COMS 573 LAB 2 HOMAGNI SAHA

DESCRIPTION OF DESIGN CHOICES MADE:

Fully connected vs. Convolutional neural networks:

We tried different cases when only fully connected neural networks were used and when a convolutional neural network was used. Generally, after performing a 2D convolution, the features need to be converted to 1 dimension and connected to the final layer.

Relu vs. Tanh:

Both are activation functions applied to the neuron. They have different characteristics and work differently on different datasets. For the hidden layers we tried two different activation functions relu and tanh.

Sum of squares vs. cross entropy:

They are two different loss functions used to calculate the final loss with the labels and backpropagate the loss throughout the network.

Hidden layers:

The number of fully connected layers may be varied in a neural network hence producing deep and shallow architectures.

Hidden units:

The number of neurons in each layer may be varied to produce either “broad” or “narrow” architectures.

Learning rate:

This is a parameter that determines how fast the weights are updated in the neural networks.

Momentum:

This is a parameter that acts like “inertia” and helps the neural network to overcome “ravine surface” problems. It also helps to speed up the rate of convergence. The optimizer acts like a ball that rolls over the gradient surface. If it faces sloping hill for a long time then it rolls faster, i.e., the learning rate increases.

Input scaling:

The inputs could be normalized to lie between a set range of values.

Epochs:

This is the number of times the training algorithm makes passes over the training data

Batch-size:

In batch gradient descent, these are the number of samples chosen in a batch

Last activation layer:

This is the activation applied to the last layer of the neural network.

Optimizer used:

Two popular optimizers: stochastic gradient descent (sgd) and root mean square propagation(rmsprop) were compared.

Square filter size:

This is the kernel size for the 2d convolution operation

Max-pool size:

This is the pooling size for the 2d convolution operation

Dropout:

This is used throughout the architecture to ensure better generalization abilities and prevent overfitting.

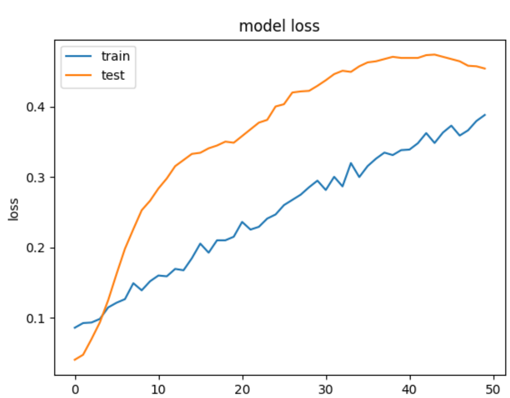
The experimental results are summarized in the tables and figures that follow. The four major possible cases that were asked for are divided into 4 tables and the comments and discussion follow at the end of each table.

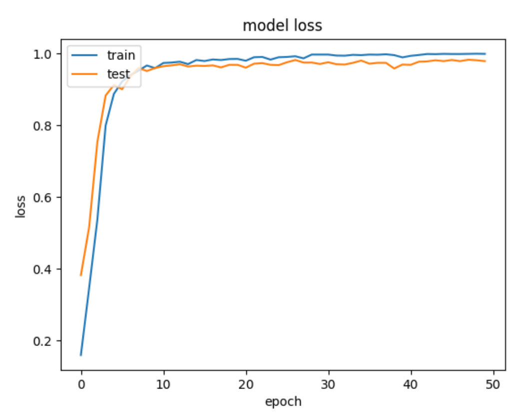
FCN CASE II	RELU CASE A	First layer after CNN has 8*8 units	GENERAL INFORMATION: These results were obtained using Keras with Theano backend. The 1st number in the results is the validation accuracy (training data was split into 30 percent validation). The 2nd number in the results denote the test accuracy. For the figures, the left-hand side denote case1 and right-hand side denote case2 of the same table. The figures are plotted to show accuracy or the reduction in the model loss across number of epochs. The discussions and comments are listed as the cases are encountered. The parameters were chosen in such a way to form a diverse distribution of accuracies.										
Case 1	SUM OF SQUARES												
		Hidden layers	Hidden Units	Learning Rate	Momentum rate/rho	Input scaling	Epochs	Batch-size	Last layer activation	Optimizer used	Filter Size	Max-pool size	Result
	Classification Accuracy	10 10 8 5 2 1	256 150 100 100 150 256	0.005 0.01 0.1 0.1 0.01 0.005	0.9 0.7 0.5 0.5 0.7 0.9	Yes No Yes No Yes No	50 30 20 20 30 50	100 50 10 10 50 100	Softmax Sigmoid Softmax Sigmoid Softmax sigmoid	Sgd Sgd Rmspr Rmspr Sgd rmspr			45.4,47.4 30.1,30.9 9.43,9.85 10.9,9.68 87.0,85.3 98.1,95.8
	Convergence Speed	Please see the following figures for a better understanding. With respect to time taken, increasing batch size reduces time taken by a huge margin, and improves generalization error.											
	Comments/discussion	1. Increasing hidden units helps a lot. A momentum of 0.5 is prohibitive. 2. Increasing epochs generally increases accuracy for deeper networks. 3. Increasing batch-size leads to smoother convergence and faster training. This is because more samples are being considered in the batch gradient and this results in a smoothing effect. 4. A moderate learning rate leads to better results, because too high (0.1) may cause the optimizer to swing back and forth the minima, making no progress, or even diverging sometimes. A very low learning rate needs a huge number of epochs to achieve considerable results. 5. Deeper architectures never perform better than shallower architectures for number of epochs within a certain range. For MNIST, broad and shallow network is preferred over deep and narrow networks. Fully connected networks prove to reach very close to same best results as convolutional neural networks for optimal parameters. 6. Increasing momentum leads to faster and better convergence											
Case 2	CROSS-ENTROPY												
		Hidden Layers	Hidden Units	Learning Rate	Momentum Rate/rho	Input scaling	Epochs	Batch-size	Last layer activation	Optimizer used	Filter size	Max-poo size	Result

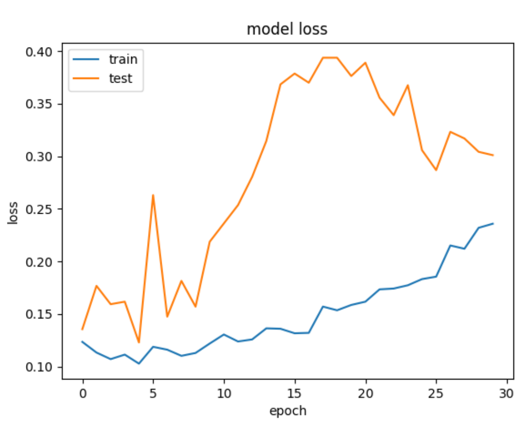
	Classification Accuracy	10	256	0.005	0.9	Yes	50	100	Softmax	Sgd			97.9,96.3
		10	150	0.01	0.7	No	30	50	Sigmoid	Sgd			96.6,94.2
		8	100	0.1	0.5	Yes	20	10	Softmax	Rmspr			9.43,9.85
		5	100	0.1	0.5	No	20	10	Sigmoid	Rmspr			10.2,9.91
		2	150	0.01	0.7	Yes	30	50	Softmax	Sgd			98.7,95.8
		1	256	0.005	0.9	No	50	100	sigmoid	rmspr			96.2,95.1
	Convergence Speed	Please see the following figures for a better understanding. With respect to time taken, increasing batch size reduces time taken by a huge margin, and improves generalization error.											
	Comments/discussion	1. In addition to above, cross-entropy leads to slightly increased accuracy for same number of epochs and reduces overfitting. This is natural because we are looking to minimize the entropy difference between prediction and result as opposed to minimizing the squared sum.											

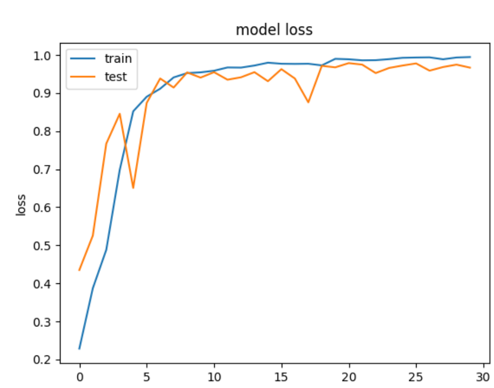
FIGURES (CASE II- CASE A):

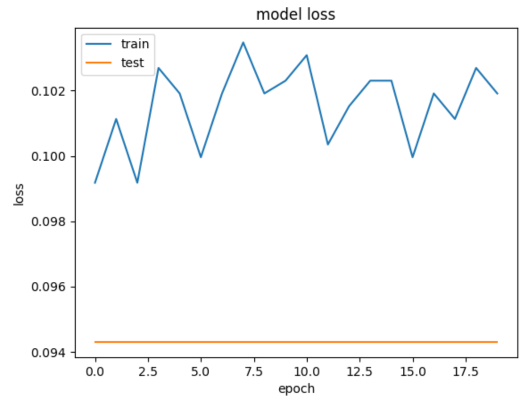
General note: By loss on the y-axis I mean reduction in loss or the increase in accuracy. So, please don't be confused. I mean it to be the model accuracy but Keras stores the training record in a different manner, so the plots have y-axis labelled as loss.

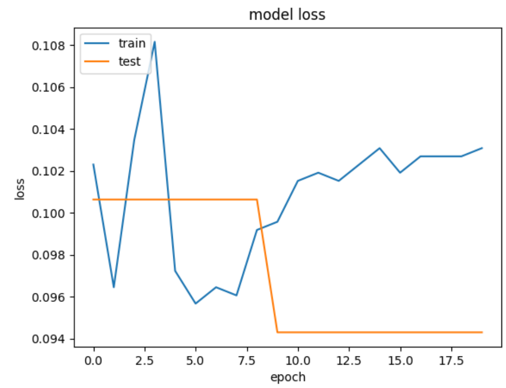


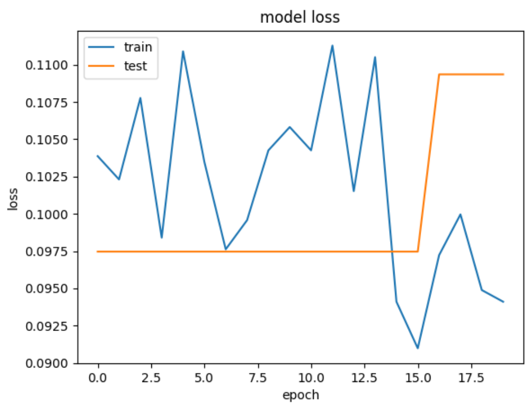


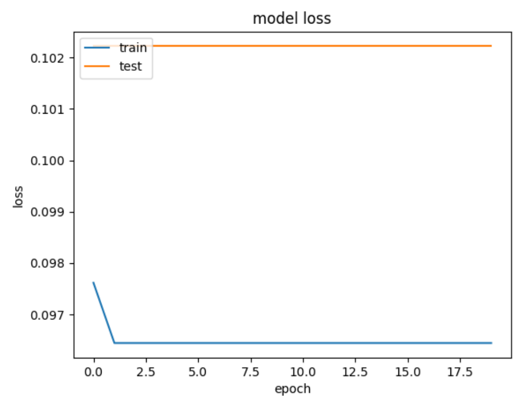


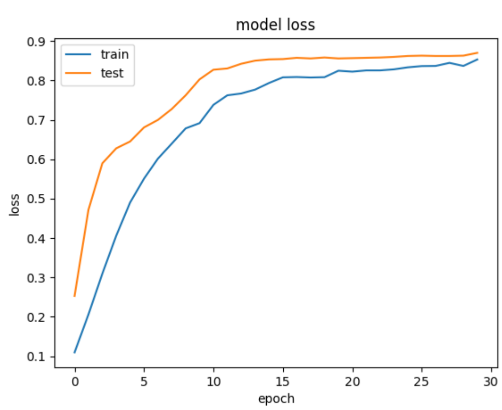


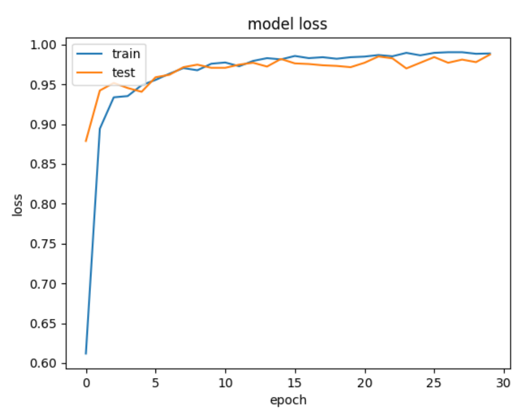

model loss
train
test
loss
epoch

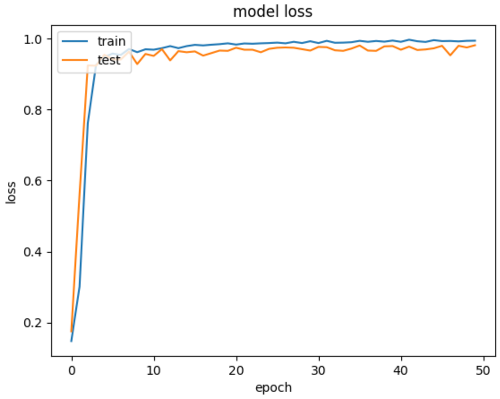

model loss
train
test
loss
epoch

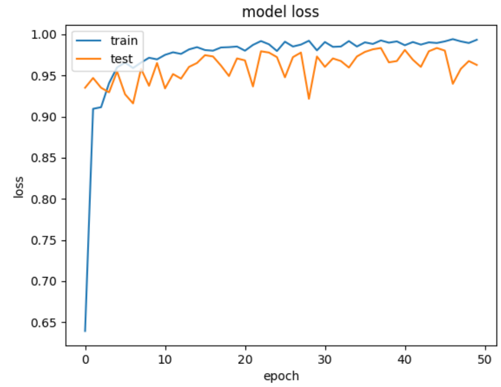

model loss
train
test
loss
epoch


model loss
train
test
loss
epoch


model loss
train
test
loss
epoch


model loss
train
test
loss
epoch

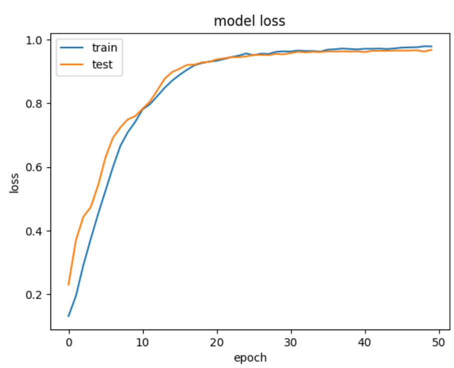


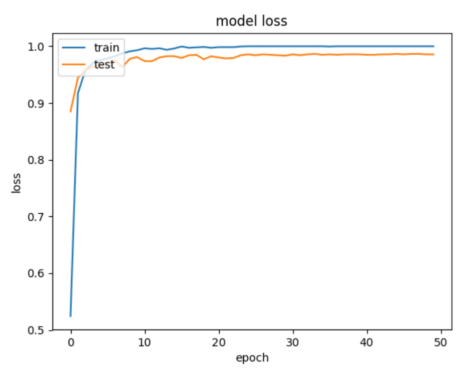


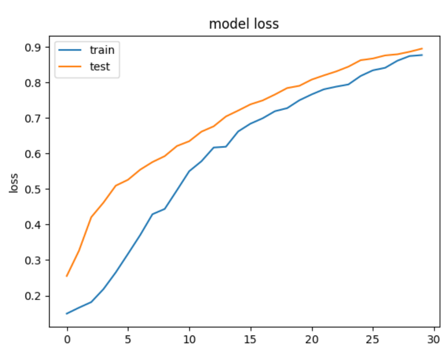
FCN CASE II	**TANH** CASE B	First layer after CNN has 8*8 units											
Case 1	SUM OF SQUARES												
		Hidden layers	Hidden Units	Learning Rate	Momentum rate/rho	Input scaling	Epochs	Batch-size	Last layer activation	Optimizer used	Filter Size	Max-pool size	Result
	Classification Accuracy	10 10 8 5 2 1	256 150 100 100 150 256	0.005 0.01 0.1 0.1 0.01 0.005	0.9 0.7 0.5 0.5 0.7 0.9	Yes No Yes No Yes No	50 30 20 20 30 50	100 50 10 10 50 100	Softmax Sigmoid Softmax Sigmoid Softmax sigmoid	Sgd Sgd Rmspr Rmspr Sgd rmspr			96.8,94.9 89.5,88.7 9.98,10.07 9.43,9.81 94.1,93.4 97.4,95.0
	Convergence Speed	Please see the following figures for a better understanding. With respect to time taken, increasing batch size reduces time taken by a huge margin, and improves generalization error.											
	Comments/discussions	1. In addition to above, tanh is found to match better with mean squared error than relu. This might be a specific case for the dataset, as there is no significant conclusive evidence (journals) on this issue.											
Case 2	CROSS-ENTROPY												
		Hidden Layers	Hidden Units	Learning Rate	Momentum Rate/rho	Input scaling	Epochs	Batch-size	Last layer activation	Optimizer used	Filter size	Max-poo size	Result

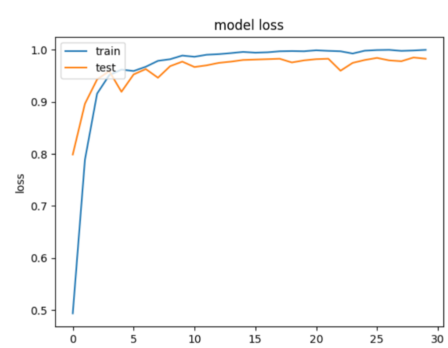
	Classification Accuracy	10 10 8 5 2 1	256 150 100 100 150 256	0.005 0.01 0.1 0.1 0.01 0.005	0.9 0.7 0.5 0.5 0.7 0.9	Yes No Yes No Yes No	50 30 20 20 30 50	100 50 10 10 50 100	Softmax Sigmoid Softmax Sigmoid Softmax sigmoid	Sgd Sgd Rmspr Rmspr Sgd rmspr			98.5,95.9 98.2,96.6 9.43,9.85 9.43,9.85 97.9,96.4 97.4,94.3
	Convergence Speed	Please see the following figures for a better understanding. With respect to time taken, increasing batch size reduces time taken by a huge margin, and improves generalization error.											
	Comments/discussions	Generally, accuracy increases with increasing epochs. This makes sense as there are more passes over the data.											

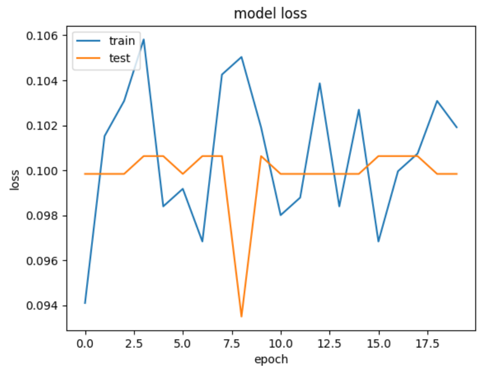
FIGURE (CASE II-CASE B):

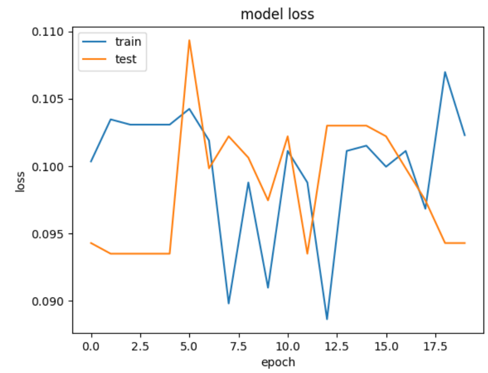


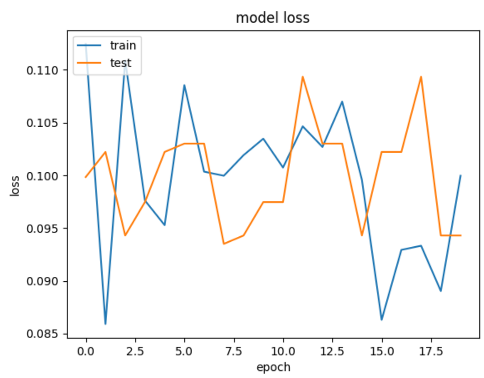


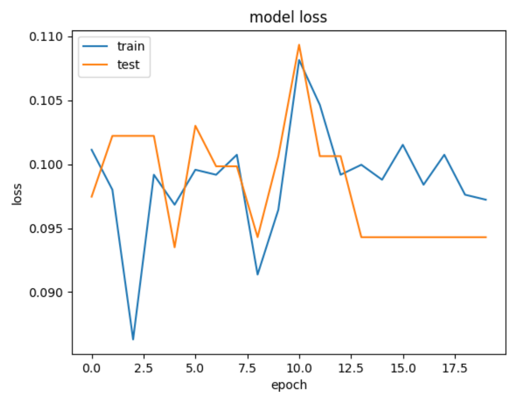


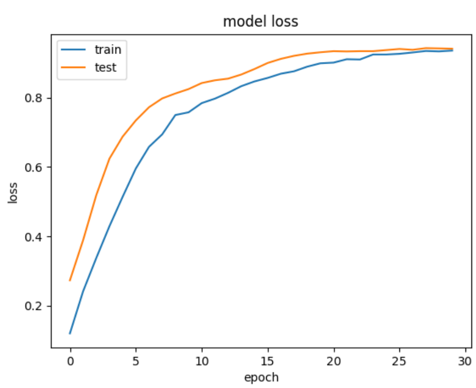


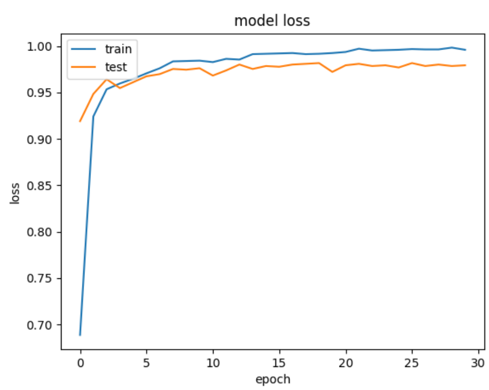

model loss
train
test
loss
epoch
0.106
0.104
0.102
0.100
0.098
0.096
0.094
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

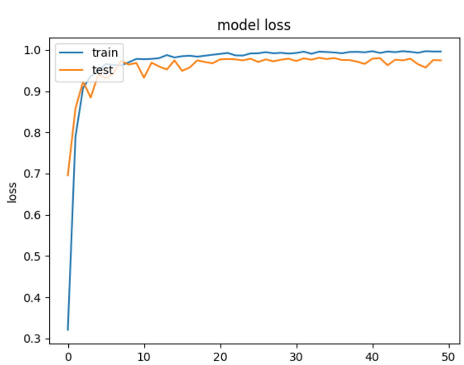

model loss
train
test
loss
epoch
0.110
0.105
0.100
0.095
0.090
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

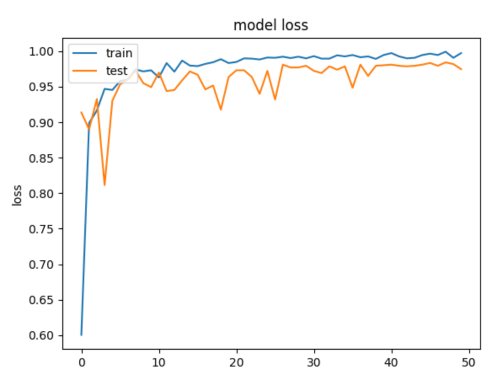

model loss
train
test
loss
epoch
0.110
0.105
0.100
0.095
0.090
0.085
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


model loss
train
test
loss
epoch
0.110
0.105
0.100
0.095
0.090
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


model loss
train
test
loss
epoch
0.8
0.6
0.4
0.2
0
5
10
15
20
25
30


model loss
train
test
loss
epoch
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0
5
10
15
20
25
30


model loss
train
test
loss
epoch
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
10
20
30
40
50

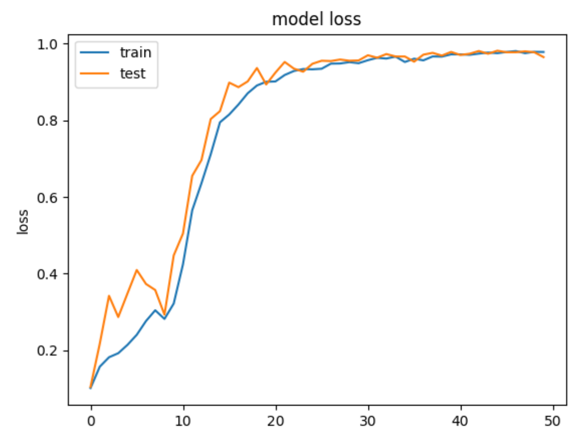

model loss
train
test
loss
epoch
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0
10
20
30
40
50

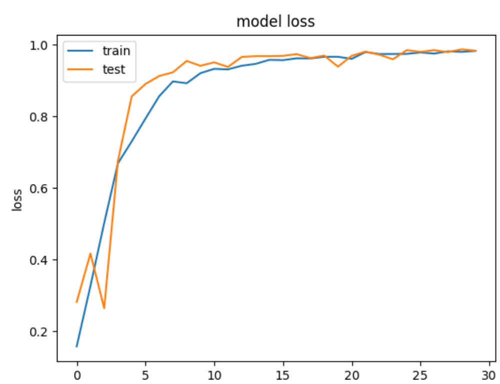
CNN CASE I	RELU CASE A	First layer after CNN has 512 units											
Case 1	SUM OF SQUARES												
		Hidden layers	Hidden Units	Learning Rate	Momentum rate/rho	Input scaling	Epochs	Batch-size	Last layer activation	Optimizer used	Filter Size	Max-pool size	Result
	Classification Accuracy	10 10 8 5 2 1	256 150 100 100 150 256	0.005 0.01 0.1 0.1 0.01 0.005	0.9 0.7 0.5 0.5 0.7 0.9	Yes No Yes No Yes No	50 30 20 20 30 50	100 50 10 10 50 100	Softmax Sigmoid Softmax Sigmoid Softmax sigmoid	Sgd Sgd Rmspr Rmspr Sgd rmspr	(3,3) (5,5) (7,7) (3,3) (5,5) (7,7)	(2,2) (3,3) (5,5) (5,5) (3,3) (2,2)	30.9,10.1 43.5,46.2 9.75,9.96 10.0,10.1 18.2,12,4 98.5,97.9
	Convergence Speed	Please see figures that follow											
	Comments/discussion	1. In addition to above tables, CNN achieves the best accuracy because of weight sharing, scale invariance and ability to detect local changes over regions of the image.											
Case 2	CROSS-ENTROPY	First layer after CNN has 512 units											
		Hidden Layers	Hidden Units	Learning Rate	Momentum Rate/rho	Input scaling	Epochs	Batch-size	Last layer activation	Optimizer used	Filter size	Max-poo size	Result
	Classification Accuracy	10 10 8 5 2 1	256 150 100 100 150 256	0.005 0.01 0.1 0.1 0.01 0.005	0.9 0.7 0.5 0.5 0.7 0.9	Yes No Yes No Yes No	50 30 20 20 30 50	100 50 10 10 50 100	Softmax Sigmoid Softmax Sigmoid Softmax sigmoid	Sgd Sgd Rmspr Rmspr Sgd rmspr	(3,3) (5,5) (7,7) (3,3) (5,5) (7,7)	(2,2) (3,3) (5,5) (5,5) (3,3) (2,2)	96.5,93.3 98.3,96.9 9.75,9.96 10.2,9.91 97.2,94.9 98.7,98.3

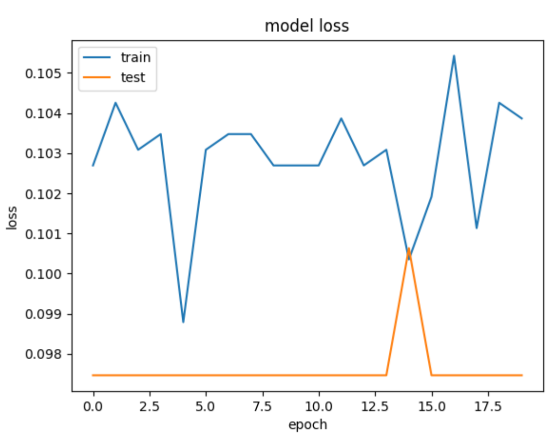
	Convergence Speed	Please see figures that follow.
	Comments	Generally, accuracy increases with increasing epochs.

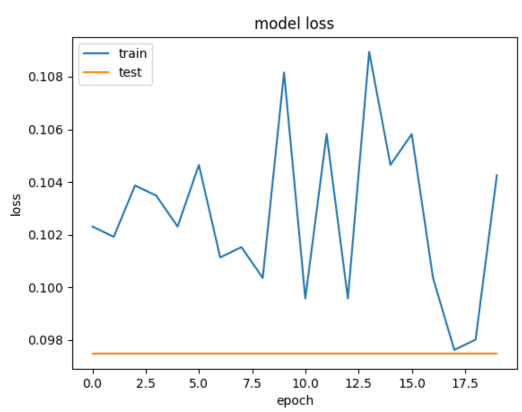
FIGURES (CASE I, CASE A):

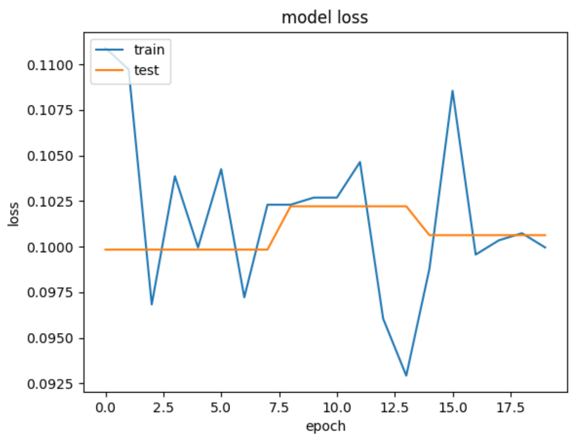
General note: By loss on the y-axis I mean reduction in loss or the increase in accuracy. So, please don't be confused. I mean it to be the model accuracy but Keras stores the training record in a different manner, so the plots have y-axis labelled as loss.

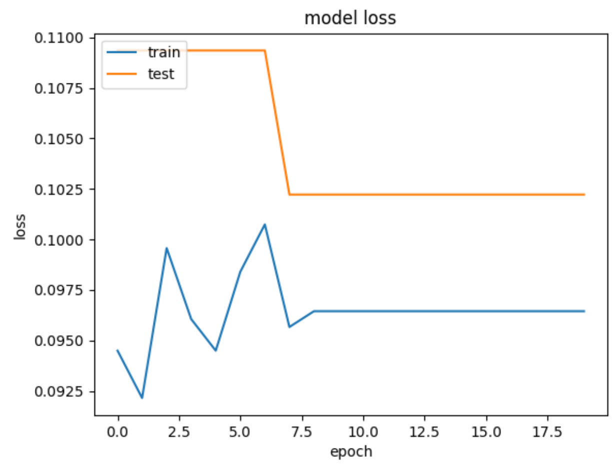


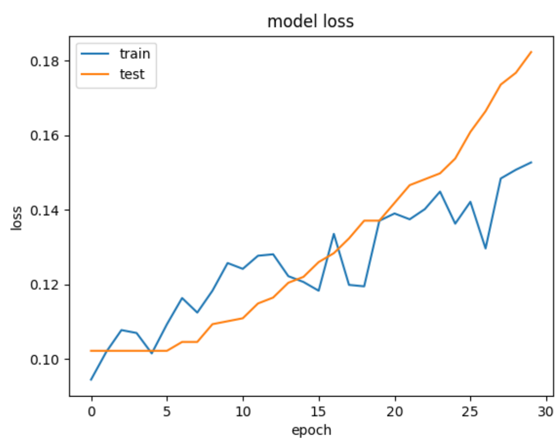


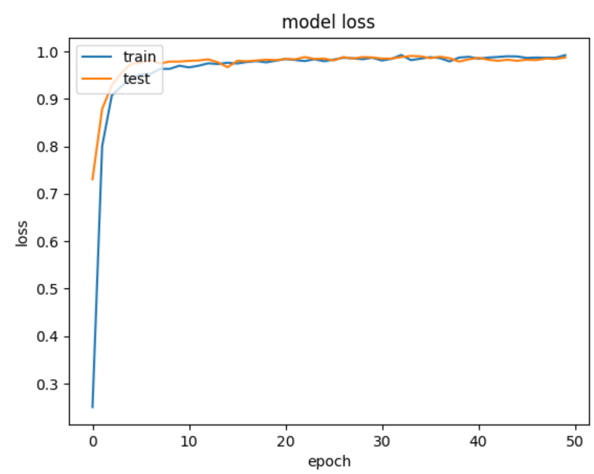


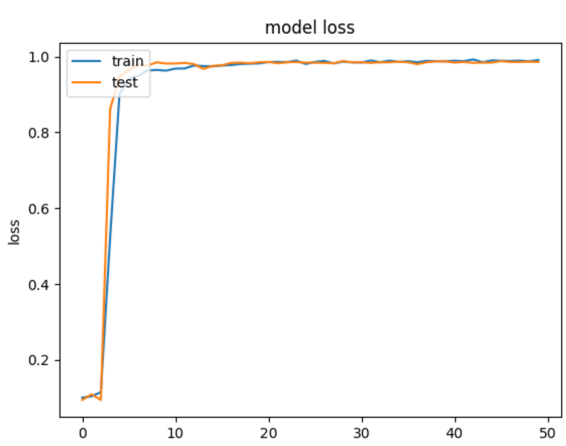


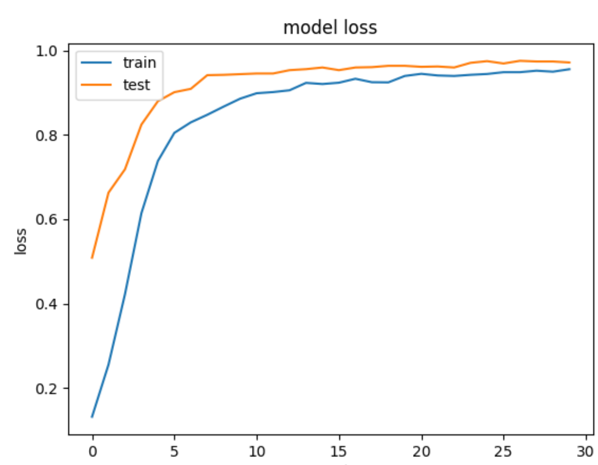

model loss
train
test
loss
epoch
0.0 2.5 5.0 7.5 10.0 12.5 15.0 17.5
0.0925 0.0950 0.0975 0.1000 0.1025 0.1050 0.1075 0.1100


model loss
train
test
loss
epoch
0.0 2.5 5.0 7.5 10.0 12.5 15.0 17.5
0.0925 0.0950 0.0975 0.1000 0.1025 0.1050 0.1075 0.1100


model loss
train
test
loss
epoch
0 5 10 15 20 25 30
0.10 0.12 0.14 0.16 0.18


model loss
train
test
loss
epoch
0 10 20 30 40 50
0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

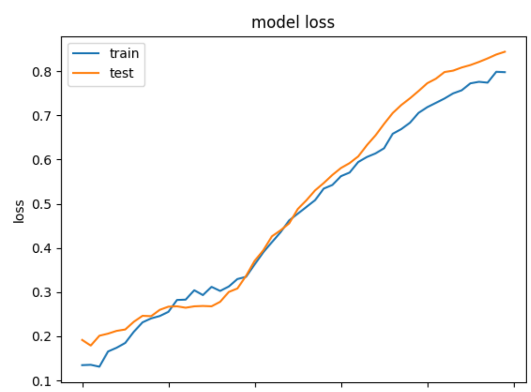

model loss
train
test
loss
epoch
0 10 20 30 40 50
0.2 0.4 0.6 0.8 1.0

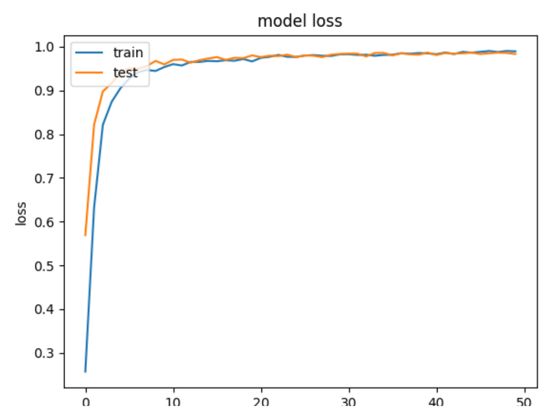

model loss
train
test
loss
epoch
0 5 10 15 20 25 30
0.2 0.4 0.6 0.8 1.0

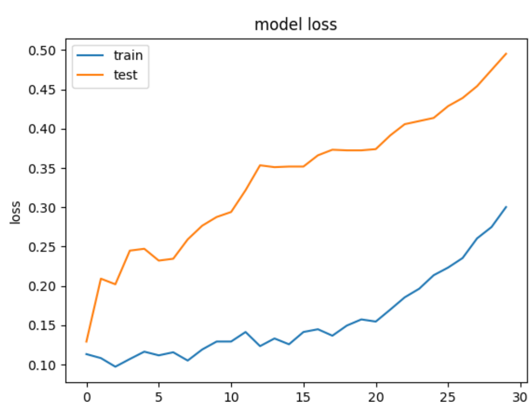
CNN CASE I	TANH CASE B	First layer after CNN has 512 units										
Case 1	SUM OF SQUARES											
		Hidden layers	Hidden Units	Learning Rate	Momentum rate/rho	Input scaling	Epochs	Batch-size	Last layer activation	Optimizer used	Filter Size	Max-pool size	Result
	Classification Accuracy	10 10 8 5 2 1	256 150 100 100 150 256	0.005 0.01 0.1 0.1 0.01 0.005	0.9 0.7 0.5 0.5 0.7 0.9	Yes No Yes No Yes No	50 30 20 20 30 50	100 50 10 10 50 100	Softmax Sigmoid Softmax Sigmoid Softmax sigmoid	Sgd Sgd Rmspr Rmspr Sgd rmspr	(3,3) (5,5) (7,7) (3,3) (5,5) (7,7)	(2,2) (3,3) (5,5) (5,5) (3,3) (2,2)	84.3,51.8 49.5,50.3 9.43,9.85 10.2,9.91 34.3,14.9 98.9,98.0
	Convergence Speed	Please see figures										
	Comments/discussions											
Case 2	CROSS-ENTROPY											
		Hidden Layers	Hidden Units	Learning Rate	Momentum Rate/rho	Input scaling	Epochs	Batch-size	Last layer activation	Optimizer used	Filter size	Max-poo size	Result
	Classification Accuracy	10 10 8 5 2 1	256 150 100 100 150 256	0.005 0.01 0.1 0.1 0.01 0.005	0.9 0.7 0.5 0.5 0.7 0.9	Yes No Yes No Yes No	50 30 20 20 30 50	100 50 10 10 50 100	Softmax Sigmoid Softmax Sigmoid Softmax sigmoid	Sgd Sgd Rmspr Rmspr Sgd rmspr	(3,3) (5,5) (7,7) (3,3) (5,5) (7,7)	(2,2) (3,3) (5,5) (5,5) (3,3) (2,2)	98.3,63.3 97.7,96.7 10.2,10.1 9.35,10.0 97.1,81.7 98.4,96.8

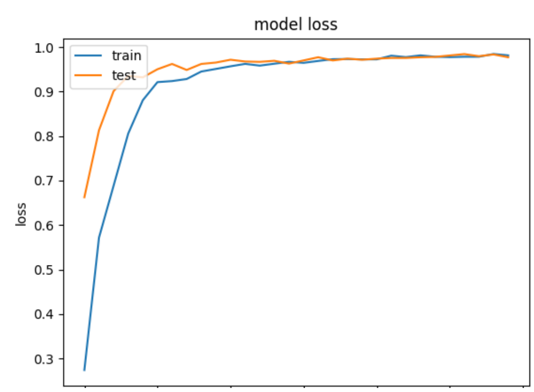
	Convergence Speed												
	Comments/discussions	Generally, accuracy increases with increasing epochs. Convolutional neural networks with cross-entropy as loss function performs the best.											

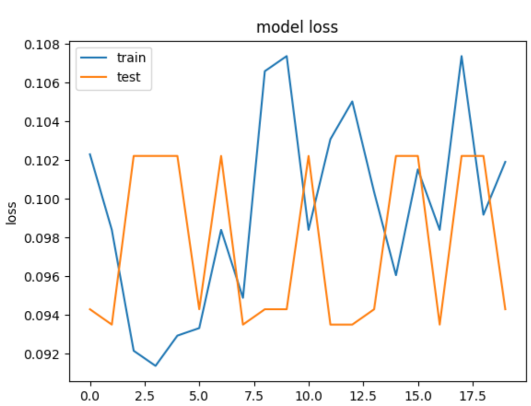
FIGURES (CASE I, CASE B)

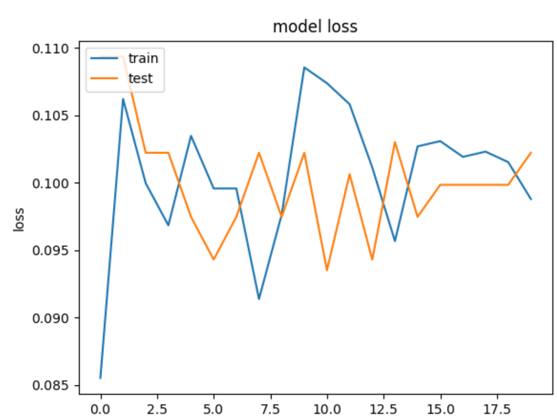


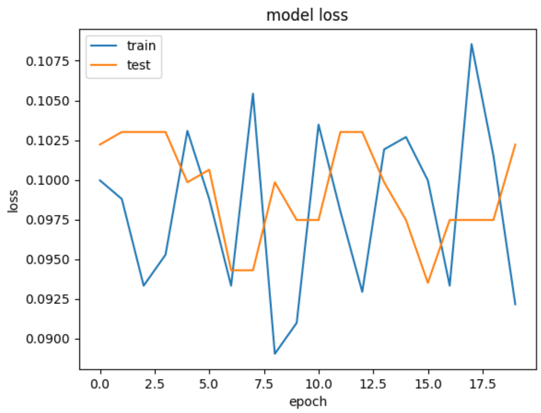


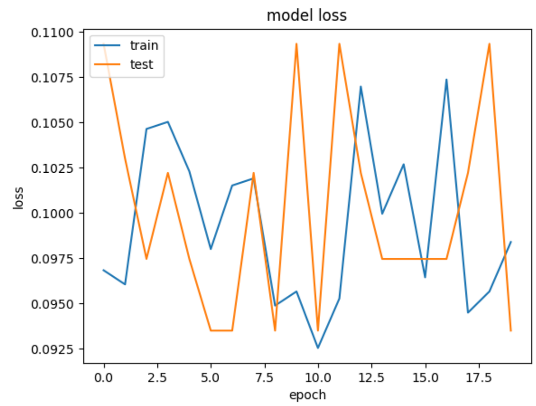


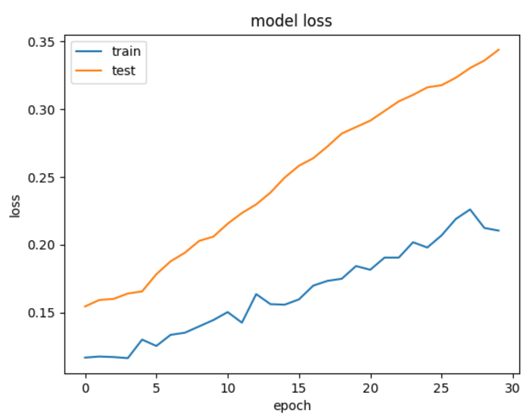


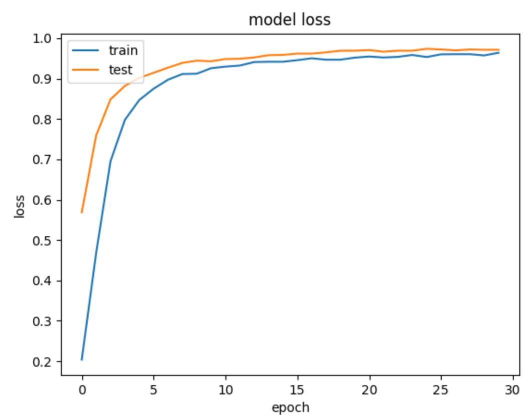


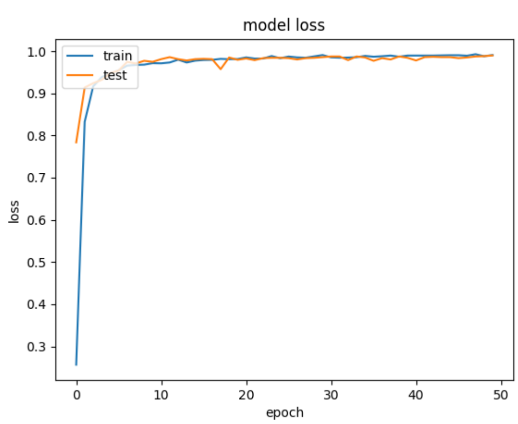


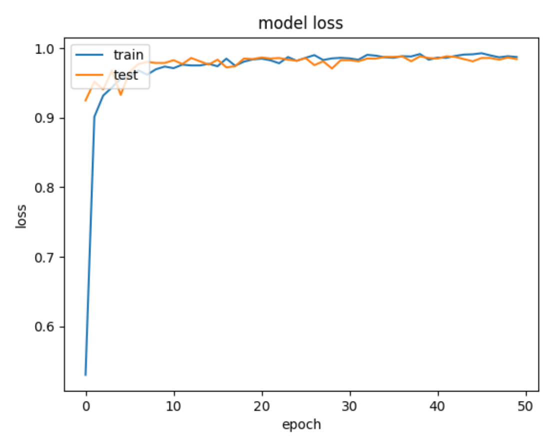

model loss
train
test
loss
epoch
0.1075
0.1050
0.1025
0.1000
0.0975
0.0950
0.0925
0.0900
0.0 2.5 5.0 7.5 10.0 12.5 15.0 17.5


model loss
train
test
loss
epoch
0.1100
0.1075
0.1050
0.1025
0.1000
0.0975
0.0950
0.0925
0.0 2.5 5.0 7.5 10.0 12.5 15.0 17.5


model loss
train
test
loss
epoch
0.35
0.30
0.25
0.20
0.15
0 5 10 15 20 25 30


model loss
train
test
loss
epoch
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0 5 10 15 20 25 30


model loss
train
test
loss
epoch
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0 10 20 30 40 50


model loss
train
test
loss
epoch
1.0
0.9
0.8
0.7
0.6
0 10 20 30 40 50