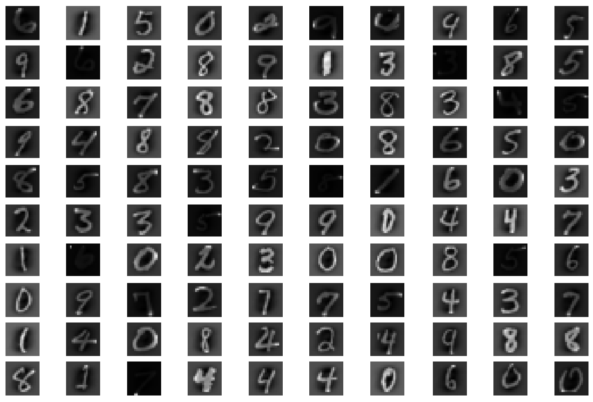
ECS308/658: Assignment 4 Report

DEEP POOJA
17074

1 Objective

To experiment with the use of Neural Networks for a multiclass classification problem, and try and interpret the high-level or hidden representations learnt by it. Also, to try and understand the effects of various parameter choices such as the number of hidden layers, the number of hidden neurons, and the learning rate.

2 Visualizing the Data

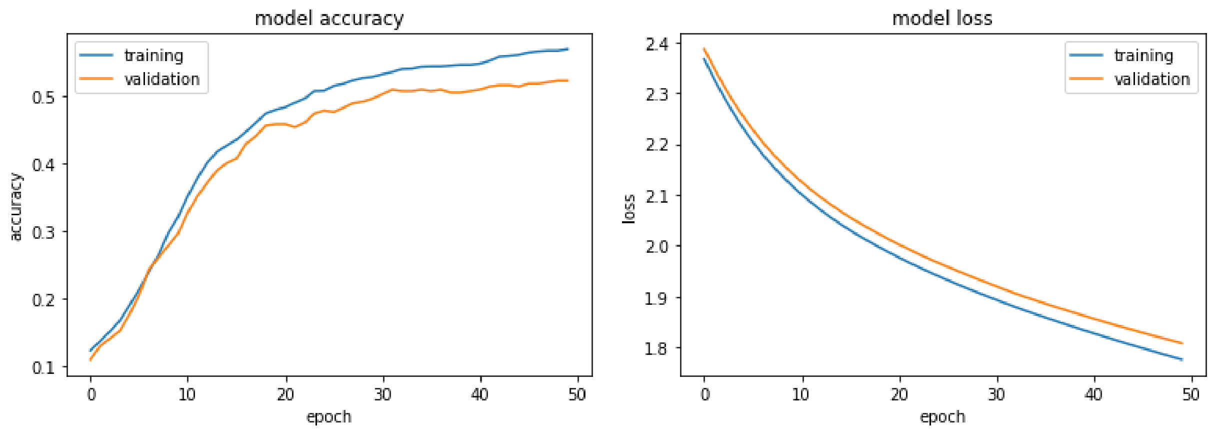


Seems gray-scale images are somewhat distorted.

3 Standard backpropagation neural net

3.1 Single hidden layer

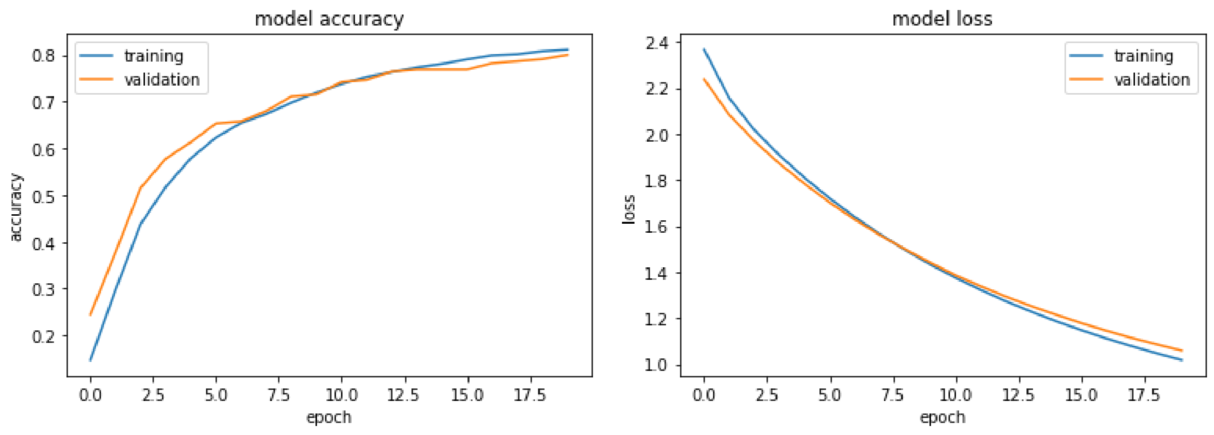
Starting with overly simple model, just with five neurons.
35us/step - loss: 1.7760 - acc: 0.5689 - val_loss: 1.8083 - val_acc: 0.52
Test loss: 1.78
Test accuracy: 0.568



Observation: Clearly, model is overfitting the data since validation accuracy $<$ training accuracy.

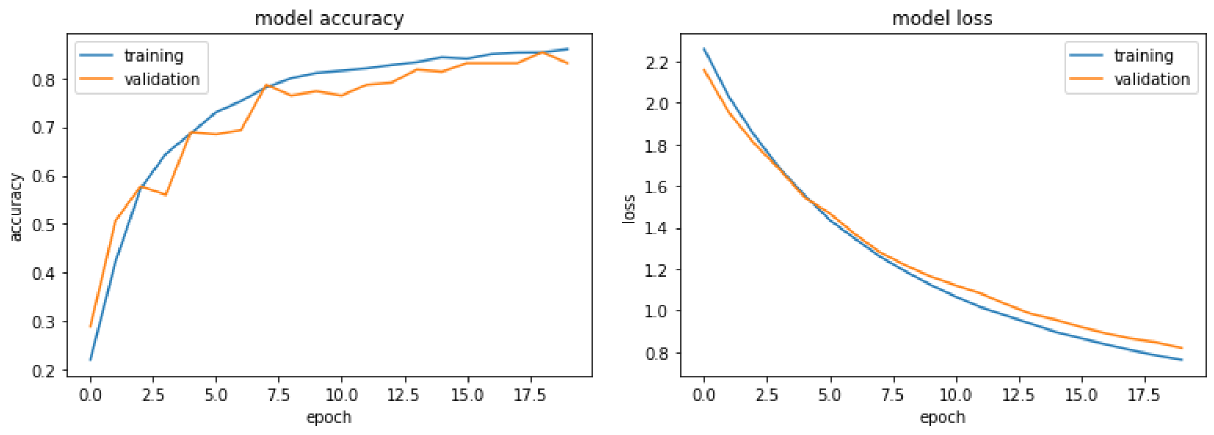
Increasing the number of neurons to 256.
60us/step - loss: 1.0178 - acc: 0.8114 - val_loss: 1.0588 - val_acc: 0.8000

Test loss: 1.0
Test accuracy: 0.816



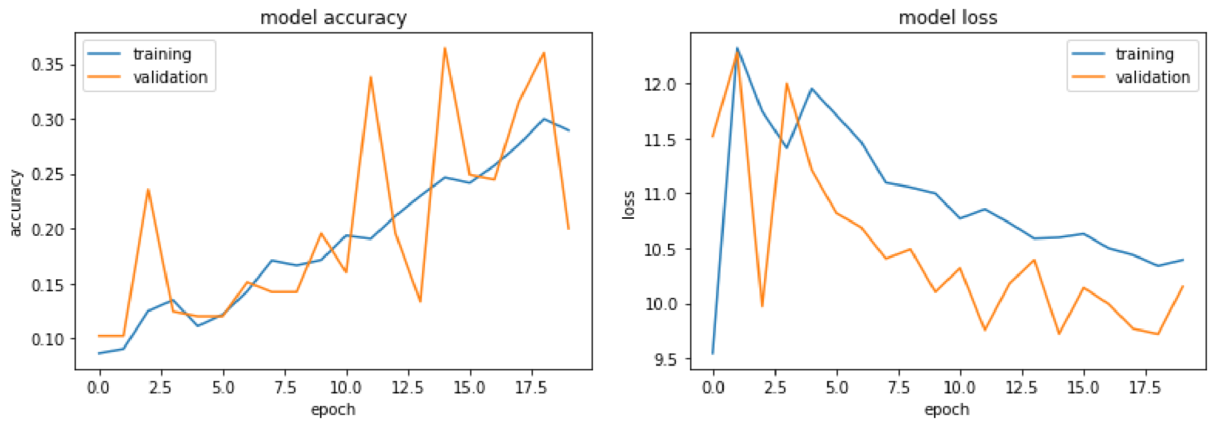
Observation: Best accuracy we got is around 80% near 5-7 epoch after which model starts overfitting.Increasing the number of neurons in hidden layer increases accuracy... but model starts overfitting the data early.

Let's add more neurons, say 2048 neurons.
245us/step - loss: 0.7612 - acc: 0.8598 - val_loss: 0.8185 - val_acc: 0.8311
Test loss: 0.747
Test accuracy: 0.863



Observation: Adding more neurons to the model, it did better..but it become sluggish as complexity of network increases, also no. of neurons increased by 10 times the previous model, but accuracy does not increase that much. Validation accuracy is fluctuating , suggesting that the model is overfitting

Let's add more neurons and see what happens.
2ms/step - loss: 10.3928 - acc: 0.2894 - val_loss: 10.1503 - val_acc: 0.2000
Test loss: 10.6
Test accuracy: 0.203

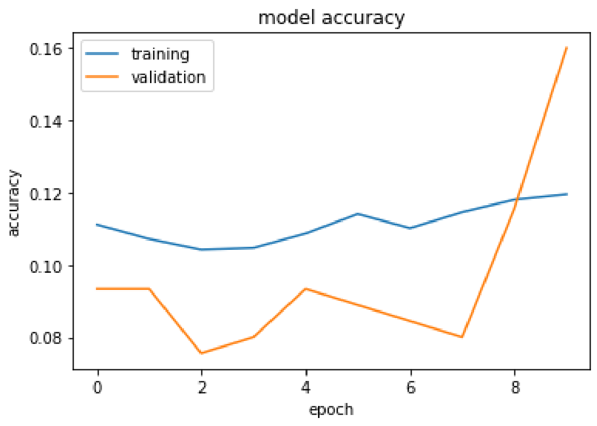


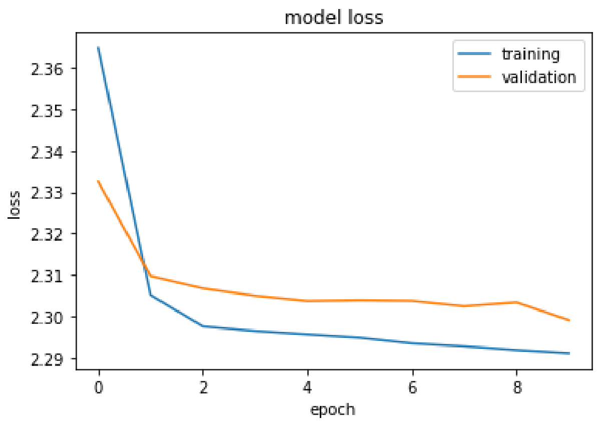
Observation: This network took longer to train. It appears to be overfit as early as the second training epoch, and indeed when we run the test data it's worse than the training performance. Not only that, but it actually performs significantly WORSE than the simpler networks above. Five nodes performed about as well ten thousand nodes, but 256 nodes performed better than either of these.

S.No.	No. of neurons	Test_loss	Test_acc	speed
1.	5	1.78	0.568	35us/step
3.	256	1.0	0.816	60us/step
4.	2048	0.747	0.863	245us/step
4.	20048	10.6	0.203	2ms/step

3.2 Multilayer perceptrons

Since we got good accuracy till now with 256 neurons, so we will keep this number and add more hidden layers say two more that is total three with 256 neurons in each layer.
110us/step - loss: 2.2911 - acc: 0.1195 - val_loss: 2.2991 - val_acc: 0.1600
Test loss: 2.29
Test accuracy: 0.167





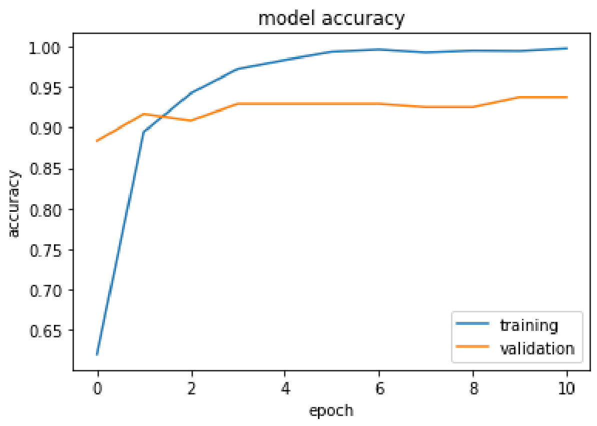
Observation: This model has performed unexpectedly poorly.

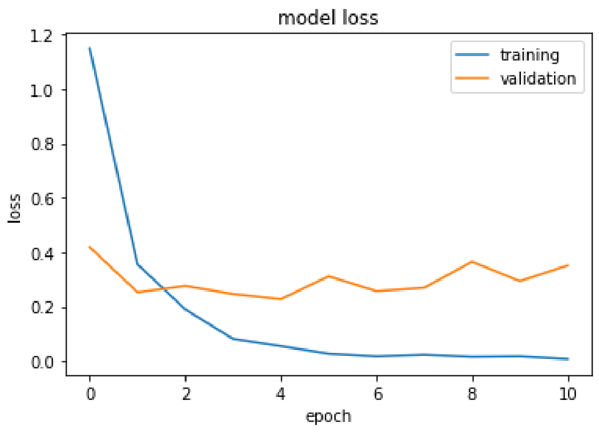
3.3 Final Model

After experimenting with no. of hidden layer, finally got a model which gives satisfactory accuracy on test set.

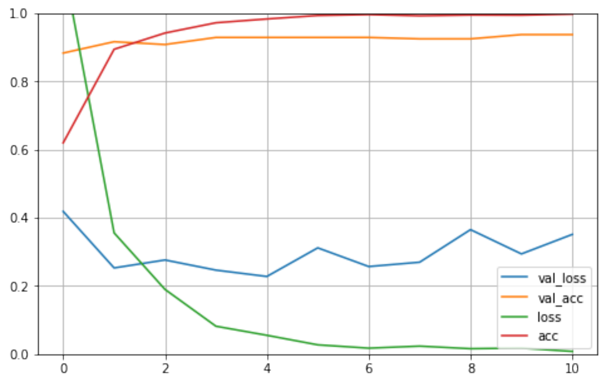
Model's Evaluation

70us/step - loss: 0.0080 - acc: 0.9977 - val_loss: 0.3514 - val_acc: 0.9375
Test loss: 0.303
Test accuracy: 0.932

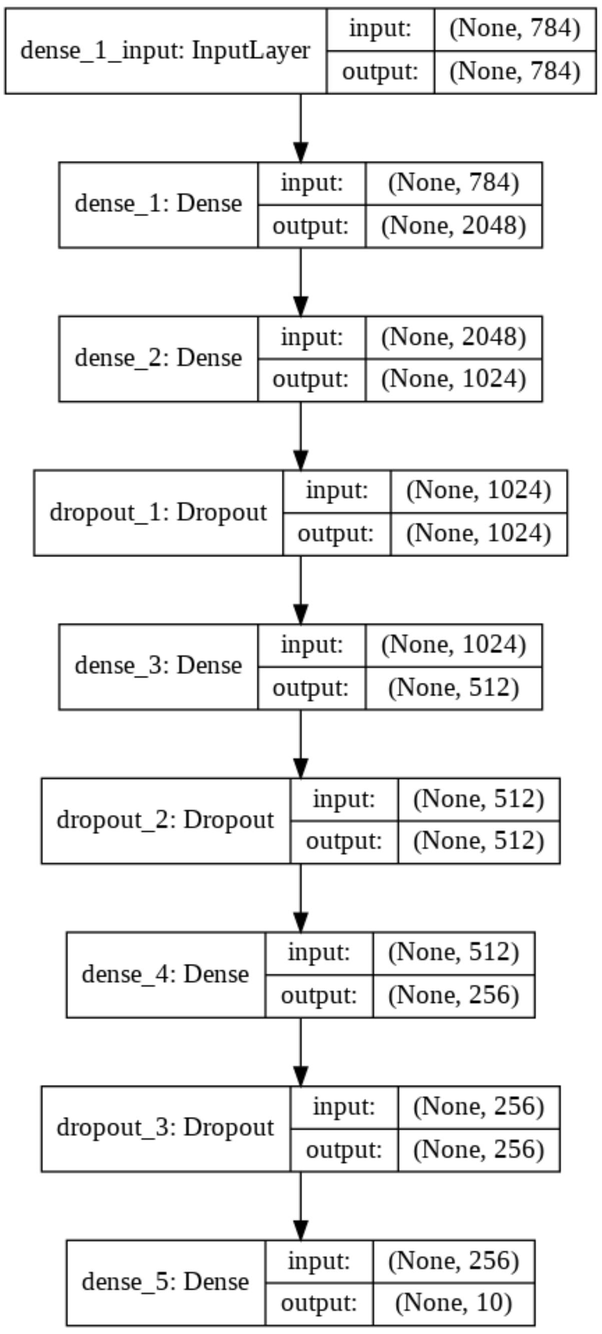




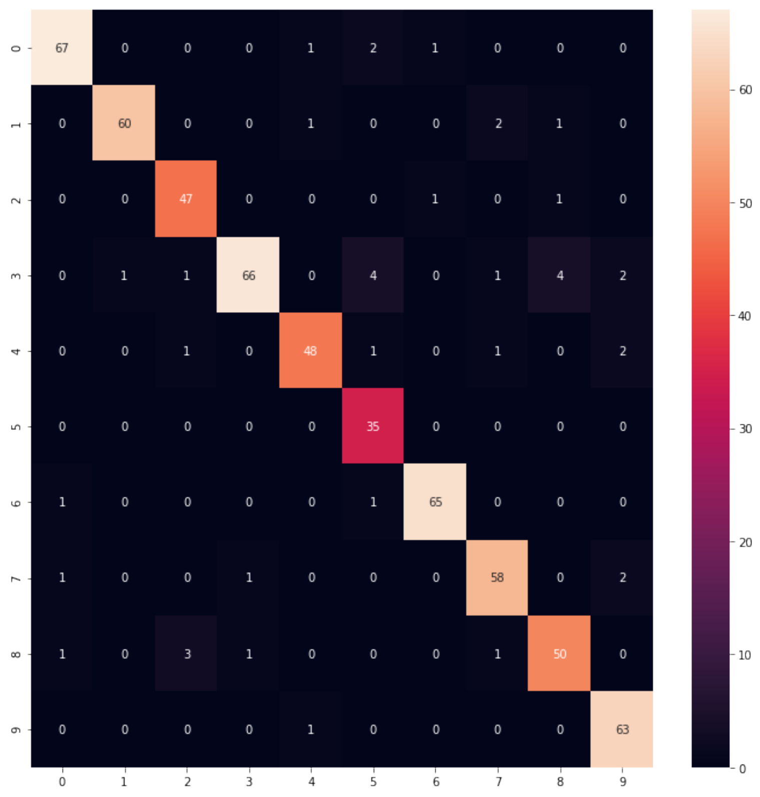
Learning Curve



Model's Architecture

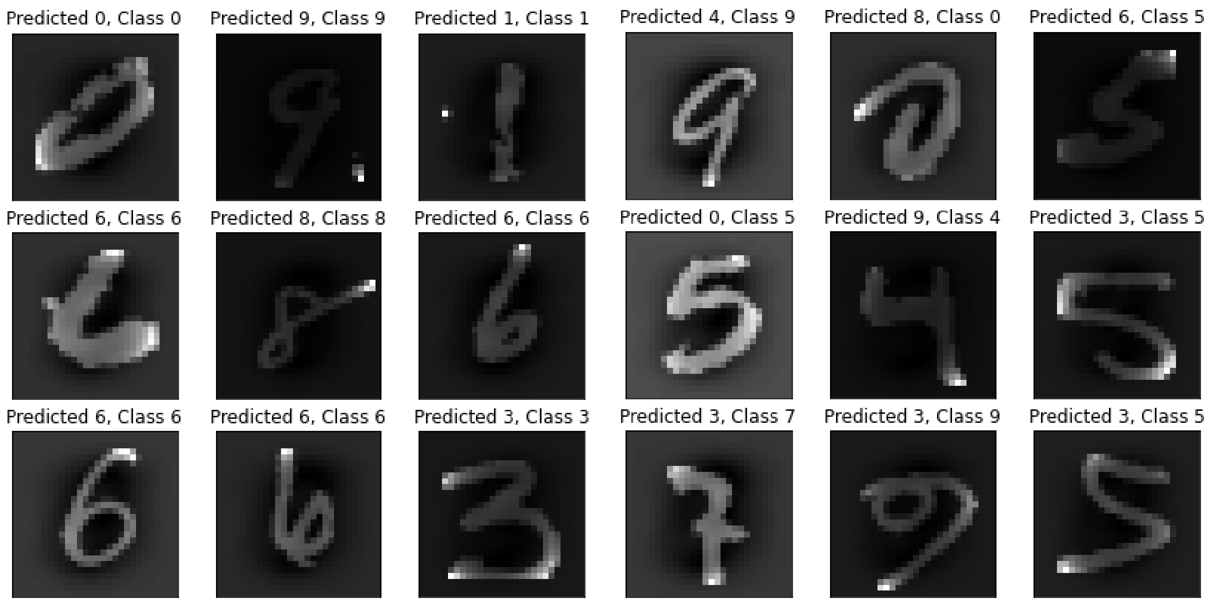


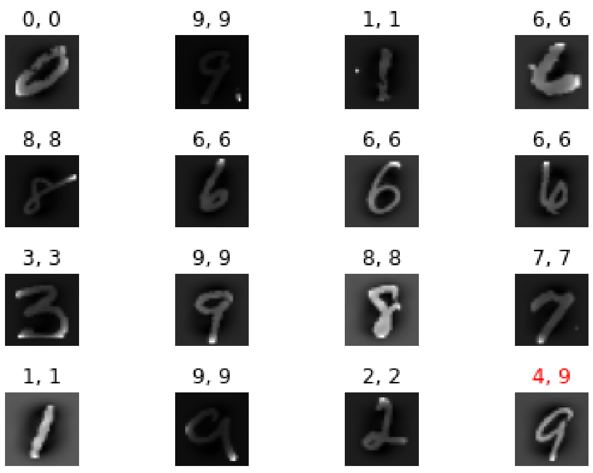
Confusion Matrix



	0	1	2	3	4	5	6	7	8	9
0	67	0	0	0	1	2	1	0	0	0
1	0	60	0	0	1	0	0	2	1	0
2	0	0	47	0	0	0	1	0	1	0
3	0	1	1	66	0	4	0	1	4	2
4	0	0	1	0	48	1	0	1	0	2
5	0	0	0	0	0	35	0	0	0	0
6	1	0	0	0	0	1	65	0	0	0
7	1	0	0	1	0	0	0	58	0	2
8	1	0	3	1	0	0	0	1	50	0
9	0	0	0	0	1	0	0	0	0	63

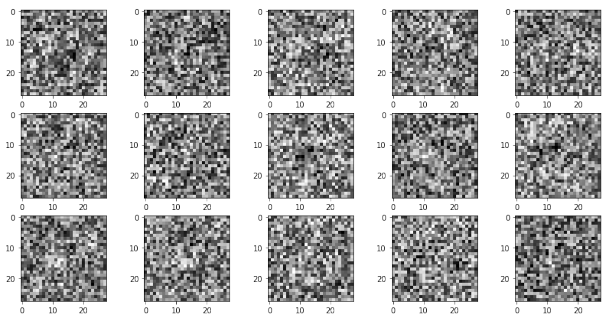
When did our model performs poorly



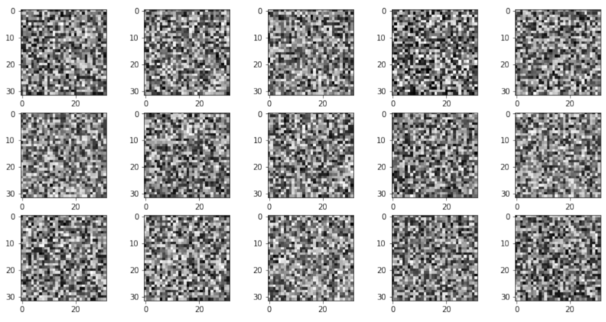


visualizing the learned representation of neuron network

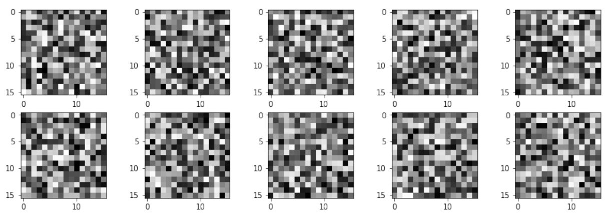
first hidden layer



middle hidden layer

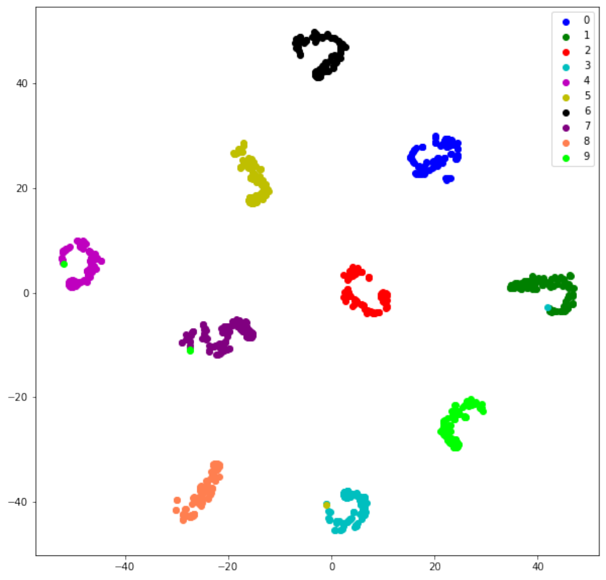


last hidden layer



This seems very high level representation of curves and edges, which does make sense even in last hidden layer of the network. There might be some abstract pattern which is not perceivable by human eye.

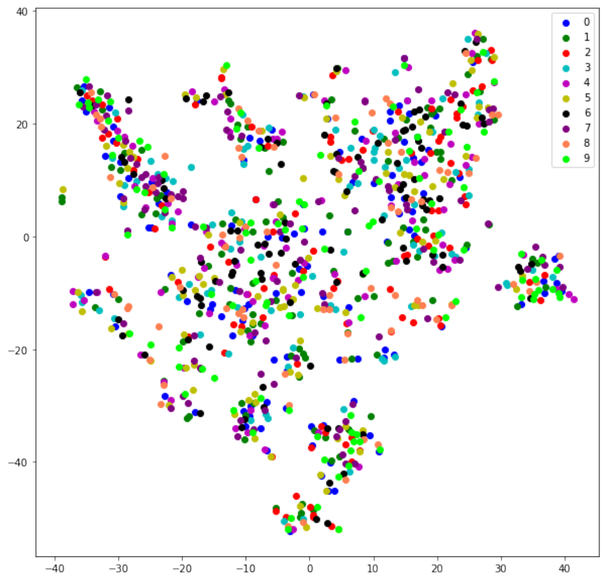
TSE representation of hidden features
This is 2-D projection of representation learned the neuron network.



4 Comparison with PCA features

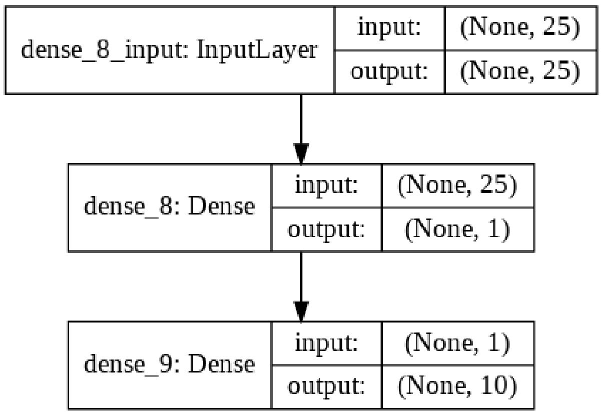
TSE representation of PCA features

This is 2-D projection of PCA features.

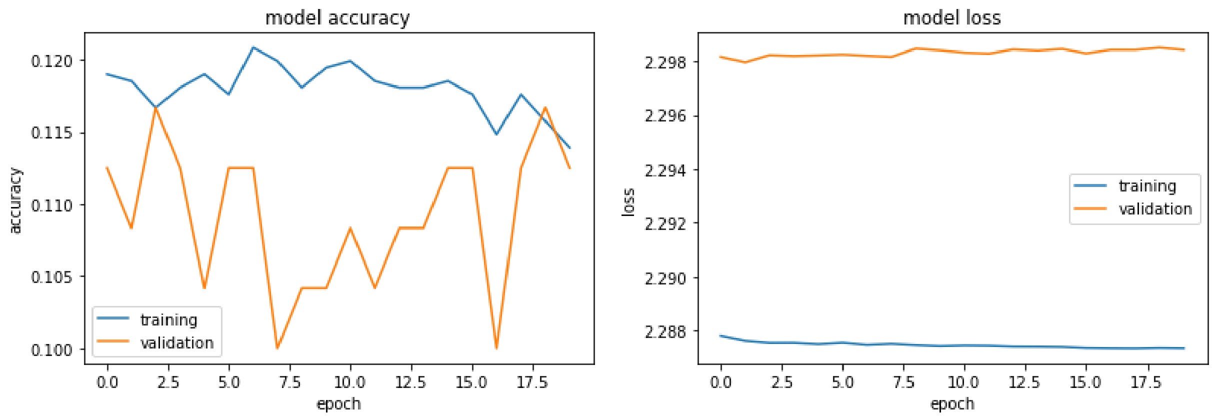


Clearly, neuron network has learned a better or almost accurate representation than PCA one.In representation learned by neuron network all the 10 classes are clearly separate except for few misclassification, but in PCA representation there is overlap of classes to a good extend and things seems hodgepodge.

4.1 No hidden layers

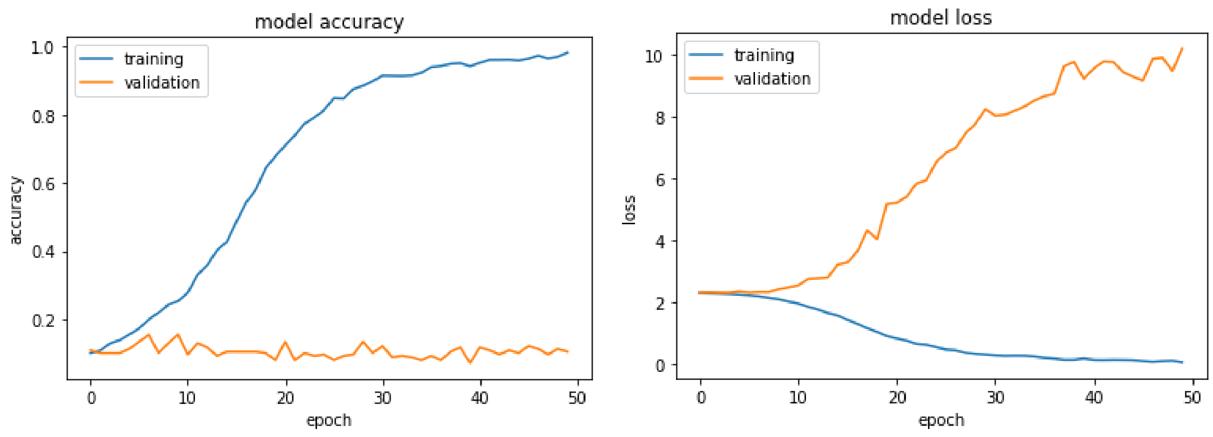


41us/step - loss: 2.2874 - acc: 0.1139 - val_loss: 2.2984 - val_acc: 0.1125
Test loss: 2.3
Test accuracy: 0.107



with added hidden layers

Adding hidde layers to pca model doesn't help, accuracy has not even improved a little bit.
Test loss: 10.5
Test accuracy: 0.107



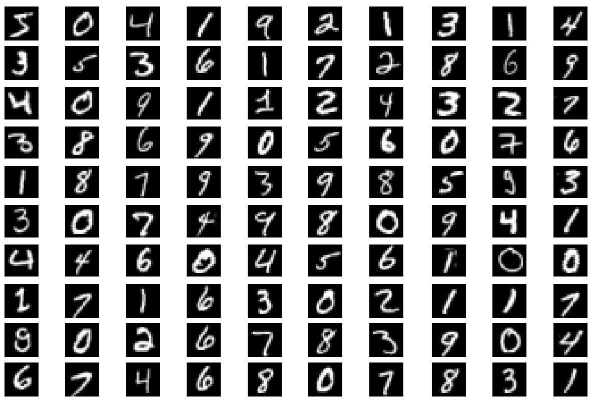
Observations:
The results obtain on training the model with PCA features are worse than raw pixels, in fact results are so poor that we cannot even compare.
The reason behind this, i am not able to fully understand, but in my opinion these features doesn't build up to an image (hand written images), these are some value doesn't have pattern of being digits. So it's like neuron network is fed with some random value without having any pattern of given labels.

5 Advanced neural networks

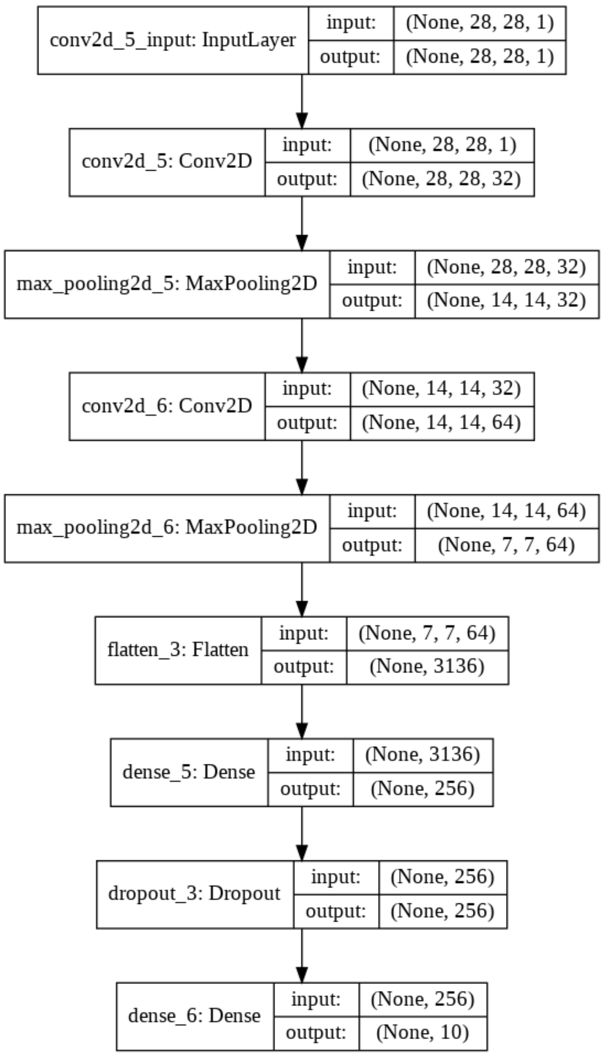
5.1 Conv Net

Visualizing the Data

The full MNIST dataset available on Yann Lecun website is in UBYTE File format. which needs to be converted into readable format. we use this `https://github.com/datapythonista/mnist` repository which already did the job for us.

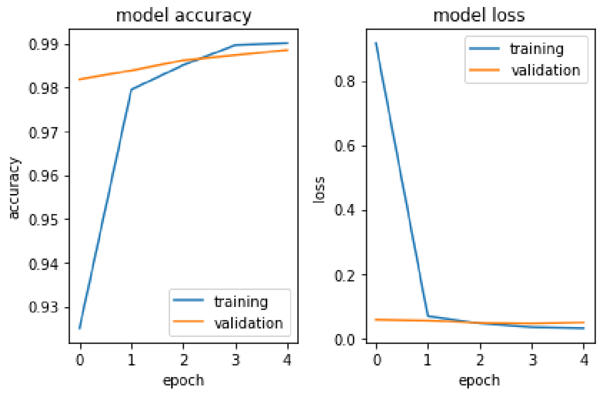


Model's architecture



Model's evaluation

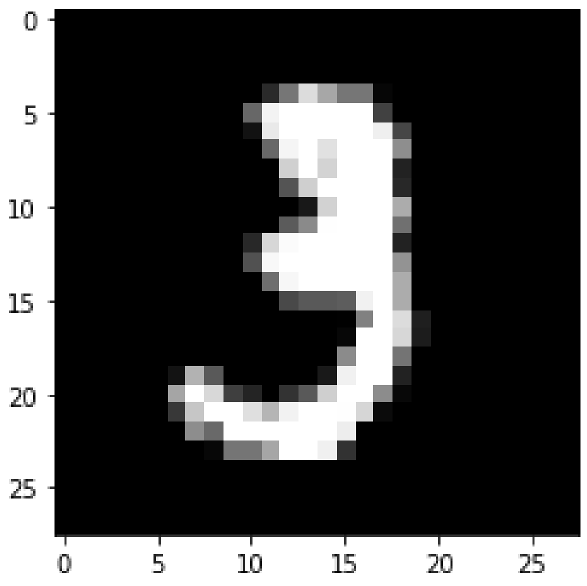
```
1ms/step - loss: 0.0310 - acc: 0.9901 - val_loss: 0.0489 - val_acc: 0.9885
Test loss: 0.0333
Test accuracy: 0.99
```



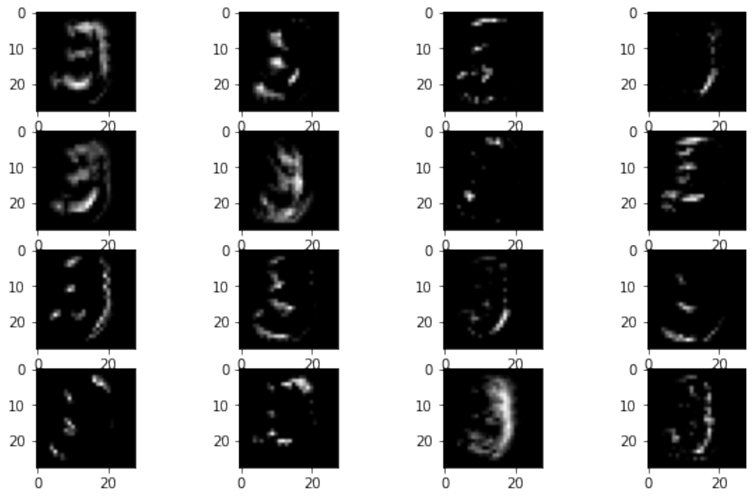
checking the model's performance

As expected, our model performs pretty well.
visualizing the learned representation by the conv net

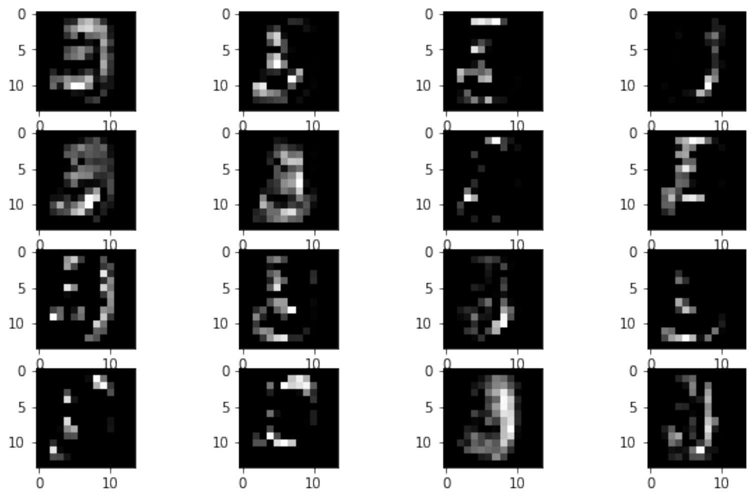
Here, I visualize with digit 3(randomly chosen from the train image set).



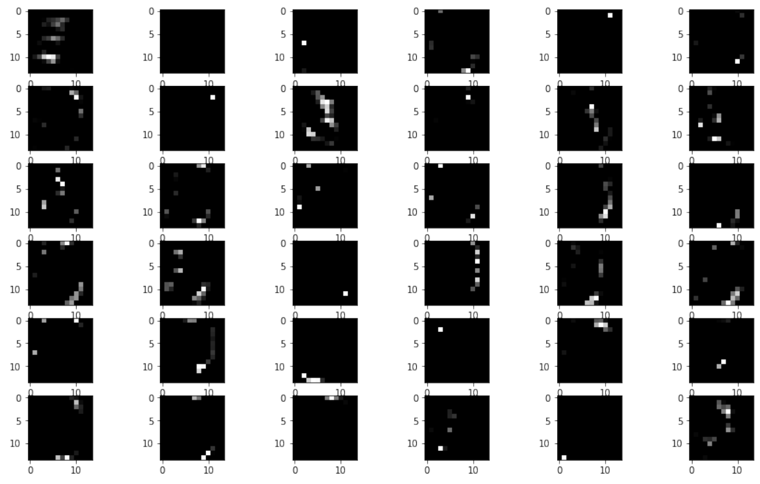
first layer



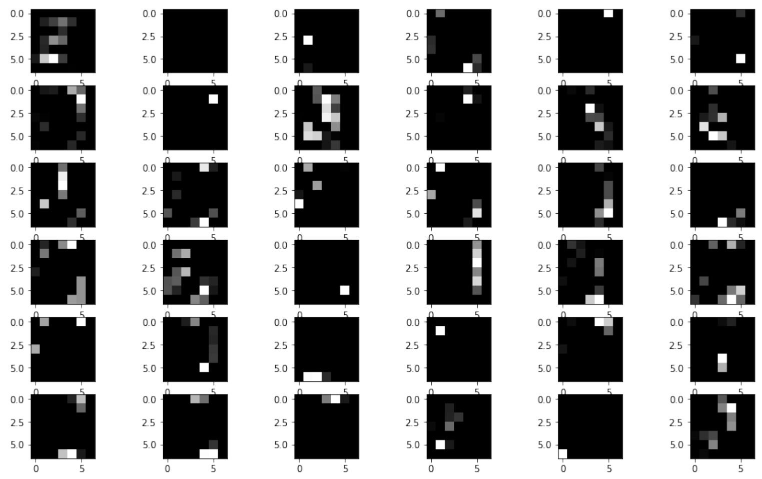
second layer



third layer

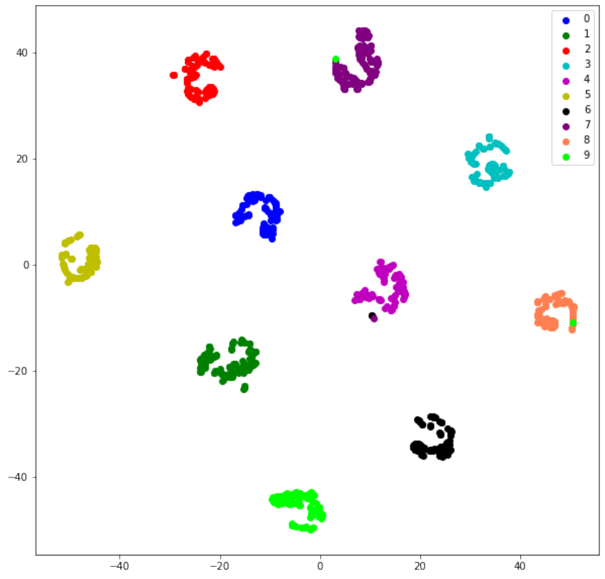


fourth layer



Observation:
1. In first and second layers,pattern learned seems quite reasonable that is natural or intuitive, but when we dive into more deeper layer, patterns becomes as abstract as standard neural net trained above.

TSE representation of hidden features learned by Conv Net



Quite similar to standard net trained above except for less number of misclassification

Changing different parameters of the model, there is changes in the learned representation, but since learned representation is abstract, it is not quite helpful in tuning the parameters so that the model can learn best representation(hidden features).

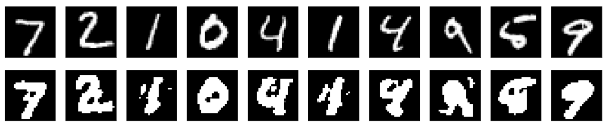
The variation arises due to change in method in which features are extracted, for example if change the size of filters or the size of striding or padding, the features (edges and curves) will be extracted differently, hence change in learned representation, that does affect the accuracy of the model.

5.2 Sparse Autoencoder

5.2.1 simple sparse Autoencoder

With single layer of encoder and decoder.

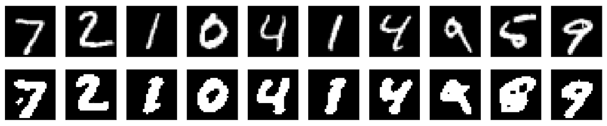
visualizing the reconstructed inputs and the encoded representations



5.2.2 Deep sparse Autoencoder

Added more layers of encoder and decoder.

visualizing the reconstructed inputs and the encoded representations



Observations:

The encoded representations learned by sparse autoencoder are more natural or intuitive as compared to Conv Net and standard Net trained above.

The representation learned y sparse autoencoder also varys subject to parameter change.

6 References

`https://blog.keras.io/building-autoencoders-in-keras.html`
`https://towardsdatascience.com/pca-using-python-scikit-learn-e653f8989e60`