# CNN Lytical Image Classification Prep

Suppose you are an interviewer, and I am an interviewee. I have written a project in my resume. you have to check my skill set and authenticity If I have actually done it or not by question giving a scenario-based question and asking in-depth question CNN-Lytical | Image Processing |Web And Coding Club-IIT Bombay [Jun'22–July'22] • Developed an image classification model from scratch using NumPy on MNIST dataset, achieving impressive accuracy of 96% • Developed and deployed Convolutional Neural Networks (CNNs) with PyTorch to classify images from the CIFAR-10 dataset • Developed an image segmentation model with the UNet architecture on the Carvana dataset, and achieving 0.92 IoU accuracy

**ChatGPT said:**

Great! Let's start the interview. I'll focus on assessing your understanding of CNNs, image processing, and deep learning fundamentals.

---

**Interviewer:**
Rajesh, I see you've worked on CNN-based image classification and segmentation models. Let's dive into some scenario-based questions to evaluate your understanding.

**Scenario 1: CNN from Scratch (NumPy - MNIST)**

1. You mentioned implementing a CNN from scratch using NumPy for the MNIST dataset. Since NumPy doesn't have automatic differentiation, how did you implement backpropagation for weight updates?
2. Given that MNIST has grayscale images, how would your CNN architecture differ if you had to train it on a dataset with RGB images?
3. Suppose your model achieves 96% accuracy, but during training, you notice that validation loss is fluctuating a lot while training loss keeps decreasing. What could be the possible reasons, and how would you fix them?

---

**Scenario 2: CNNs with PyTorch (CIFAR-10 Classification)**

4. You worked with PyTorch for CIFAR-10 classification. Can you explain the role of the DataLoader class in PyTorch, and how it helps in training?
5. Suppose after training your CNN, you notice that it performs well on training data but poorly on test data. What steps would you take to address this issue?
6. If you wanted to speed up training using a GPU, what changes would you need to make in your PyTorch code?

---

**Scenario 3: Image Segmentation (U-Net - Carvana Dataset)**

7. You implemented a U-Net model for segmentation. Can you describe why skip connections are important in U-Net?
8. Suppose after training, your model achieves 0.92 IoU but struggles with segmenting small objects in the image. What architectural changes or techniques would you consider to improve performance on small objects?
9. Let's say your model is taking too long to infer on a single image. How would you optimize inference speed while maintaining accuracy?

---

**General Deep Learning & Image Processing**

10. If you were to deploy your image segmentation model for real-time applications, what optimizations would you consider?
11. Explain how batch normalization helps in deep networks. Would you use it in every layer of your CNN? Why or why not?
12. What are some alternative architectures to U-Net for segmentation, and how do they compare in terms of performance?

---

**Interviewer Note:** If you confidently and correctly answer these questions with in-depth reasoning, it would strongly indicate that you genuinely worked on these projects. However, struggling with key implementation details, optimization techniques, and architectural reasoning could suggest a lack of hands-on experience.

**You said:**

Explain how batch normalization helps in deep networks. Would you use it in every layer of your CNN? Why or why not? I don't know this..can you explain it?

11/11

**ChatGPT said:**

Sure! Let me break it down step by step.

**What is Batch Normalization?**

Batch Normalization (BN) is a technique used in deep neural networks to **stabilize and accelerate training**. It helps by normalizing the inputs to each layer so that they have a more consistent distribution.

**How Does It Work?**

# CNN Lytical Image Classification Prep

For each mini-batch during training:

1. **Calculate Mean & Variance:** Compute the mean and variance of the activations for that batch.
2. **Normalize:** Subtract the mean and divide by the standard deviation to get a standardized activation.
3. **Scale & Shift:** Introduce two learnable parameters, **γ (scale)** and **β (shift)**, which allow the network to learn the best representation for the data.

Mathematically, for each feature xxx in the batch:

x^=x−μσ\hat{x} = \frac{x - \mu}{\sigma}x^=σx−μ y=γx^+βy = \gamma \hat{x} + \betay=γx^+β

where μ\muμ is the batch mean and σ\sigmaσ is the batch standard deviation.

---

### Why is Batch Normalization Useful?

1. **Faster Convergence** – Helps gradients flow more smoothly, reducing the number of training epochs required.
2. **Reduces Internal Covariate Shift** – Prevents drastic changes in activations across layers, leading to more stable learning.
3. **Acts as a Regularizer** – Adds slight noise to activations, reducing the need for Dropout in some cases.
4. **Allows Higher Learning Rates** – Since activations are well-scaled, we can use a larger learning rate without the risk of divergence.

---

### Would You Use Batch Normalization in Every Layer?

**Not necessarily.** While BN is useful, using it everywhere isn't always ideal.

✅ **Where to use it?**

- After **convolutional** layers (before activation functions like ReLU)
- After **fully connected** layers in deep networks

🚫 **Where not to use it?**

- In the **output layer** (e.g., softmax or sigmoid) because we don’t want to distort probability distributions.

# CNN Lytical Image Classification Prep

- When using architectures like **ResNets**, where residual connections already help stabilize training.
- For very small batch sizes (e.g., < 8), BN can introduce instability since batch statistics vary too much.

---

**Alternative to BN?**

- **Layer Normalization** – Works better for NLP tasks where batch sizes are small.
- **Group Normalization** – Useful when batch sizes are too small for BatchNorm.

Would you like a practical example in PyTorch? 😊