Essential Steps for CNN and their needed implementations for RISC V:

1. **Input Layer**
    - This is where the image data goes in.
    - Each image is represented as a grid of pixel values.
    - Unsure what size image we want, 28x28 is the typical but it might need to be reduced for computation time

RISC V Implementation:

- Load image data from memory using LW (load word) or LB (load byte) if stored as bytes.
- Store it into registers or arrays in memory for processing.
- Looping is needed for multi pixel images.

2. **Convolutional Layer**
    - This layer applies filters (kernels) to scan parts of the image.
    - Each filter is a small matrix that slides across the image matrix.
    - The filter detects patterns like edges, shapes, and textures depending on values inside filter matrix.
    - The result is a new version of the image called a feature map, highlighting the details that each filter corresponds to.

RISC V Implementation:

- Nested loops to slide the filter across the image.
- Perform multiply-accumulate (MAC) operations using MUL and ADD.
- Put values into memory.

3. **Activation Function**
    - After convolution, relu activation function is applied.
    - Keeps positive values and sets negatives to zero.
    - Basically needed (boosted accuracy from 12% to ~50% when softmax wasnt used).

RISC V Implementation:

- Check register and if <0, it should be forced to zero.

4. **Flattening**
    - The feature maps are still in a grid format (like an image).
    - Flatten them into a single long vector to pass into the next layers.

RISC V Implementation:

- Loop the 2D array and append these values into a new 1D array.

5. **Fully Connected Layer (Dense Layer)**
    - This layer connects every neuron to every other neuron.
    - It takes the flattened values and learns to recognize high-level patterns (like digits, faces, or objects).
    - It uses weights and biases to adjust importance for different features.

RISC V Implementation:

- Load the weights and inputs, then begin summations.
- Loops for all values.

6. **Output Layer**
    - This is the layer that makes the prediction, last layer must represent the number of outputs (MNIST digits needs 10 neurons on last layer).
    - Apply softmax or sigmoid depending on what is easier to implement (both are above 90% accuracy).

    RISC V Implementation:

    - Approximate e for softmax/sigmoid usage
    - Output is a logit until softmax is applied (softmax makes it a probability)
    - 10 value array for storing logits
    - Usage of argmax is needed to get value and index (index is the digit its making the prediction for)