1. **Experiment with fully-connected feed-forward neural networks.**
2. **Experiment with convolutional networks (CNNs).**

Task 1(a) Sum-of-squares error *vs.* cross-entropy error function

- ***Sum-of-squares error:***

Step1: do experiment with different values of parameters:

1) number of hidden layers,
2) number of hidden units in each layer,
3) learning rates,
4) momentum rates,
5) Epochs
6) Patience in Early Stopping Monitor

Step2:

Table I: Results of experiments with Sum-of-squares error function

Hidden layer / units	Output_dim	Epoch	learning rates	momentum rates	Patience	Loss & Accuracy (train data & test data)
1 / 10	10	327	0.01	0.75	5	loss_test: 0.0122 acc_test: 0.9204 loss_train: 0.0047 acc_train: 0.9761
3 / 50	10	240	0.05	0.95	5	loss_test: 0.0196 acc_test: 0.8519 loss_train: 0.0137 acc_train: 0.8934
3 / 500	10	67	0.01	0.75	3	loss_test: 0.0062 acc_test: 0.9621 loss_train: 0.0018 acc_train: 0.9944
2 / 300	10	97	0.05	0.95	5	loss_test: 0.0068 acc_test: 0.9549 loss_train: 0.0015 acc_train: 0.9957
3/800	10	89	0.05	0.75	5	loss_test: 0.0057 acc_test: 0.9666 loss_train: 0.0012 acc_train: 0.9977

Step3: Table II: Best parameters config for Sum-of-squares error function

Config	Result
Num of hidden layers =3 Num of hidden layer units =800 Learning rates = 0.05, Momentum rates= 0.75, Activation function="ReLU" Epoch=89 Patience = 5	Execution time: 179.1158 Secs Accuracy (train data): 0.9977 Loss (train data): 0.0012 Accuracy (test data): 0.9666 Loss (test data): 0.0057

Step4: Confusion matrix for training set with the config of Table II:

293	0	0	0	0	0	0	0	0	0
0	311	0	0	0	0	0	1	0	1
0	0	306	0	0	0	0	0	0	0
0	0	0	305	0	1	0	0	0	1
0	0	0	0	311	0	0	0	0	0
0	0	0	0	0	311	0	0	0	0
0	2	0	0	0	0	304	0	0	0
0	0	0	0	0	0	0	314	0	0
0	1	0	0	0	0	0	0	291	0
0	0	0	0	0	0	0	0	0	305

Step5: Class accuracies:

Class 0 : 1.0
Class 1 : 0.9936102236421726
Class 2 : 1.0
Class 3 : 0.993485342019544
Class 4 : 1.0
Class 5 : 1.0
Class 6 : 0.9934640522875817
Class 7 : 1.0
Class 8 : 0.9965753424657534
Class 9 : 1.0

Step6: Confusion matrix for test set with the config of Table II:

177	0	0	0	1	0	0	0	0	0
0	182	0	0	0	0	0	0	0	0
0	1	176	0	0	0	0	0	0	0
1	0	3	172	0	3	0	1	1	2
0	1	0	0	176	0	0	1	3	0

0	0	0	0	0	179	1	0	0	2
0	1	0	0	3	0	176	0	1	0
0	0	0	0	0	5	0	166	0	8
0	8	0	0	1	2	0	0	157	6
0	0	0	0	1	2	0	0	1	176

Step7: Class accuracies:

Class 0 : 0.9943820224719101
Class 1 : 1.0
Class 2 : 0.9943502824858758
Class 3 : 0.9398907103825137
Class 4 : 0.9723756906077348
Class 5 : 0.9835164835164835
Class 6 : 0.9723756906077348
Class 7 : 0.9273743016759777
Class 8 : 0.9022988505747126
Class 9 : 0.9777777777777777

- ***Cross-entropy error function***

Step1: do experiment with different values of parameters:

1) number of hidden layers,
2) number of hidden units in each layer,
3) learning rates,
4) momentum rates,
5) Epochs,
6) Patience in Early Stopping Monitor

Step2:

Table III: Results of experiments with Cross-Entropy error function

Hidden layer / units	**Output_dim**	**Epoch**	**learning rates**	**momentum rates**	**Patience**	**Loss & Accuracy (train data & test data)**
1 / 10	10	25	0.0015	0.95	0	loss_test: 0.3287 acc_test: 0.9031 loss_train: 0.1759 acc_train: 0.9464
3 / 50	10	33	0.01	0.90	5	loss_test: 0.1627 acc_test: 0.9504 loss_train: 0.0302 acc_train: 0.9931
3 / 100	10	20	0.05	0.90	1	loss_test: 0.2590 acc_test: 0.9182 loss_train: 0.0711 acc_train: 0.9841
2 / 200	10	23	0.05	0.75		loss_test: 0.1027

					3	acc_test: 0.9666 loss_train: 0.0067 acc_train: 1
3/800	10	27	0.009	0.95	5	loss_test: 0.09423 acc_test: 0.9705 loss_train: 0.0045 acc_train: 1

Step3: Table IV: Best parameters config for Cross-Entropy error function

Config	Result
Num of hidden layers =3 Num of hidden layer units =800 Learning rates = 0.009, Momentum rates= 0.95, Activation function="ReLU" Epoch=27 Patience = 5	Execution time: 56.1165 Secs Accuracy (train data): 1 Loss (train data): 0.0045 Accuracy (test data): 0.9705 Loss (test data): 0.09423

Step4: Confusion matrix for training set with the config of Table IV:

293	0	0	0	0	0	0	0	0	0
0	313	0	0	0	0	0	0	0	0
0	0	306	0	0	0	0	0	0	0
0	0	0	307	0	0	0	0	0	0
0	0	0	0	311	0	0	0	0	0
0	0	0	0	0	311	0	0	0	0
0	0	0	0	0	0	306	0	0	0
0	0	0	0	0	0	0	314	0	0
0	0	0	0	0	0	0	0	292	0
0	0	0	0	0	0	0	0	0	305

Step5: Class accuracies:

Class 0 : 1.0
Class 1 : 1.0
Class 2 : 1.0
Class 3 : 1.0
Class 4 : 1.0
Class 5 : 1.0
Class 6 : 1.0
Class 7 : 1.0
Class 8 : 1.0
Class 9 : 1.0

Step6: Confusion matrix for test set with the config of Table IV:

178	0	0	0	0	0	0	0	0	0
0	181	0	0	0	0	0	0	1	0
0	2	174	0	0	0	1	0	0	0
0	0	4	174	0	2	0	0	1	2
0	1	0	0	176	0	0	1	3	0
0	0	0	0	0	180	0	0	0	2
0	0	0	0	1	0	179	0	1	0
0	0	0	0	1	4	0	166	1	7
0	7	0	0	0	1	0	0	162	4
0	0	0	2	0	3	0	0	1	174

Step7: Class accuracies:

Class 0 : 1.0
Class 1 : 0.9945054945054945
Class 2 : 0.9830508474576272
Class 3 : 0.9508196721311475
Class 4 : 0.9723756906077348
Class 5 : 0.989010989010989
Class 6 : 0.988950276243094
Class 7 : 0.9273743016759777
Class 8 : 0.9310344827586207
Class 9 : 0.9666666666666667

- We can conclude that Cross-entropy error function is better than sum-of-squares error function as the accuracy with cross-entropy function is higher.
- Also, the number of epochs/iterations required before early stopping is quite high for sum-of-squares error function.

Task 1(b) tanh vs. ReLU hidden units

- ***Experiments using ReLU hidden units***

Step1:

Table V: Results of experiments with ReLU hidden units

Hidden layer / units	Output_dim	Epoch	learning rates	momentum rates	Patience	Loss & Accuracy (train data & test data)
1 / 10	10	25	0.0015	0.95	0	loss_test: 0.3287 acc_test: 0.9031 loss_train: 0.1759 acc_train: 0.9464
3 / 50	10	33	0.01	0.90	5	loss_test: 0.1627 acc_test: 0.9504 loss_train: 0.0302 acc_train: 0.9931

3 / 100	10	20	0.05	0.90	1	loss_test: 0.2590 acc_test: 0.9182 loss_train: 0.0711 acc_train: 0.9841
2 / 200	10	23	0.05	0.75	3	loss_test: 0.1027 acc_test: 0.9666 loss_train: 0.0067 acc_train: 1
3/800	10	27	0.009	0.95	5	loss_test: 0.09423 acc_test: 0.9705 loss_train: 0.0045 acc_train: 1

Step2: Table VI: Best parameters config with ReLU hidden units

Config	Result
Num of hidden layers =3 Num of hidden layer units =800 Learning rates = 0.009, Momentum rates= 0.95, Activation function="ReLU" Epoch=27 Patience = 5	Execution time: 56.1165 Secs Accuracy (train data): 1 Loss (train data): 0.0045 Accuracy (test data): 0.9705 Loss (test data): 0.09423

Step3: Confusion matrix for training set with the config of Table VI:

293	0	0	0	0	0	0	0	0	0
0	313	0	0	0	0	0	0	0	0
0	0	306	0	0	0	0	0	0	0
0	0	0	307	0	0	0	0	0	0
0	0	0	0	311	0	0	0	0	0
0	0	0	0	0	311	0	0	0	0
0	0	0	0	0	0	306	0	0	0
0	0	0	0	0	0	0	314	0	0
0	0	0	0	0	0	0	0	292	0
0	0	0	0	0	0	0	0	0	305

Step4: Class accuracies:

Class 0 : 1.0
Class 1 : 1.0
Class 2 : 1.0
Class 3 : 1.0
Class 4 : 1.0
Class 5 : 1.0
Class 6 : 1.0
Class 7 : 1.0
Class 8 : 1.0
Class 9 : 1.0

Step5: Confusion matrix for test set with the config of Table VI:

178	0	0	0	0	0	0	0	0	0
0	181	0	0	0	0	0	0	1	0
0	2	174	0	0	0	1	0	0	0
0	0	4	174	0	2	0	0	1	2
0	1	0	0	176	0	0	1	3	0
0	0	0	0	0	180	0	0	0	2
0	0	0	0	1	0	179	0	1	0
0	0	0	0	1	4	0	166	1	7
0	7	0	0	0	1	0	0	162	4
0	0	0	2	0	3	0	0	1	174

Step6: Class accuracies:

Class 0 : 1.0
Class 1 : 0.9945054945054945
Class 2 : 0.9830508474576272
Class 3 : 0.9508196721311475
Class 4 : 0.9723756906077348
Class 5 : 0.989010989010989
Class 6 : 0.988950276243094
Class 7 : 0.9273743016759777
Class 8 : 0.9310344827586207
Class 9 : 0.9666666666666667

- ***Experiments using tanh hidden units***

Step1:
Table VII: Results of experiments with tanh hidden units

Hidden layer / units	**Output_dim**	**Epoch**	**learning rates**	**momentum rates**	**Patience**	**Loss & Accuracy (test data & train data)**
1 / 50	10	97	0.06	0.8	5	loss_test: 0.1732 acc_test: 0.9421 loss_train: 0.0328 acc_train: 0.9961
3 / 80	10	51	0.01	0.75	5	loss_test: 0.1639 acc_test: 0.9510 loss_train: 0.0168 acc_train: 0.9996
2 / 100	10	21	0.05	0.1	1	loss_test: 0.1618 acc_test: 0.9499 loss_train: 0.0588 acc_train: 0.9921

2 / 170	10	38	0.005	0.95	3	loss_test: 0.1027 acc_test: 0.9666 loss_train: 0.0067 acc_train: 1
3/600	10	69	0.05	0.95	5	loss_test: 0.0961 acc_test: 0.9677 loss_train: 0.0028 acc_train: 1

Step2: Table VIII: Best parameters config with tanh hidden units

Config	Result
Num of hidden layers =3 Num of hidden layer units =600 Learning rates = 0.05, Momentum rates= 0.95, Activation function="tanh" Epoch=69 Patience = 5	Execution time: 73.5298 Secs Accuracy (train data): 1 Loss (train data): 0.0028 Accuracy (test data): 0.9677 Loss (test data): 0.0961

Step3: Confusion matrix for training set with the config of Table VIII:

293	0	0	0	0	0	0	0	0	0
0	313	0	0	0	0	0	0	0	0
0	0	306	0	0	0	0	0	0	0
0	0	0	307	0	0	0	0	0	0
0	0	0	0	311	0	0	0	0	0
0	0	0	0	0	311	0	0	0	0
0	0	0	0	0	0	306	0	0	0
0	0	0	0	0	0	0	314	0	0
0	0	0	0	0	0	0	0	292	0
0	0	0	0	0	0	0	0	0	305

Step4: Class accuracies:

Class 0 : 1.0
Class 1 : 1.0
Class 2 : 1.0
Class 3 : 1.0
Class 4 : 1.0
Class 5 : 1.0
Class 6 : 1.0
Class 7 : 1.0
Class 8 : 1.0
Class 9 : 1.0

Step5: Confusion matrix for <u>test set</u> with the config of Table VIII:

177	0	0	0	0	1	0	0	0	0
0	181	0	0	0	0	0	0	1	0
0	3	173	1	0	0	0	0	0	0
1	0	2	172	0	2	0	2	2	2
0	0	0	0	177	0	0	1	3	0
0	0	0	0	0	180	0	0	0	2
0	0	0	0	2	0	178	0	1	0
0	0	0	0	0	5	0	165	2	7
0	7	0	0	1	2	0	0	160	4
0	0	0	2	1	1	0	0	0	176

Step6: Class accuracies:

Class 0 : 0.9943820224719101
Class 1 : 0.9945054945054945
Class 2 : 0.9774011299435028
Class 3 : 0.9398907103825137
Class 4 : 0.9779005524861878
Class 5 : 0.989010989010989
Class 6 : 0.9834254143646409
Class 7 : 0.9217877094972067
Class 8 : 0.9195402298850575
Class 9 : 0.9777777777777777

- We can conclude that using ReLU as the activation function is better than using tanh function.
- Also, execution time reduces when using ReLU as activation functions as compared to tanh activation functions.

2. Experiment with convolutional networks (CNNs)

Step1: do experiment with different values of parameters:

1) Filter size
2) Dropout
3) Kernel size
4) number of hidden layers,
5) number of hidden units in each layer,
6) learning rates,
7) momentum rates,
8) Epochs
9) Patience in Early Stopping Monitor = 5

Step2:

Table IX: Results of experiments with cross-entropy error function

Hidden layer / units	Filter size	Epoch	learning rates	momentum rates	dropout	Kernel size	Loss & Accuracy (train data & test data)
1 / 1000	64	37	0.05	0.75	0.3	(5,5)	loss_test: 0.0633 acc_test: 0.9766 loss_train: 0.0085 acc_train: 0.9987
2 / 500	64	32	0.01	0.95	0.3	(5,5)	loss_test: 0.0796 acc_test: 0.9733 loss_train: 0.0113 acc_train: 0.9980
2 / (50,30)	32	29	0.01	0.95	0.2	(3,3)	loss_test: 0.0744 acc_test: 0.9732 loss_train: 0.0292 acc_train: 0.9908
2 / 300	32	26	0.05	0.95	0.2	(2,2)	loss_test: 0.1132 acc_test: 0.9605 loss_train: 0.0531 acc_train: 0.9833
2/(500,300)	64	34	0.05	0.75	0.3	(2,2)	loss_test: 0.0057 acc_test: 0.9666 loss_train: 0.0012 acc_train: 0.9977

Step3: Table X: Best parameters config for cross-entropy error function

Config	Result
Num of hidden layers = 2 Num of hidden layer units = 500,300 Learning rates = 0.05, Momentum rates= 0.75, Activation function="ReLU" Filter Size = 64 Kernel size = (2,2) Dropout = 0.3 Epoch=34 Patience = 5	Execution time: 48.29061 Secs Accuracy (train data): 0.9931 Loss (train data): 0.0194 Accuracy (test data): 0.9788 Loss (test data): 0.0736

Step4: Confusion matrix for <u>training set</u> with the config of Table X:

293	0	0	0	0	0	0	0	0	0
0	309	0	0	0	0	0	1	1	2
0	0	306	0	0	0	0	0	0	0
0	0	0	305	0	2	0	0	0	0
0	0	0	0	307	0	2	0	0	2
0	0	0	0	0	311	0	0	0	0
0	2	0	0	0	0	304	0	0	0
0	0	1	0	0	0	0	313	0	0
0	0	0	0	0	1	0	0	291	0
0	1	0	2	1	3	0	0	0	298

Step5: Class accuracies:

Class 0 : 1.0
Class 1 : 0.987220447284345
Class 2 : 1.0
Class 3 : 0.993485342019544
Class 4 : 0.9871382636655949
Class 5 : 1.0
Class 6 : 0.9934640522875817
Class 7 : 0.9968152866242038
Class 8 : 0.9965753424657534
Class 9 : 0.9770491803278688

Step6: Confusion matrix for <u>test set</u> with the config of Table X:

178	0	0	0	0	0	0	0	0	0
0	181	0	1	0	0	0	0	0	0
0	0	174	0	1	0	0	1	1	0
0	0	0	180	0	1	0	0	1	1
0	0	0	0	179	0	1	0	0	1
0	0	0	0	0	181	0	0	0	1
1	0	0	0	1	1	177	0	1	0
0	0	1	1	0	3	0	170	2	2
0	4	0	0	0	2	0	0	163	5
0	0	0	2	0	1	0	0	1	176

Step7: Class accuracies:

Class 0 : 1.0
Class 1 : 0.9945054945054945
Class 2 : 0.9830508474576272
Class 3 : 0.9836065573770492
Class 4 : 0.988950276243094
Class 5 : 0.9945054945054945
Class 6 : 0.9779005524861878
Class 7 : 0.9497206703910615
Class 8 : 0.9367816091954023
Class 9 : 0.9777777777777777

- Convolutional Networks has higher accuracy than the fully connected neural network
- CNNs have higher accuracy due to reduced overfitting problem as compared to fully connected neural networks.
- CNNS build the model within less number of epochs as compared to fully connected neural networks