Ramakrishna Mission Vivekananda Educational & Research Institute
Belur Math, Howrah, West Bengal
School of Mathematical Sciences, Department of Computer Science

End Semester Examination - **CS411: Applications of Computer Vision and Deep Learning**
M.Sc. Computer Science and Big Data Analytics Date: 17-May-2026
Time: 11:00 AM to 02:00 PM
Instructor: Jimut Bahan Pal Max marks: 100

EXAMINATION INSTRUCTIONS

IMPORTANT: Answer all questions completely.

This is a closed-book examination. No reference materials are permitted except for **calculators**.

Time Requirements:

- Minimum time in examination hall: 1 hour
- Maximum examination duration: 3 hours
- Restroom breaks are permitted after 1 hour

Response Guidelines:

- Provide concise, focused answers to all subjective questions
- If clarification is needed, make appropriate assumptions and continue

Prohibited Activities:

- Discussion with other examinees
- Use of unauthorized materials

Remain calm and good luck!

1. **Perceptron** (Q1. Total = **10**)

(a) Sketch the architecture of a standard perceptron, clearly labeling its inputs, weights, bias, and activation function. Define the functional and geometric margins for a linear classifier, and provide their mathematical equations. [1+2+2]

(b) Consider a misclassified data point $(\mathbf{x}_n, y_n)$ during the training of a perceptron, triggering a weight update. Prove that the functional margin of this specific point strictly increases (or remains equal) after a single weight update, such that $\gamma_n^{new} \geq \gamma_n^{old}$. Show all steps of your derivation. [5]

2. **Misc: loss functions, gradients, divergence and convexity of functions** (Q2. Total = **35**)

(a) Using the principle of Maximum Likelihood Estimation (MLE), derive the Binary Cross-Entropy (BCE) loss function from the Bernoulli distribution for a single data point. Then, derive the Categorical Cross-Entropy (CCE) loss function from the Multinoulli (Categorical) distribution. [5+5]

(b) Consider a softmax output layer in a neural network where the pre-activation inputs are denoted as $net_1, net_2, \ldots, net_C$ and the corresponding softmax outputs are $o_1, o_2, \ldots, o_C$. Derive the gradient of a specific softmax output o_k with respect to a pre-activation input net_j. Explicitly show the derivation for two cases:
(1) when the indices are the same ($k = j$), and
(2) when the indices are different ($k \neq j$). [5]

(c) State the mathematical axioms a distance function must satisfy to be considered a valid metric. Explain why the Kullback-Leibler (KL) Divergence is not a valid metric. If we symmetrize the KL divergence, what are the common resulting measures called? Do they become valid metrics? If not, which metric property do they still violate? [3+2+2+2+1]

(d) Consider a minimalist two-layer neural network with a single input x, one hidden neuron with linear activation, and a single output $\hat{y}$. Let the weight of the first layer be w_1 and the weight of the second layer be w_2, assuming no biases. The network's prediction is given by $\hat{y} = w_1 w_2 x$. Given a single training data point $(x = 1, y = 1)$, the Squared Error loss function is defined as $L(w_1, w_2) = (w_1 w_2 - 1)^2$. By deriving the Hessian matrix and analyzing its properties, prove that this neural network loss function is strictly non-convex with respect to its weights (w_1, w_2). ***(Hint: Hessian becomes exactly -4)*** [10]

3. **Autoencoders and VAEs** (Q3. Total = **20**)

(a) Consider a linear autoencoder (an encoder-decoder network with a single hidden layer and linear activations) trained to minimize the Mean Squared Error (MSE) reconstruction loss.
(i) Explain how this architecture inherently performs dimensionality reduction. [2]
(ii) Prove mathematically that the optimal weight matrices of this autoencoder span the same principal subspace as Principal Component Analysis (PCA). [3]

(b) In the context of a Variational Autoencoder (VAE), explain why computing the true posterior distribution $p_\theta(z|x)$ is computationally intractable. Provide the mathematical formulation of the posterior and the marginal likelihood to justify your answer. [5]

(c) Derive the Evidence Lower Bound (ELBO) used in Variational Inference to approximate intractable posteriors.

Once derived, decompose your final ELBO equation to clearly identify and explain the conceptual meaning of its two primary mathematical components. [10]
(Hint: Begin your derivation with the Kullback-Leibler (KL) Divergence between an approximate posterior $q(z)$ ***and the true posterior*** $p(z|x)$***. Apply Bayes' theorem to the true posterior, expand the expectation, and isolate the log marginal likelihood,*** $\log p(x)$***).***

4. **Attention and Transformers for Sequence to Sequence tasks** (Q4. Total = **20**)

   (a) Identify and briefly describe five distinct types of attention mechanisms or attention scoring functions commonly used in deep learning architectures (such as Seq2Seq models and Transformers). [5]

   (b) Consider the Transformer model as originally proposed in "Attention is All You Need" for the task of Neural Machine Translation, specifically translating English to Hindi (e.g., translating the source sequence "How are you ?" to the target sequence "Aap kaise ho?"). Draw a complete, labeled diagram of the encoder-decoder architecture, focusing on the standard model structure and clearly indicating the flow of information for this specific example sentence, showing how the English source input and Hindi target output (during training) are processed with arrows denoting connections. Accompany your diagram with a comprehensive written description that explains how source and target words are converted into continuous embeddings, the specific problem positional embeddings solve, and the exact sinusoidal functions (relating position, embedding dimension, and total dimension for even and odd indices) used to calculate them. Furthermore, explain the conceptual differences and purposes of Encoder Self-Attention, Masked Decoder Self-Attention, and Encoder-Decoder Cross-Attention, and derive the full mathematical equation for Multi-Head Attention, including the generation of queries, keys, values, the scaled dot-product attention, concatenation, and the final linear projection. Conclude your explanation by detailing the architecture of the position-wise feed-forward networks, the operations (linear layer and activation function) that transform the final hidden state into a target vocabulary probability distribution, and the role and placement of residual connections and layer normalization within the model. [5+6+4]

5. **Convolutional Neural Networks and parameter calculations** (Q5. Total = **15**)

Given the PyTorch implementation of a 10-layer Convolutional Neural Network (SimpleCNN) provided below, calculate and tabulate the Output Shape, the Feature Volume (for a single input instance, ignoring the batch dimension), and the exact number of Trainable Parameters (assuming biases are true for all convolutional and linear layers) for each of the 10 distinct sequential operations when provided with a 2D image input tensor of shape [1, 28, 28] (Channels $\times$ Height $\times$ Width). Finally, compute the total number of trainable parameters for the entire network. [7+6+2]

```
1  import torch
2  import torch.nn as nn
3
4  class SimpleCNN(nn.Module):
5    def __init__(rkm):
6      super(SimpleCNN, rkm).__init__()
7      rkm.layer1 = nn.Conv2d(in_channels=1, out_channels=16, kernel_size
   =3, stride=1, padding=1, bias=True)
8      rkm.layer2 = nn.BatchNorm2d(num_features=16)
```

```
      rkm.layer3 = nn.ReLU()
      rkm.layer4 = nn.MaxPool2d(kernel_size=2, stride=2)
      rkm.layer5 = nn.Conv2d(in_channels=16, out_channels=32, kernel_size
   =3, stride=1, padding=1, bias=True)
      rkm.layer6 = nn.ReLU()
      rkm.layer7 = nn.MaxPool2d(kernel_size=2, stride=2)
      rkm.layer8 = nn.Flatten()
      rkm.layer9 = nn.Linear(in_features=1568, out_features=64, bias=True)
      rkm.layer10 = nn.Linear(in_features=64, out_features=10, bias=True)

    def forward(rkm, x):
      x = rkm.layer1(x)
      x = rkm.layer2(x)
      x = rkm.layer3(x)
      x = rkm.layer4(x)
      x = rkm.layer5(x)
      x = rkm.layer6(x)
      x = rkm.layer7(x)
      x = rkm.layer8(x)
      x = rkm.layer9(x)
      x = rkm.layer10(x)
      return x
```