**Model Architechture and Algorithm:**

**Step-by-Step Layers of CNN:**

1. **Conv Layer 1:**
   * 32 filters, **kernel size 3x3**, **stride = 1**, **padding = 1**.
   * Extracts low-level features (edges, textures).
   * **ReLU Activation:** Applies max(0, x) to remove negative values and introduce non-linearity.
2. **Max Pooling 1:**
   * Pool size: 2x2, Stride: 2.
   * **Reduces the feature map size from 32x32 → 16x16** (halves dimensions).
   * Keeps important features while reducing computational cost.
3. **Conv Layer 2:**
   * 64 filters, **kernel size 3x3**, **stride = 1**, **padding = 1**.
   * Extracts more complex patterns.
   * **ReLU Activation:** Keeps positive values, removes negative ones.
4. **Max Pooling 2:**
   * Pool size: 2x2, Stride: 2.
   * **Reduces size from 16x16 → 8x8**.
5. **Flatten Layer:**
   * Converts the **3D feature map (64x8x8) → 1D vector (4096)**.
6. **Fully Connected Layer 1:**
   * **128 neurons, ReLU activation.**
   * Learns complex patterns.
7. **Fully Connected Layer 2:**
   * **10 neurons, Softmax activation.**
   * Converts output into class probabilities.

**3. Training Process**

For each **epoch** (repeat for num\_epochs times):

1. **Set model to training mode (model.train()).**
2. **Loop through batches of images:**
   * Move images to GPU (if available).
   * Forward pass:
     + Pass images through CNN.
   * Compute **loss** using CrossEntropyLoss.
   * Backpropagation:
     + Compute gradients.
   * Update weights using **Adam optimizer**.
   * Track training loss & accuracy.
3. **Evaluate on test data (model.eval()).**