Exercise 6

## MLP

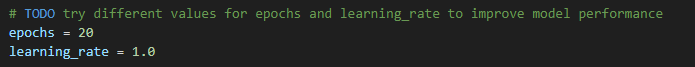
A screenshot of a computer

AI-generated content may be incorrect.



The accuracy with the layers above results in a test accuracy of 0.9810 for the Multilayer perceptron model

The learning rate and epochs used for this result are as follows



## CNN

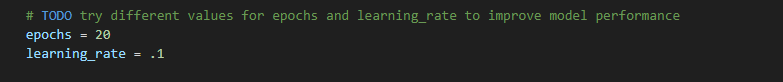
A screenshot of a computer program

AI-generated content may be incorrect.



The accuracy with the layers above results in a test accuracy of 0.9909 for the Convolutional Neural Network model.

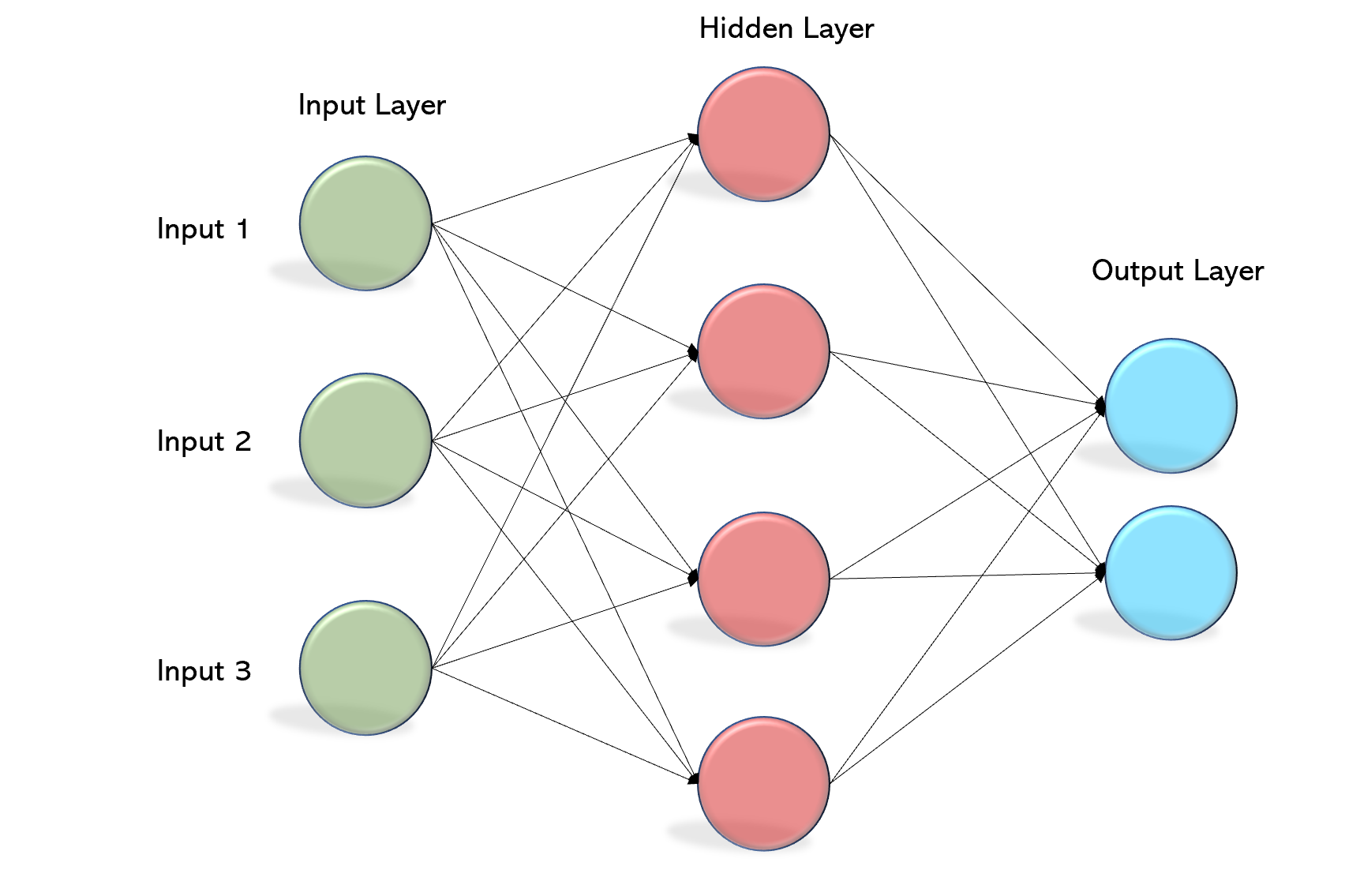
The learning rate and epochs used for this result are as follows



# Comparison

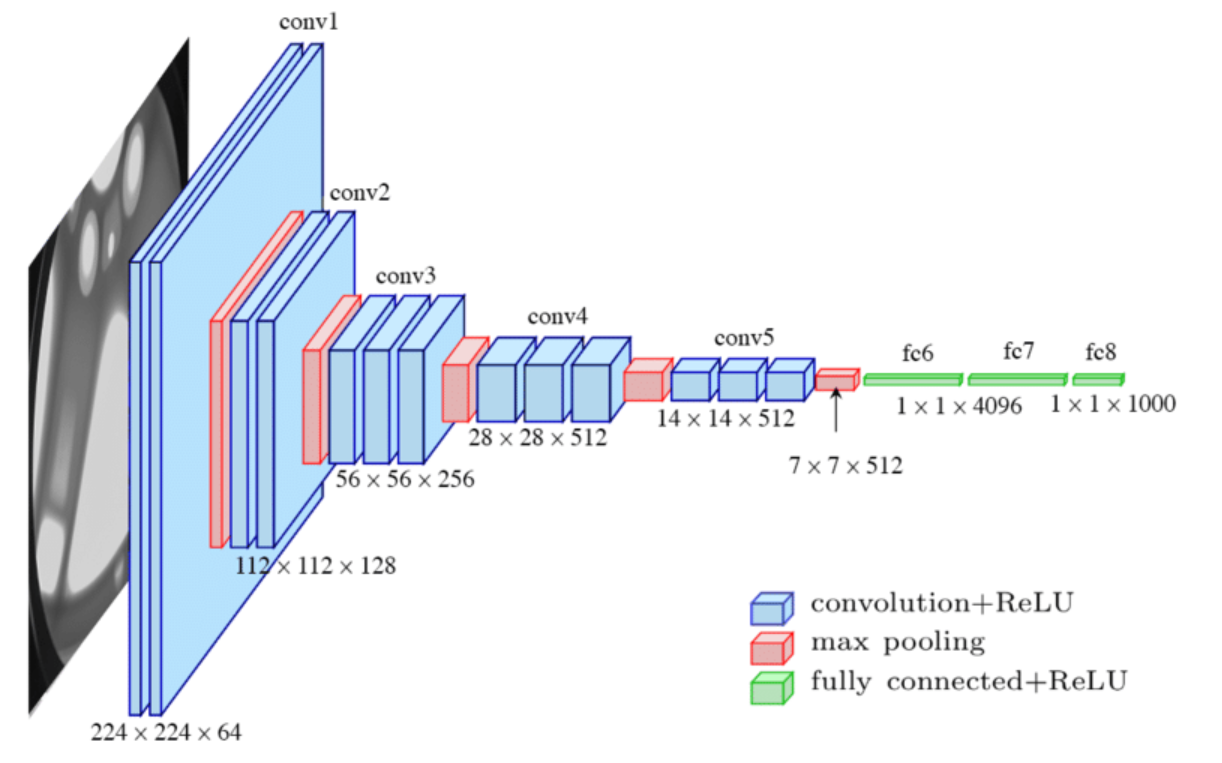
The MLP achieved a maximum accuracy of around 97%-98% after tuning the number of hidden layers and neurons with a new learning rate. The CNN on the other hand got up to 98% - 99% and generalized better than the MLP, which can be seen in the validation\_accuracy field after each epoch.

Since the MNIST images are too simple to classify for both models, as long as both models have reasonable layout for the layers, you can expect high accuracy from both. But for more complex datasets containing images, CNNs can significantly outperform MLPs.



This figure above is a MLP layout where each neuron is connected to every other neuron, making it a fully connected neural network. A perception has weights, inputs and biases (usually has an activation function as well which does not need to be considered for parameter calculations because it is simply for transformation purposes). Since every layer is fully connected the parameter can be calculated by multiplying previous layer neuron count with current layer neuron count and adding a bias of 1 for each neuron in the layer. For our MLP model we had for the task the total number of parameters is

The figure below is a CNN model. A CNN consists of convolutional layers as well as a fully connected layer (towards the end). The figure below depicts a CNN with increasing filters starting from 64 and continuing through 128, 256 and ending at 512 filters before the fully connected layers. This transition from small number of filters to more filters towards the end is to let the first convolution layers pick up the low-level details first and move towards higher level details towards the end. These filters (kernels) are randomized kernels in Keras where they are gradually updated with backpropagation and gradient descent for refinement. Each filter creates its own feature map, so if we specify 64 filters for the first convolution layer as shown in the figure below then we get 64 different feature maps within the same layer. The kernel size on the other hand focuses on the size of the kernel that will be used for cross-correlation (Yes, CNNs don’t actually use convolution, it’s not necessary, they use something similar called cross-correlation where the flipping of kernels is skipped because filters are learned instead of being predefined). CNNs are good for images because they extract spatial features, because each filter operates on a small region making finding localized patterns such as edges and corners easier. They are also robust against translation of images because of the same filter being applied across different parts of the image.

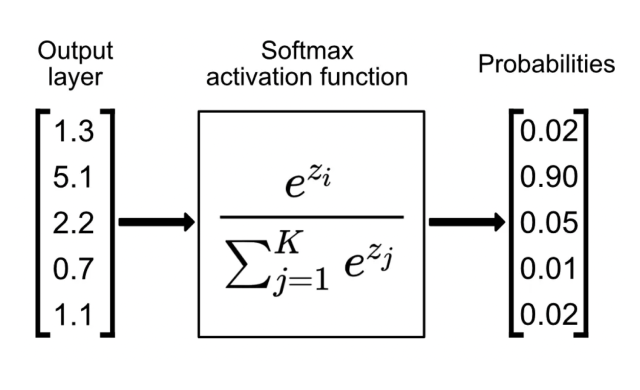


Similar to the MLP calculation, we can do the same kind of calculation for our CNN model. In the convolution layer, we have kernels and channels to keep track of instead. Since kernels have their own weights and biases that are learned during training and images could have input channels (RGB, greyscale (in our case of the MNIST dataset which results in value of 1 for input channel) etc). Additionally, a layer also operates with different filters, for example in our layer 1, we have 32 filters which will create 32 feature maps (The thickness of the rectangular shapes of the convolution layers in the figure above).

In conclusion, we see that CNNs are better for such tasks as image classification. Parameter count is also considerably lower, even though in my case it became a bit more because I used a lot more layers and used a lot of features on the CNN while the MLP only had two hidden layers.

## Categorical cross entropy and SoftMax

The output layer is a SoftMax activation function with 10 units (one for each digit). Softmax ensures the output values represent probabilities ranging from 0 to 1. The model predicts the class with the highest probability.



The categorical cross entropy loss is used to penalize bad predictions. If the model made a poor prediction, the loss would be higher. For example if it said the number 5 was 5 with a predicted probability of .75 the loss would be

