Use of Linear and Matrix Algebra in Convolutional Neural Networks

Aakash Gireesh Tripathi and Leela Pavan Kumar Parvathaneni
Linear and Matrix Algebra

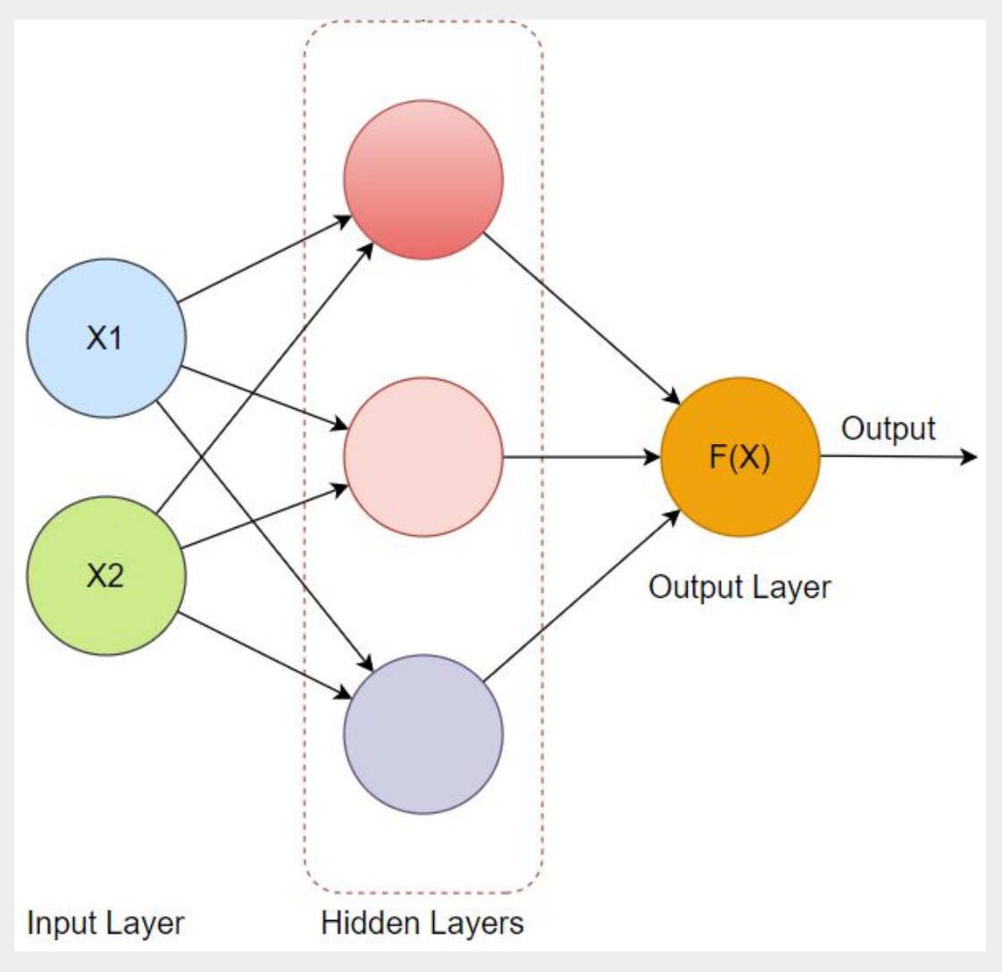
UNIVERSITY OF SOUTH FLORIDA

Abstract

Here, we will discuss the use of linear and matrix algebra in a class of deep neural networks called convolutional neural networks(CNNs). We also demonstrate the advantages of using CNNs in the task of classifying handwritten digits and compare its performance with a basic fully connected neural network (FCNN).

Background

➢**Neural Networks :** A typical neural network is composed of an input layer, one or more hidden layer and an output layer.Input layer is responsible for receiving the input data.The hidden layers are responsible for the processing the input data and extracting features from it.The output layer is responsible for producing the output.



➢**Convolution :** It is an operation which can be done by using filter called kernel on an input data.The result of the convolution operation is a feature map.

Convolutional Neural Networks

➢**Activation Function :** The activation function is a nonlinear function that is applied to the output of the linear transformation. It is responsible for introducing nonlinearity to the neural network. we use the ReLU activation function for this project.

$$\sigma(x) = \max(0, x),$$

➢**Forward Propagation :** The output of the convolution layers is a 2D matrix. This matrix is then flattened into a 1D vector and passed to the fully connected layers. To perform the linear transformation, the input vector is multiplied by a weight matrix and then added to a bias vector. The output of the linear transformation is then passed to a non-linear activation function. The output of the activation function is then passed to the next layer. Given below is the equation for forward propagation process for a single layer.

$$y = \sigma(Wx + b)$$

➢**Error Calculation :** The error is calculated by comparing the output of the neural network with the target output. The error is then used to adjust the weights of the connections between neurons. For this project, we use the cross-entropy loss function. It is defined as

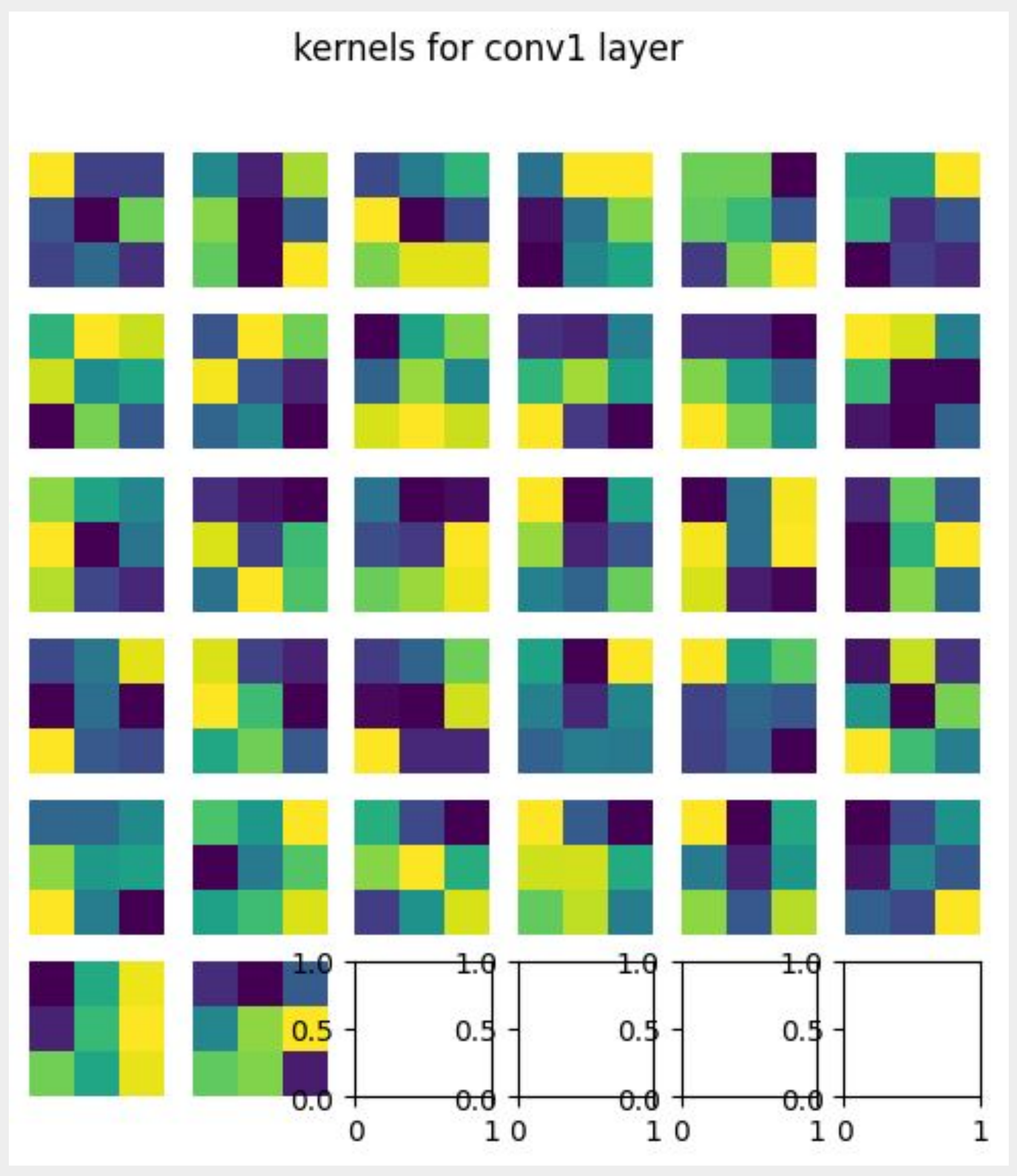
➢**Backward Propagation :** It is the process of adjusting the weights of the connections between neurons. The weights are adjusted by calculating the gradient of the loss function with respect to the weights. An optimization algorithm is then used to update the weights. The optimizer used in this project is the Adam optimizer.

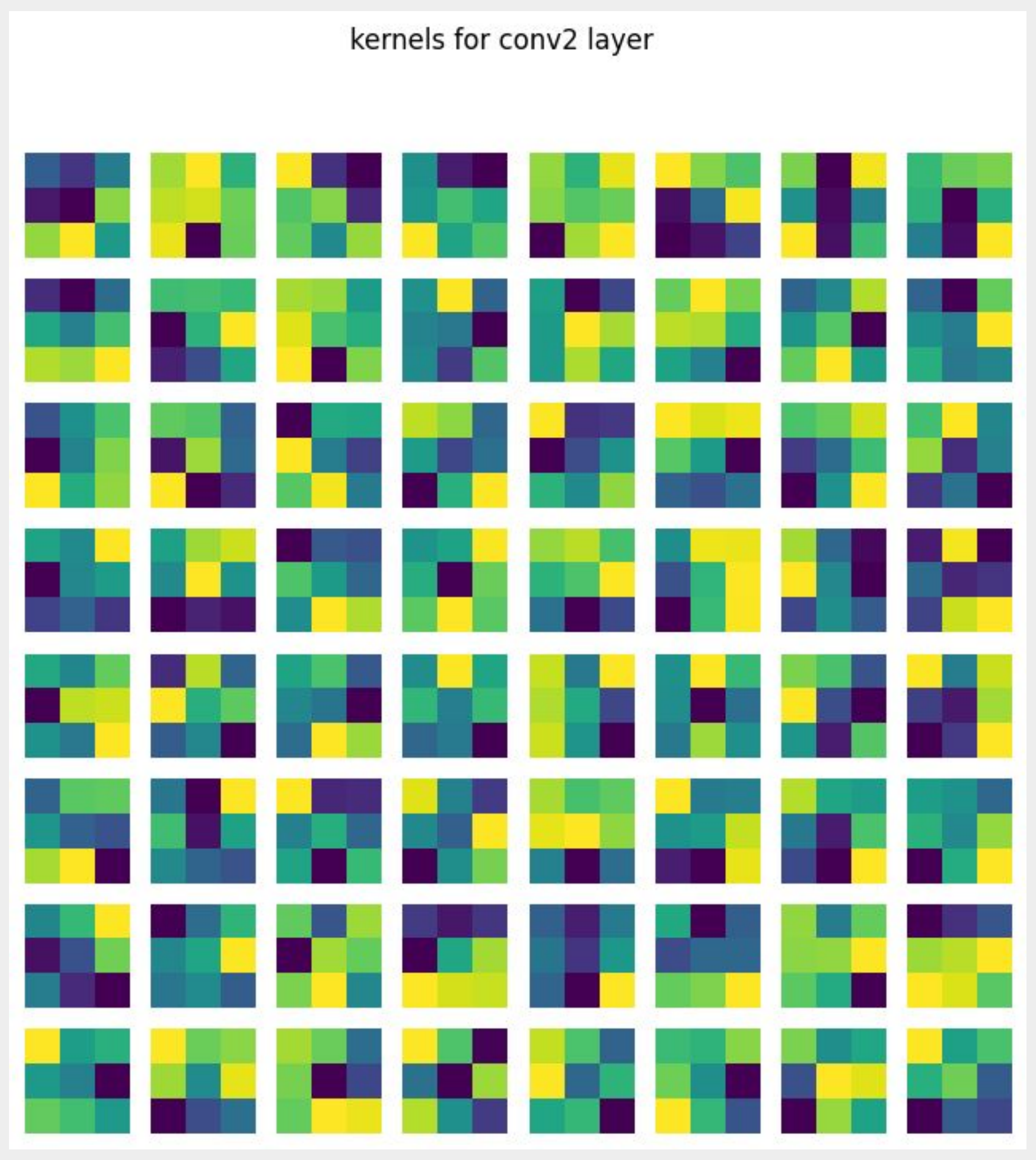
Model	Loss	Accuracy
CNN	0.03	99.07
FCNN	0.10	96.94

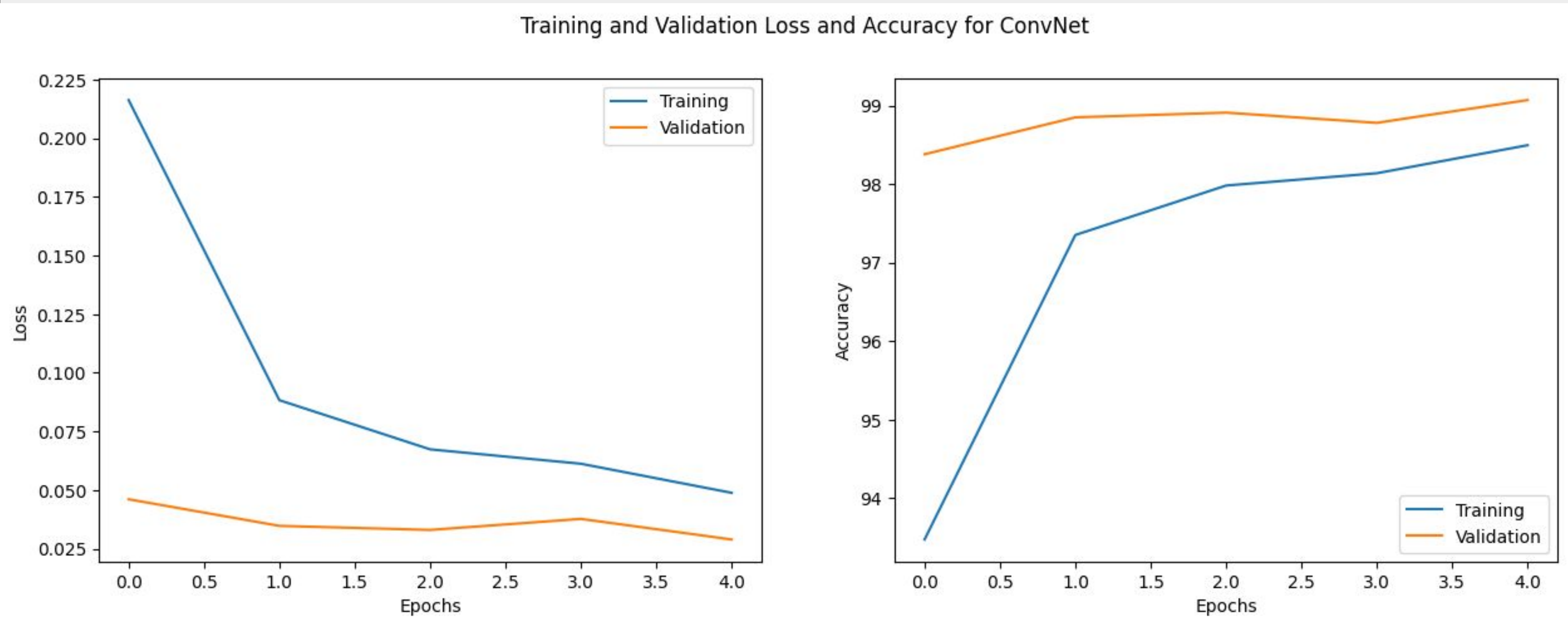
TABLE I: Results of the CNN and the fully connected network

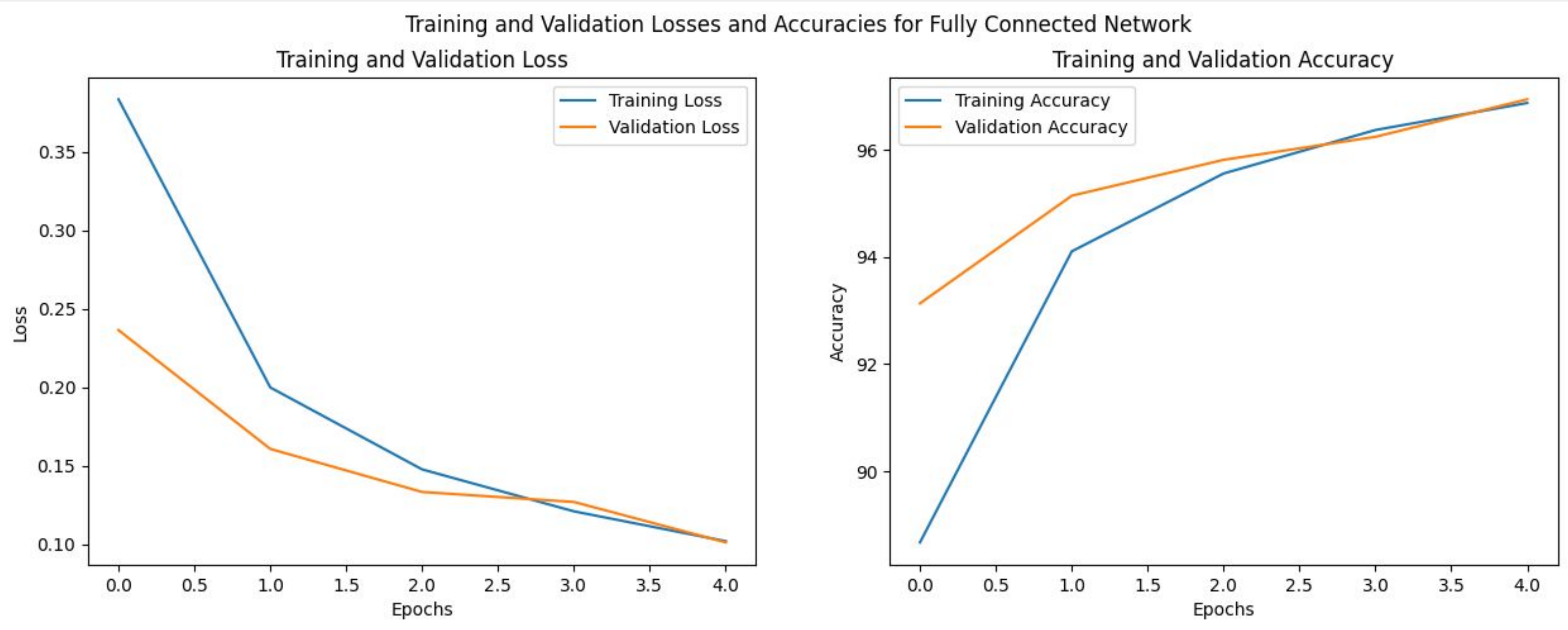
Conclusion

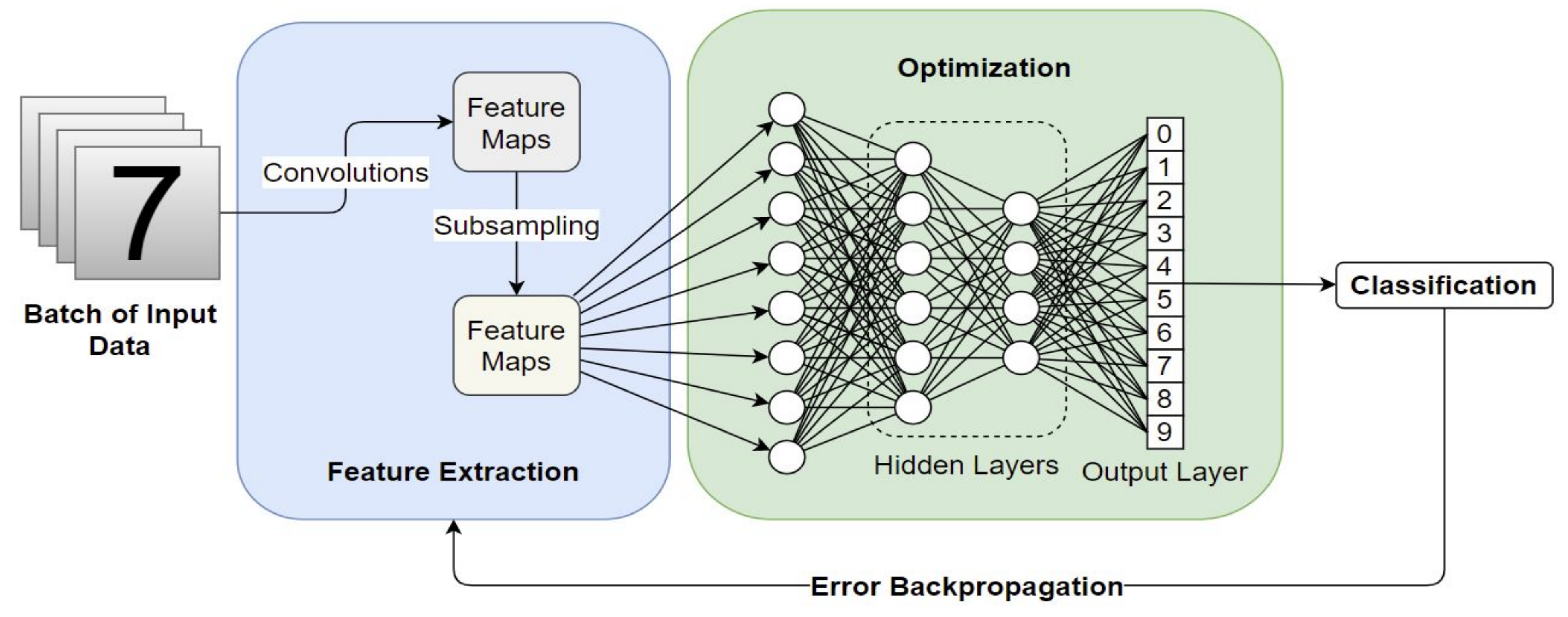
In this project, a CNN model was implemented and trained on the MNIST dataset. The CNN model was able to achieve an accuracy of 99.07 on the test set. The fully connected network was able to achieve an accuracy of 96.94 on the test set.

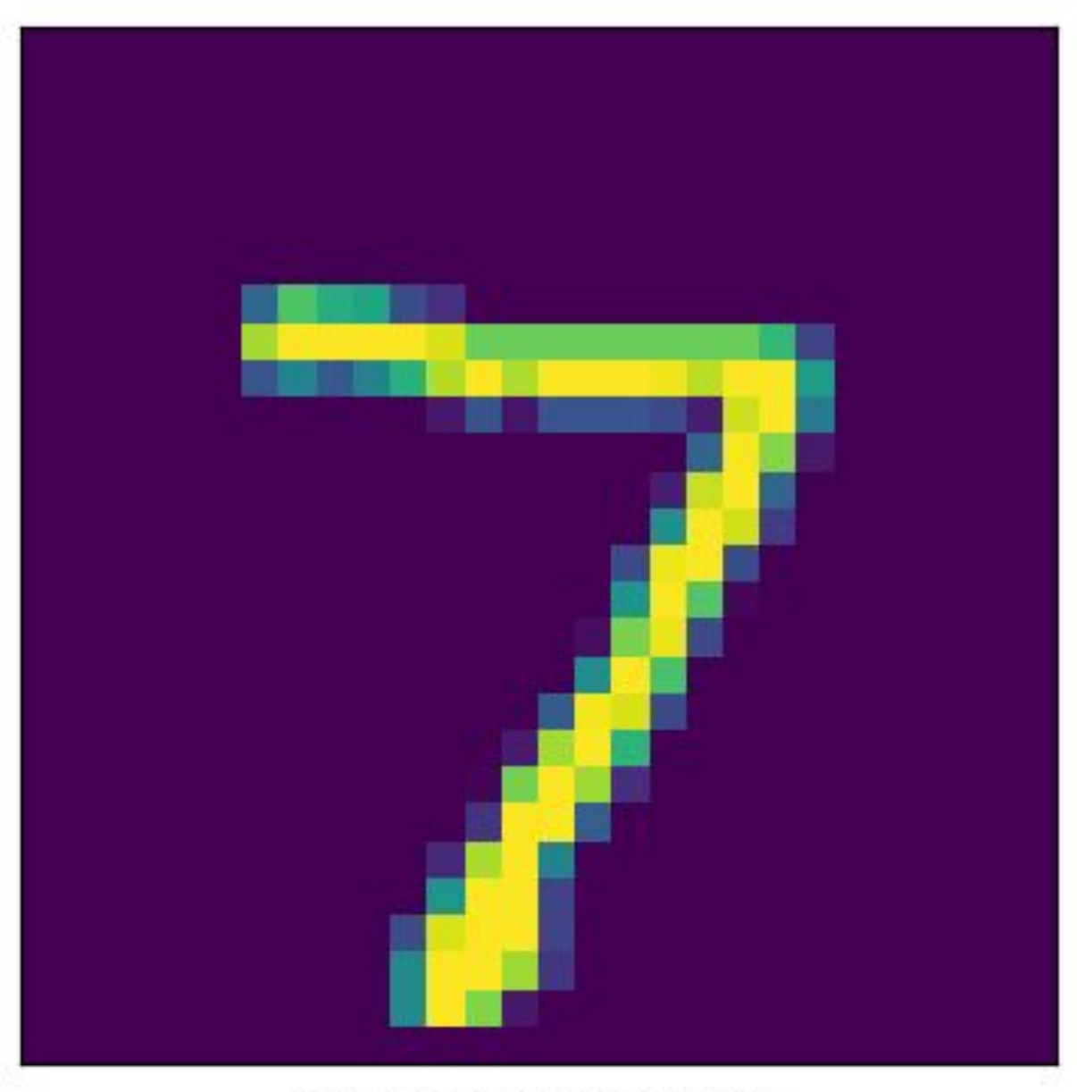


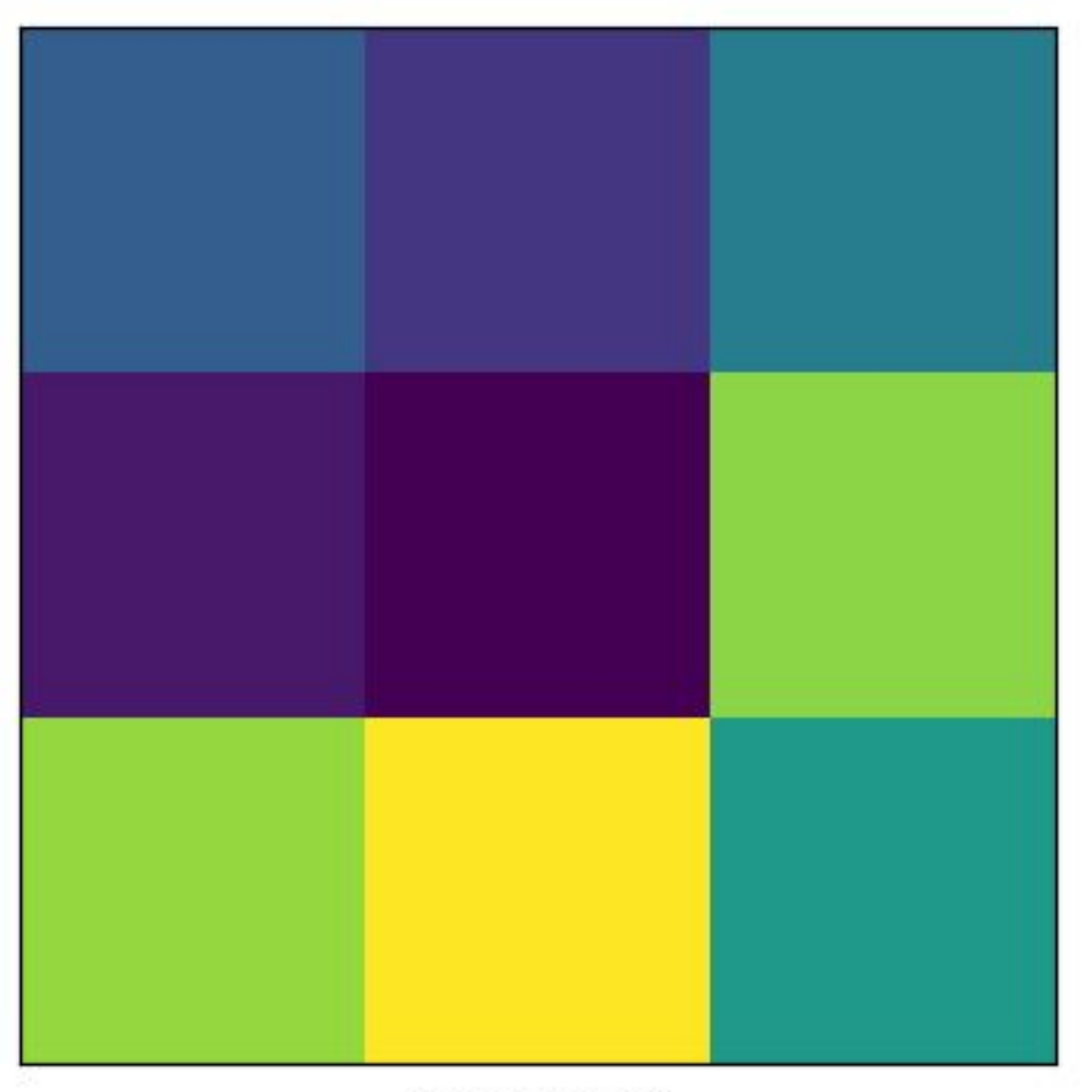


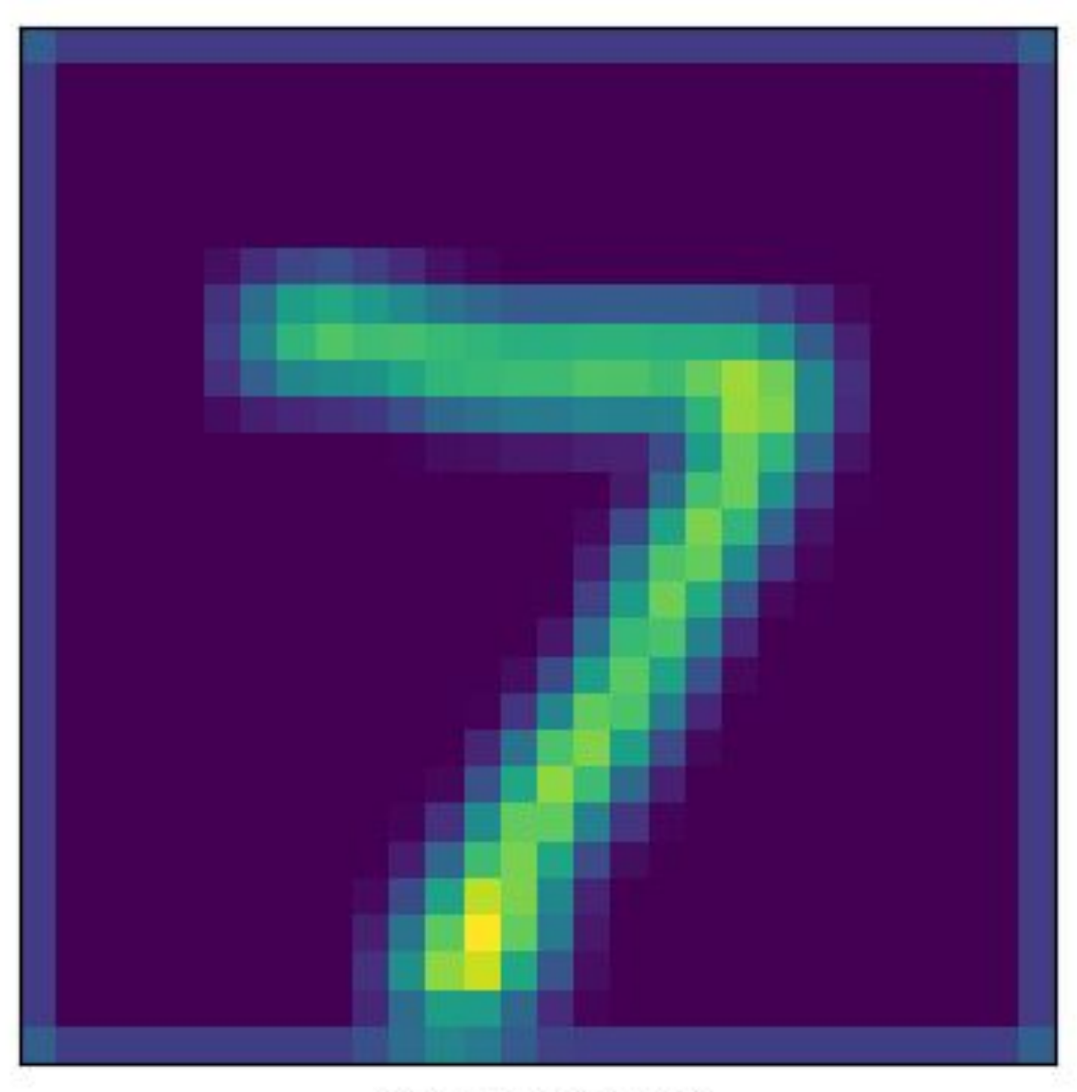







Original Image (28x28)


Kernel (3x3)


Output (28x28)