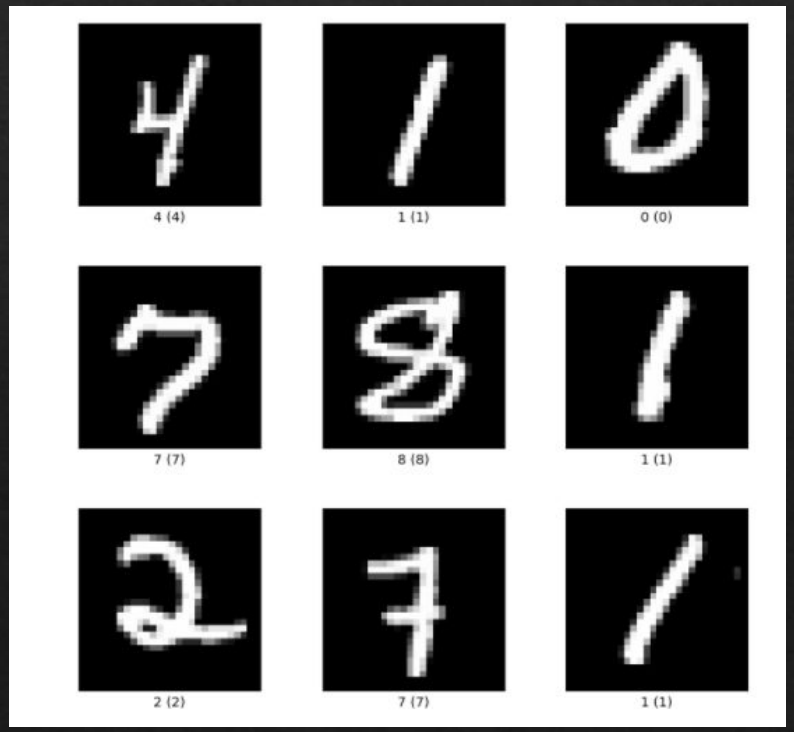
Handwriting Recognition

Team Members

- Tejassingh Patoi-19BAI10087
- Anand Sinha -19BAI10104
- Mansi Shrivastav- 19BAI10175
- Mandavi Pandey- 19BAI10133

Introduction

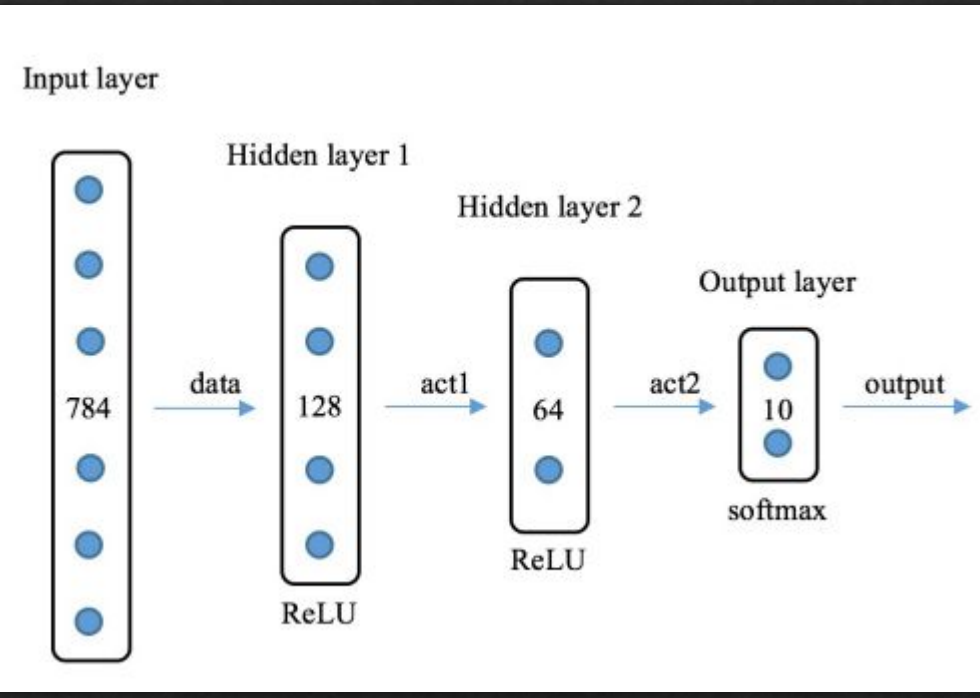
- A Machine Learning algorithm to recognize and convert handwritten text to digital text.
- We are using the MNIST Dataset of handwritten digits

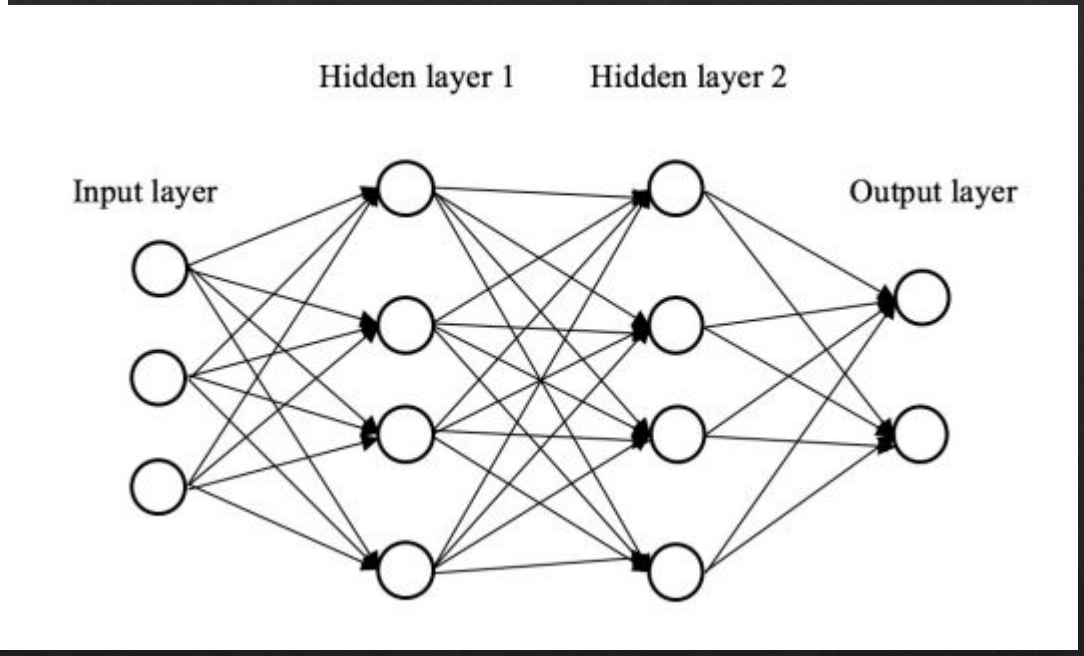


Model

- We start off with probably the most basic classifier, the logistic regression, to be specific multinomial logistic regression as it is a multiclass case.
- Basically, logistic regression is a feed-forward neural network without a hidden layer, where the output layer directly connects with the input layer. In other words, logistic regression is a single neuron that maps the input to the output layer. Theoretically, the neural networks with an additional hidden layer between the input and output layer should be able to learn more about the relationship underneath.
- We have just achieved 91.3% accuracy with a single-layer neural network model. Theoretically, we can obtain a better one with more than one hidden layer.
- Weight optimization in feed-forward deep neural networks is also realized through the backpropagation algorithm, which is identical to single-layer networks. However, the more layers, the higher the computation complexity, and the slower the model convergence. One way to accelerate the weight optimization, is to use a more computational efficient activation function. The most popular one in recent years is the rectified linear unit (ReLU):

Model design





```
> mx.set.seed(42)
> model_dnn <- mx.model.FeedForward.create(softmax, X=data_train.x,
y=data_train.y, ctx=devices, num.round=30, array.batch.size=100,
learning.rate=0.01, momentum=0.9, eval.metric=mx.metric.accuracy,
initializer=mx.init.uniform(0.1),
epoch.end.callback=mx.callback.log.train.metric(100))
Start training with 1 devices
[1] Train-accuracy=0.724793650793651
[2] Train-accuracy=0.904715189873417
[3] Train-accuracy=0.925537974683544
[4] Train-accuracy=0.939936708860759
[5] Train-accuracy=0.950379746835443
[6] Train-accuracy=0.95873417721519
[7] Train-accuracy=0.96509493670886
[8] Train-accuracy=0.969905063291139
[9] Train-accuracy=0.974303797468355
[10] Train-accuracy=0.977784810126584
[11] Train-accuracy=0.980696202531648
[12] Train-accuracy=0.983164556962027
[13] Train-accuracy=0.985284810126584
[14] Train-accuracy=0.987405063291141
[15] Train-accuracy=0.988924050632913
[16] Train-accuracy=0.990727848101267
[17] Train-accuracy=0.992088607594938
[18] Train-accuracy=0.993227848101268
[19] Train-accuracy=0.994398734177217
[20] Train-accuracy=0.995284810126584
[21] Train-accuracy=0.995854430379748
[22] Train-accuracy=0.996835443037975
[23] Train-accuracy=0.997183544303798
[24] Train-accuracy=0.997848101265823
[25] Train-accuracy=0.998164556962026
[26] Train-accuracy=0.998575949367089
[27] Train-accuracy=0.998924050632912
[28] Train-accuracy=0.999177215189874
[29] Train-accuracy=0.999367088607595
[30] Train-accuracy=0.999525316455696
```

Results

Results

	prediction_dnn									
	0	1	2	3	4	5	6	7	8	9
0	1041	0	2	0	0	1	3	0	8	1
1	0	1157	3	1	1	0	1	3	1	0
2	2	1	993	3	3	1	2	13	5	2
3	1	3	14	1033	1	13	0	5	14	6
4	0	2	1	0	991	0	4	4	1	12
5	4	2	3	12	3	892	4	3	6	8
6	10	0	1	0	3	4	988	0	4	0
7	0	5	9	1	2	0	0	1116	2	1
8	4	8	3	5	0	8	3	2	1020	12
9	1	1	0	4	13	3	0	16	2	957

Confusion Matrix

Our model achieved 99.95% accuraccy

Conclusion

- Hopefully with these results we may extend our project to different languages and an extension could be created with which we could scan text in a different language and have it be translated to English in digital form.