such operations not required by this model →?



There are no requirements for any specific sparse representations or operations in this paper because upto 75% of the input image is masked leaving only few blocks to be sent as an Input. These unmasked parts acts as dense information, reducing the computation load for encoder drastically.

Ques 2 - Applying to other CNN architectures

* It is particularly mentioned in the paper that convolutions typically operate on regular grids and it is not straightforward to integrate "indicators" such as mask tokens or positional embeddings into convolutional networks.

* That being said, it is not completely impossible. It will however be computationally expensive to integrate the positional embeddings required in the encoder and the mask tokens required in the decoder of the proposed masked autoencoder. We will need extra embedding layers to integrate the information and convolutions will have to be hand-crafted for the information to integrate as required.

* Other models/architectures like → variant of vision transformers like (RVT) or OpenNet (which treats all as vectors) will be easier to work with with the given specifications for the input to the encoder & decoder.

Conclusion → We can apply masking to different architectures even CNN. However, it can be computationally expensive to use the proposed technique in the CNN-based model.

Part 3- Masking Techniques

* In the paper — (1) generate a token for every input patch (non-overlapping patches) using linear projection with an added positional embedding.

(2) Randomly shuffle the list of tokens and remove the last portion of the list based on the masking ratio (high to create complex task)

* Aim of the technique. → The high masking ratio is used here for proper training but it also allows to reduce the computation by decreasing the input size of the encoder.

→ Sampling random patches without replacement eliminates redundancy

→ The uniform distribution prevents a potential center bias.

→ Also, the self-attention has "quadratic" complexity, using high masking ratio allows to avoid ~~to~~ increasing the self-attention's complexity.

* Other strategies used → block-wise & grid-wise sampling. In both the cases, conclusion was that random sampling works better as it allows for a higher-masking ratio, which is good for speedup, computation cost & accuracy.

Other masking techniques

→ pixel-based masking : using pixels to mask rather than tokens

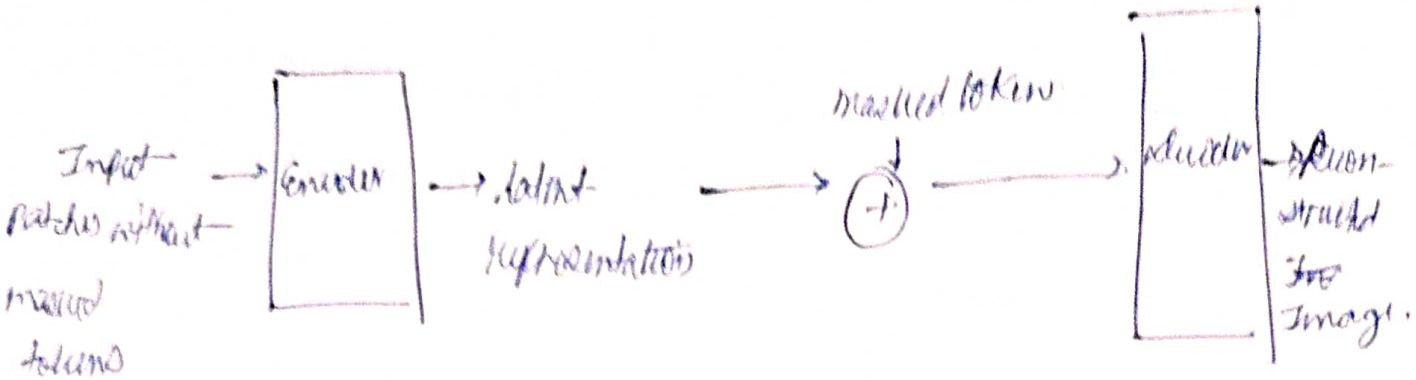
→ grid-based masking : dividing images into grids and using specific grids to be masked rather than tokens.

→ hand-crafted masking : using eigen vectors or SVD decomposition,

These decomposition techniques can be used to mask specific tokens or pixels of the image rather than random sampling techniques.

	Pixel-based	Grid-based	Hand-crafted	Proposed Random sampling
Semantic segmentation & Object Recognition	It can be beneficial as semantic segments are identifying pixels into specified classes	grid-based masking can be fine-tuned but might not capture the feature as pixel-based can	hand-crafted masking can change based on whether reduction is happening on pixels, tokens, or grid	Pixel-based MAE performs better than token-based.
Object Recognition & face Recognition	hand-grid-based masking might work better as pixel based masking can endlessly increase computation. However it might increase performance on smaller objects	Grid-based masking might help capture relevant features as object Recognition looks for bounding box progression	hand-crafted masking might just introduce more computation without much improvement over other techniques	MAE proposed works well on object Recognition, reaching the max metric value.
X-Ray classification	might just increase computation based as classification is on the entire image	might not learn the affected areas in the medical image as good as pixel or tokens	hand-crafted masking can help especially knowing the specific domain	proposed MAE → especially pixel-based could work well

Quiz 4 Self-supervised pre-training



Encoder:

I/P → MAE gets tokens as input to the encoder. Tokens are prepared by dividing images into patches, converting them to tokens by integrating positional embeddings and then masking tokens by random sampling. No masked tokens are given as input.

O/P → A latent representation is returned as the output of the encoder.

Decoder

I/P → The latent representation (O/P of encoder) is joined with the masked tokens in order (as per the original patch list) and sent as input to decoder.

O/P → a reconstructed image is returned as the output of the decoder.

Pre-training : self-supervised task: get the masked tokens and reconstruct the original image

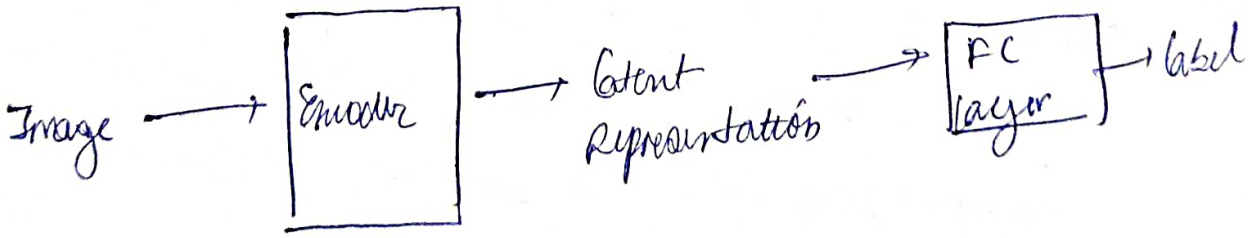
- The pre-training occurs in an asymmetric fashion with only non-masked tokens as encoders Input and entire patch list as decoders Input [latent variables + masked tokens]
- High masking ratio is used to make the task complex enough for the MAE to learn significant features which is then used for other tasks like Object Recognition, Semantic segmentation and so on.

Alternate pre-training Mechanism

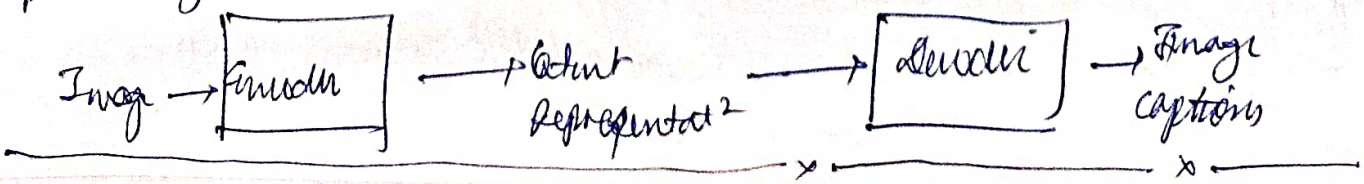
↓

without self-supervision

We can pre-train the model by giving image as input and getting its label as output. A simple classification task which is supervised (as it requires annotated labels) can be used as pre-training mechanism. However, a huge amount of examples will be required to do meaningful pre-training.



For MAE as the architecture, we can use another supervised task like generating image captions using the decoder given an Input image. We will need image captions for all the images.



PART-B 2 (The trick for Few-shot learning)

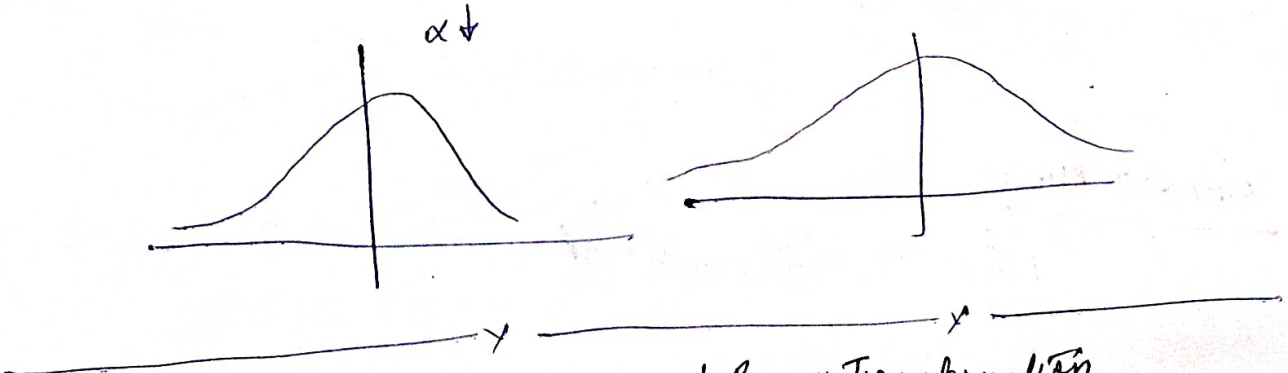
Ques 1. Equation 6,

$$\mu' = \frac{\Sigma_{i \in S_N} \mu_i^{\circ} + \tilde{x}}{k+1}$$

$$\Sigma' = \frac{\Sigma_{i \in S_N} \Sigma_i^{\circ}}{k} + \alpha_i$$

$\alpha \rightarrow$ is a hyper-parameter that determines the degree of dispersion of features sampled from the calibrated distribution.

This hyperparameter indicates how far or how dispersed we are willing to go from the calibrated distribution. This could be thought of as a regularization term. which allows to avoid any overfitting or as in this case, will help avoid any strict distribution. It allows dispersion from the computed variance and allow for a broader variance in the distribution based on the value of the 'α' term.



Ques 2 Section 3.2.1, Tukey's Ladder of Powers Transformation

↓ This power transformation is not used for computing the statistics of the base dataset for the following reasons →

(1) Tukey's Ladder of Power Transformation is used to make the feature distribution more ~~Gaussian~~ Gaussian-like, however, for base classes we start with the assumption that the feature distribution is Gaussian. (20)

(2) Support set and query set are transformed in the ~~paper~~ paper to reduce the skewness of distributions and make them more Gaussian-like. Doing this, the authors are bringing novel classes closer to the base classes. Assuming that base classes are Gaussian and making novel classes more Gaussian-like, it can be visualized as taking novel classes closer to base classes, hence, making it easier to look for similar feature distributions.

(3) Computationally, this makes sense, as assuming base classes to be Gaussian to begin with, the computation ~~expts~~ expense ~~does~~ during while transforming only the novel classes - whose examples are already scarce and thus, doesn't create any significant computation ~~over~~ overhead.

Sol 8. Calibrated mean,

$$\text{numerator} = \Sigma_{i \in S_N} \mu_i + \tilde{x}$$

from equation →

$$\mu' = \frac{\Sigma_{i \in S_N} \mu_i + \tilde{x}}{K+1}.$$

The correct form of numerator is →

$$\left(\Sigma_{i \in S_N} x_i\right) + \tilde{x}$$

Because we are calculating the mean of a distribution. The mean added $\tilde{x}$ is computed off of a distribution (from the novel classes). We can imagine this as adding one more example to the feature set as is depicted in the denominator by "K+1". The mean of the novel classes is added to the statistics of the base class to give calibrated distribution ~~statistics~~.

The paper clearly states that $\tilde{x}$ is nothing but a feature vector of the novel class on which we are computing the calibrated distributions. We are using the nearest base classes as top$k$( ) operator is being used.

Que 4, Empirical Results for table 4.

table 4 → Provides results for miniImageNet → 5 way 1 shot and 5 way 5 shot problems when tukey's transformation and training with generated features is used or not used.

In paper → Authors have stated that without either of the two, the accuracy takes a significant dip leading to the conclusion that both are a requirement for a better performance.

Other than that, we can make the following observations →

(3) There is no better option between the two. That is without Tukey transformation, 5way 1 shot and 5 way 5 shot have quite similar accuracies to when Tukey transformed is performed but with no training from generated features.

Tukey	Generated features	5way 1 shot	5 way 5 shot
✓	×	64.30	81.33
×	✓	63.70	82.26

(4) Furthermore, we can say that with more examples the effect of these 2 ~~of~~ tricks — Tukey transformation and training with generated features ~~was~~ — is reduced.

That is, for 1shot, that is, 1 example in each novel class there is an improvement of 12%

But for 5 shot, that is, 5 examples in each novel class, there is only an improvement of 4% (approx.) when both tricks are used as opposed to done of the ~~trick~~ trick being used.

Ques. Testing of transfer of statistics

Evaluation metrics in the paper for testing → (1) Top-1 accuracy on 2 settings, 5-way 1shot & 5way 5shot

→ (2) t-SNE visualization (visualization of feature distributions)

for the transfer of base to novel classes' statistics, we can use the following testing methods to be more thorough →

(1) use # classes with several samples available as novel classes. We can withhold the available samples using only 1shot or 5shot settings. After transferring the statistics, we can average the computed statistics with that when computed using other available examples. The closer the two are, the better transfer of statistics happened from base to novel classes.

(2) We can introduce multi-domains to increase the complexity and sub-domains to allow for more accurate relationship or closer relationship between base and novel classes. Using sub-domains we can truly say that base and novel classes are similar and thus calibrated distribution can give better accuracy or at least it can utilized during testing to see whether similar domains are being used in the top(1) operator or not and what classes are forming closer relationships.