Introduction:

Dense Neural Network:

The DNN is a feedforward model. The data is inserted into the input layer where it is then sequentially passed through the hidden layers. Each neuron in a layer applies a weighted sum to the inputs from the previous layer. The activation function used is a ReLU which allows the network to extract patterns. The output layer then receives this data to generate a classification of the numbers.

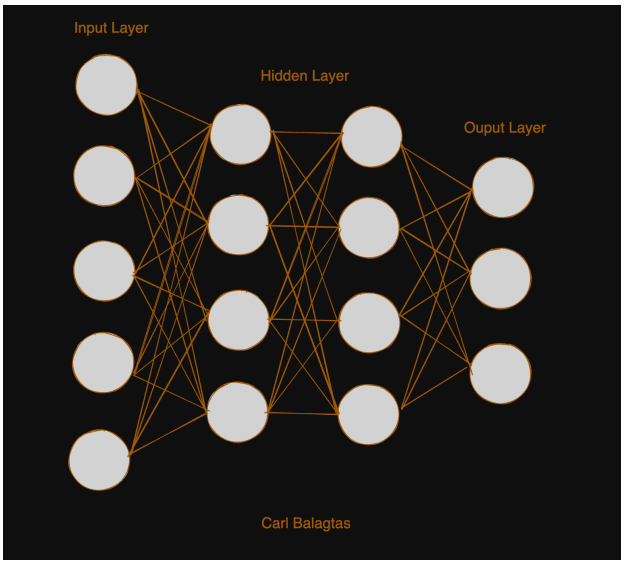


Figure 1

Convolutional Neural Network:

CNN implementation uses convolutional and max pooling layers to process the input. The convolutional layer applies a filter/kernel to extract features from the input image. This generates a feature map through the scanning process. After each convolution, the max pooling layer reduces the dimensions of the feature map which helps decrease computational complexity.

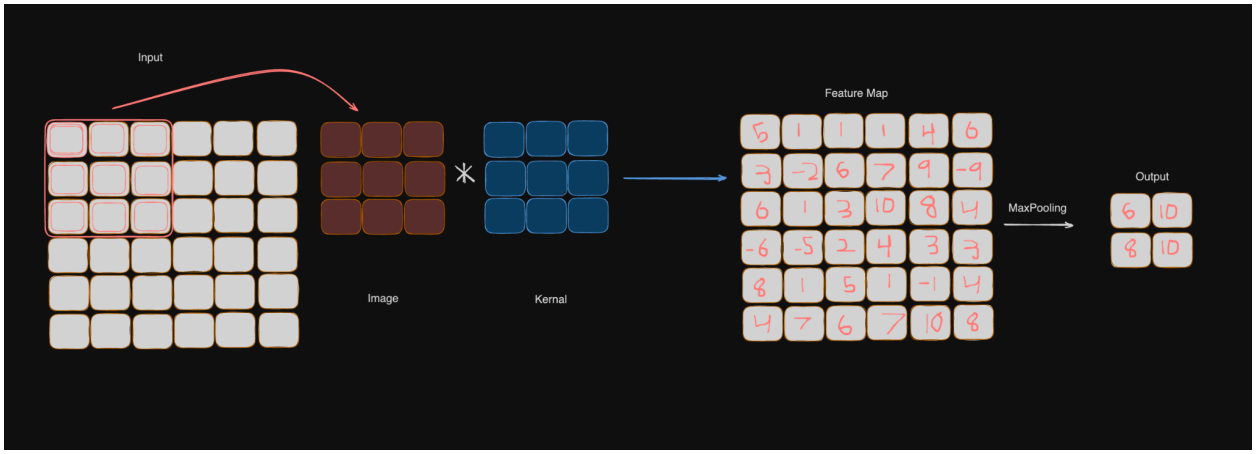


Figure 2

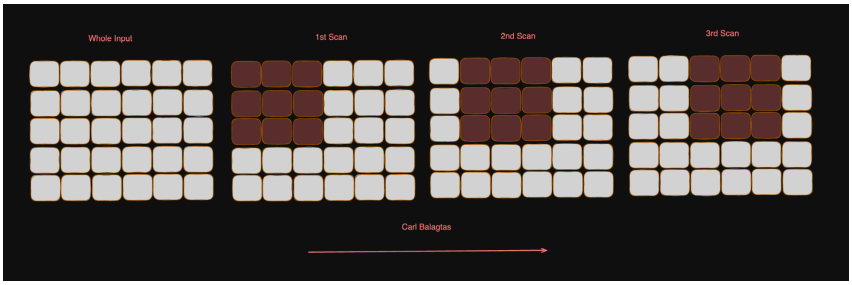


Figure 3

Recurrent Neural Network:

RNN neurons are unique because they have recurrent connections that loop back to themselves or previous layers. This design enables the network to incorporate information from past outputs when analyzing the current input. Each node as shown in figure 4 loops back to a previous or current layer.

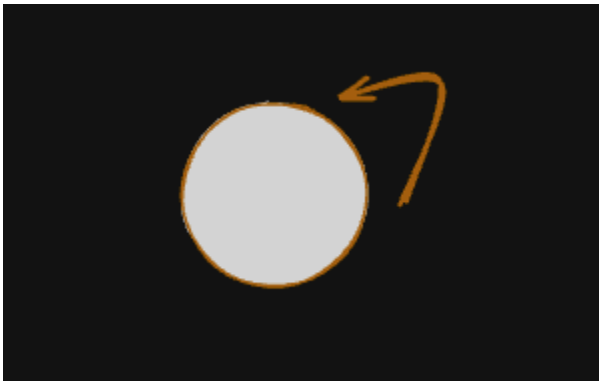


Figure 4

Convolution Neural Network:

The kernel size, stride and padding are parameters that influence how an input image is processed. The kernel size determines the dimension of the filter sliding over the image, the larger the kernels the more information. Referring to **figure 3**, the example size of the kernel is a 3x3 and it will keep sliding but if the kernel size were to increase it would cover more space per slide. The smaller the kernel the more it focuses on finer details but may miss larger features.

The stride defines how many pixels the kernel moves with each step. In **figure 3**, the stride used as an example is 1 meaning that every shift of the filter it moves by 1 pixel. If the stride was to increase the filter would move by larger increments making the image processing faster but with less details.

Padding adds extra pixels around the input image so that when convolution happens important information won't be lost.

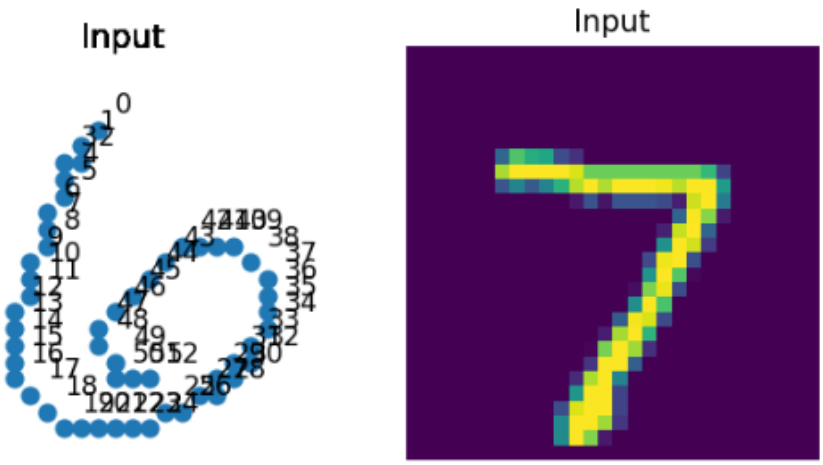
CNN is effective for image processing due to the kernels being able to capture essential features like edges and textures. Since the input data is scanned fully by the convolutional layers, meaning that the features can get detected no matter the position of the features in the image which makes it good for object recognition and classification.

Recurrent Neural Network:

The input structure of the RNN differs from the CNN because in the CNN the input is typically a 4D tensor(instance_count, height, width, input_channel_count) representing a batch of 2D images. While the input of the RNN, particularly a GRU is a 3D tensor(instance_count,time_step_count,input_dimensionality) capturing sequence over time, with each time step consisting of a vector of features. In this case CNN process each instance as a 2D image while RNN sees each instance as coordinates.

```
#CNN
layers = [
    ls.Convolution2D(name='conv2d', input_channel_count=1, output_channel_count=32, kernel_size=3, stride=1, padding=1, activation_function='relu'),
    ls.MaxPooling2D(name='pool2d', pooling_size=2, stride=2, padding=0, activation_function='none'),
    ls.Convolution2D(name='conv2d_1', input_channel_count=32, output_channel_count=64, kernel_size=3, stride=1, padding=0, activation_function='relu'),
    ls.MaxPooling2D(name='pool2d_1', pooling_size=2, stride=2, padding=0, activation_function='none'),
    ls.Flatten(name='flat'),
    ls.Dense(name='dense', input_dimensionality=2304, output_dimensionality=10, activation_function='softmax')
]
```

```
#RNN
layers = [
    ls.GatedRecurrentUnit(name='gru', input_dimensionality=2, output_dimensionality=32, return_state_sequence=False),
    ls.Dense(name='dense', input_dimensionality=32, output_dimensionality=10, activation_function='softmax')
]
```



The gated recurrent unit(GRU) operates on the input sequences by utilizing the memory of previous runs to retain contextual information. GRU has the update age, reset gate, candidate hidden state and new hidden state. The update gate controls the retention of the previous hidden state while the reset gate determines how much of the previous state influences the candidate hidden state which is influenced by the current input. The sigmoid activation function is used for the gates due to its output range(0,1) allowing for a smooth transition in information flow whereas the tanh function is used for the candidate hidden state because its range of (-1,1) captures both positive and negative influences. This combination enables the GRU to learn long term dependencies by dynamically managing information at each time step.