MNIST Digits Classification with MLP

Introduction

Ignoring setups used to select hyperparameters, I have looked at the required 2 activations * 2 layers * 2 loss functions = 8 different setups (see **Results Summary**). Additionally, I have also looked at Model5 which differs from Model4 only in the number of neurons in its layers. Altogether, there are 10 different setups.

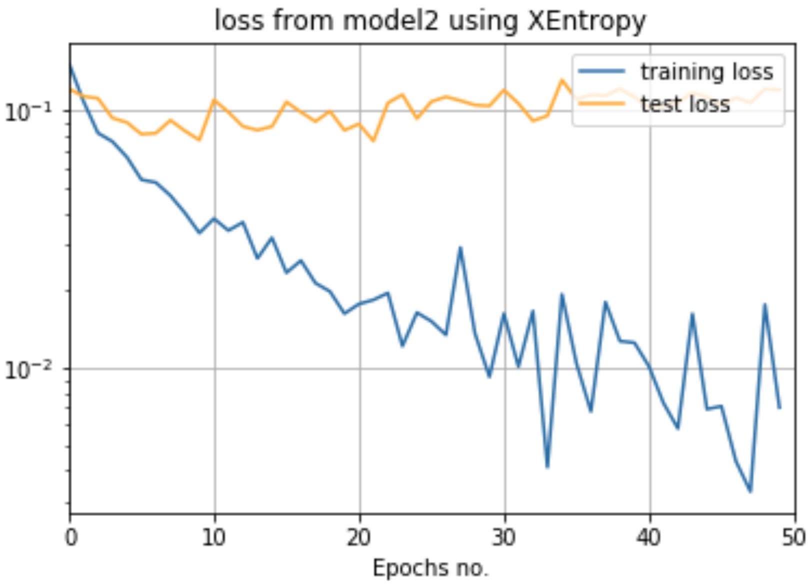
Hyperparameter Selection

In theory, to select the right hyperparameters, a full exhaustive search that iterates through every combination should be done with a **validation dataset**. However, for simplicity's sake, this is done mainly on an ad hoc basis, where the parameters work well enough in combination.

Hyperparameters of interests are reproduced below.

Hyperparameter	Value
Batch size	100
Weight decay	0
Momentum	0.9
Max epoch	50/100 depending on convergence
Learning rate	0.01/0.001 depending on convergence

Particularly, a learning rate of 0.01 generally works well on all models in 50 epochs. This is not the case for models with ReLU and XEntropy loss though. As can be seen from this plot of a bad example below, learning rate of 0.01 is too high, producing the zigzag shape of training loss.



Learning rate is thus reduced to 0.001 and max epoch increased to 100 to produce better convergence.

Results Summary

Here, convergence refers to the convergence of loss on test dataset. Convergence tends to be ill defined when we try to compare different loss functions – it therefore only indicates an approximate epoch no. here.

Model no.	Activation	Layers	Loss function	Learning rate	Convergence at epoch no.	Accuracy	Loss
Model1	Sigmoid	1	Euclidian	0.01	50	97.5%	10^{-2}
Model1	Sigmoid	1	XEntropy	0.01	30	98.0%	10^{-1}
Model2	ReLU	1	Euclidian	0.01	30	98.1%	10^{-2}
Model2	ReLU	1	XEntropy	0.001	70	96.9%	10^{-1}
Model3	Sigmoid	2	Euclidian	0.01	30	97.3%	10^{-2}
Model3	Sigmoid	2	XEntropy	0.01	30	98.1%	10^{-1}
Model4	ReLU	2	Euclidian	0.01	50	98.2%	10^{-2}
Model4	ReLU	2	XEntropy	0.001	30	96.6%	10^{-1}
Model5	ReLU	2	Euclidian	0.01	30	98.4%	10^{-2}
Model5	ReLU	2	XEntropy	0.001	30	97.2%	10^{-1}

Architecture of Model4 & Model5:
Model4: (784 x 512) -> (512 x 128) -> (128 x 10)
Model5: (784 x 392) -> (392 x 196) -> (196 x 10)

Comments and Discussion on Results

Sigmoid vs. ReLU:
Generally, ReLU is preferred over sigmoid.

Model4 & model5 of **2 hidden layers** of **ReLU** activation and **Euclidian** loss function perform the best out of the setups.

1 hidden layer vs. 2 hidden layers:
Prediction accuracy does not always improve when we add a layer. This depends on the activation function and loss function. Notably, model4 & model5 (2 layers) do have improvements over model2 (1 layer).

Generally, models with 2 layers tend to converge faster, as the plots show (see **Plots** *model1_Euclidean_accuracy* & *model3_Euclidean_accuracy*).

Euclidian vs. Cross-Entropy Loss:

Between the two, cross-entropy loss introduces more overfitting (see **Plots** *model3_Xentropy_loss*).

Architecture:

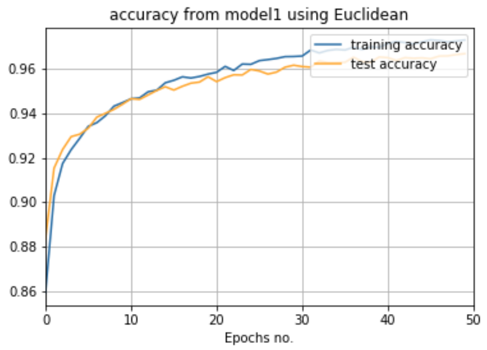
Not much difference between Model4 & Model5.

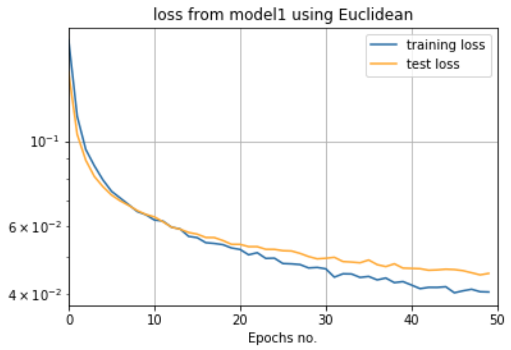
Overfitting:

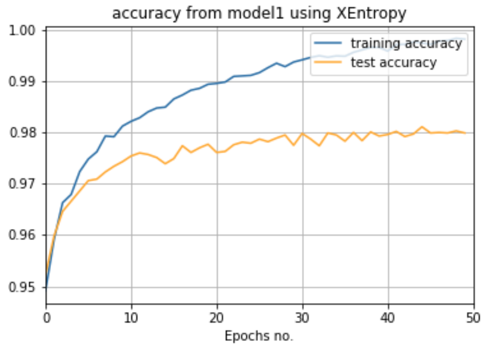
From the **Plots**, it is fairly obvious that overfitting plagues each model to various degrees. To improve accuracy, we could include regularization in our trainig process.

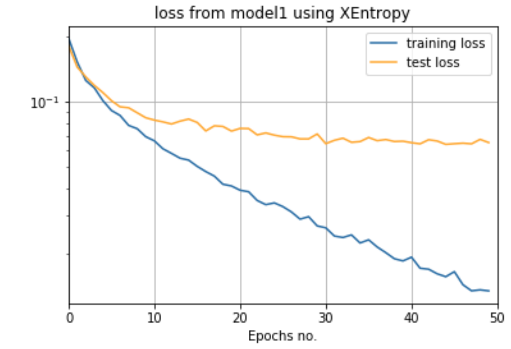
Plots

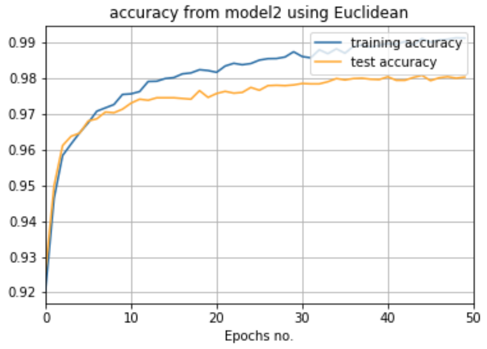
The loss (training & test) and accuracy (training & test) of each setup are plotted below.

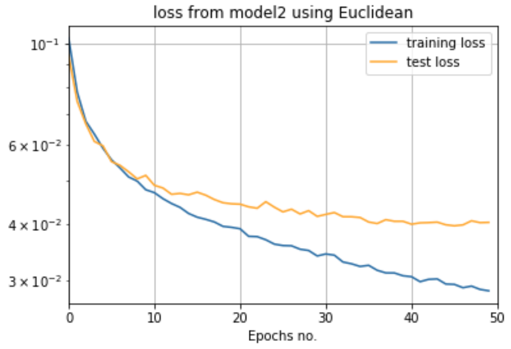


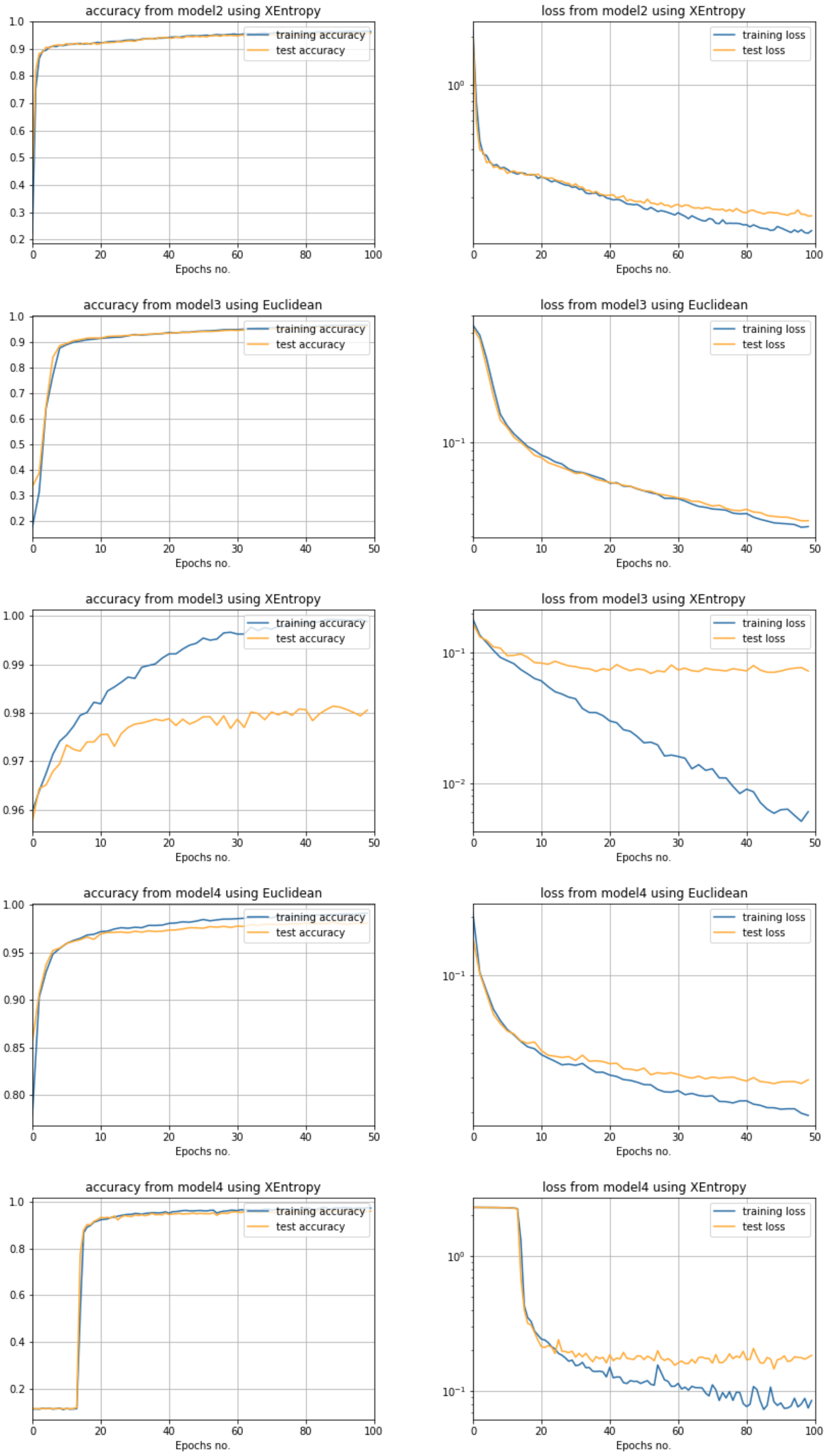


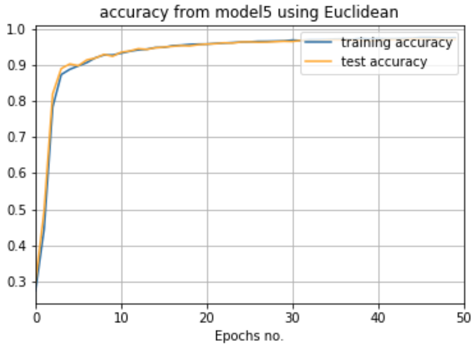


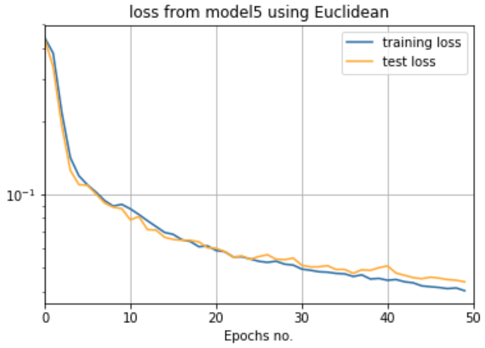


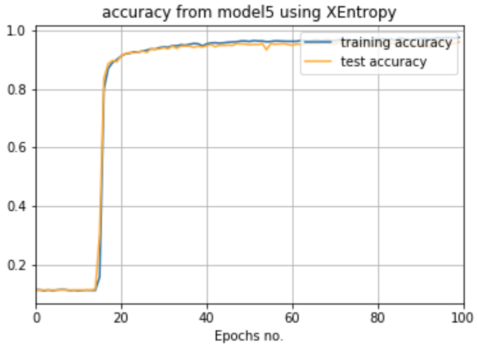


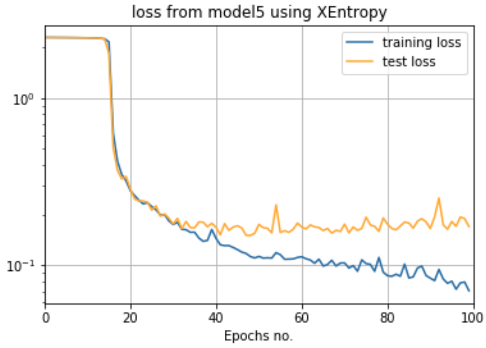



accuracy from model2 using XEntropy
training accuracy
test accuracy
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
20
40
60
80
100
Epochs no.
loss from model2 using XEntropy
training loss
test loss
10^0
0
20
40
60
80
100
Epochs no.
accuracy from model3 using Euclidean
training accuracy
test accuracy
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
10
20
30
40
50
Epochs no.
loss from model3 using Euclidean
training loss
test loss
10^{-1}
0
10
20
30
40
50
Epochs no.
accuracy from model3 using XEntropy
training accuracy
test accuracy
1.00
0.99
0.98
0.97
0.96
0
10
20
30
40
50
Epochs no.
loss from model3 using XEntropy
training loss
test loss
10^{-1}
10^{-2}
0
10
20
30
40
50
Epochs no.
accuracy from model4 using Euclidean
training accuracy
test accuracy
1.00
0.95
0.90
0.85
0.80
0
10
20
30
40
50
Epochs no.
loss from model4 using Euclidean
training loss
test loss
10^{-1}
0
10
20
30
40
50
Epochs no.
accuracy from model4 using XEntropy
training accuracy
test accuracy
1.0
0.8
0.6
0.4
0.2
0
20
40
60
80
100
Epochs no.
loss from model4 using XEntropy
training loss
test loss
10^0
10^{-1}
0
20
40
60
80
100
Epochs no.


accuracy from model5 using Euclidean
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
10
20
30
40
50
Epochs no.
training accuracy
test accuracy


loss from model5 using Euclidean
10^{-1}
0
10
20
30
40
50
Epochs no.
training loss
test loss


accuracy from model5 using XEntropy
1.0
0.8
0.6
0.4
0.2
0
20
40
60
80
100
Epochs no.
training accuracy
test accuracy


loss from model5 using XEntropy
10^{0}
10^{-1}
0
20
40
60
80
100
Epochs no.
training loss
test loss