2 Convolutional NN

2.1 Class definition

- nn.Conv2d and nn.MaxPool2d in PyTorch affects the dimensions of inputs and outputs

1. **2D Convolution (nn.Conv2d)**

The nn.Conv2d layer applies a set of filters (also called kernels) to an input image, producing an output feature map that captures local spatial patterns in the input. Each filter "slides" over the input, performing element-wise multiplications followed by summation, and produces one output channel.

The output height (H_out) and width (W_out) of a convolutional layer are given by:

H_out = ((H_in + 2 * padding - kernel_size) / stride) + 1

W_out = ((W_in + 2 * padding - kernel_size) / stride) + 1

2. **2D Max Pooling (nn.MaxPool2d)**

The nn.MaxPool2d reduces the spatial dimensions of the input by taking the maximum value in each specified window. Pooling helps reduce computational complexity and makes the model more robust by focusing on the most salient features.

The output height (H_out) and width (W_out) of a convolutional layer are given by:

H_out = ((H_in + 2 * padding - kernel_size) / stride) + 1

W_out = ((W_in + 2 * padding - kernel_size) / stride) + 1

2.2 Fitting

Obtained values:

MLP Epoch 10: Train Loss = 0.1086, Test Loss = 0.3983, Accuracy = 86.50%

CNN Epoch 10: Train Loss = 0.1138, Test Loss = 0.1864, Accuracy = 92.00%

Trainable parameters in MLP model: 109386

Trainable parameters in CNN model: 14682

Conclusions from results:

1. CNN outperforms MLP in accuracy (92%) for 10 epochs

2. CNN better suited for images to learn features like edges, textures. MLP treats each pix. as features so it can’t capture complex spatial features properly.

3. MLP (109386) has more parameters over CNN (14682) due to convolutional layers of image data. CNN learn data efficiently over image reducing overfitting and more efficient for image data.