Handwritten Character Recognition using SVM, NN and CNN

Presented by Jason Cho and Hannah Lee

Link to GitHub repo : https://github.coecis.cornell.edu/hl838/MNIST

Project Objective

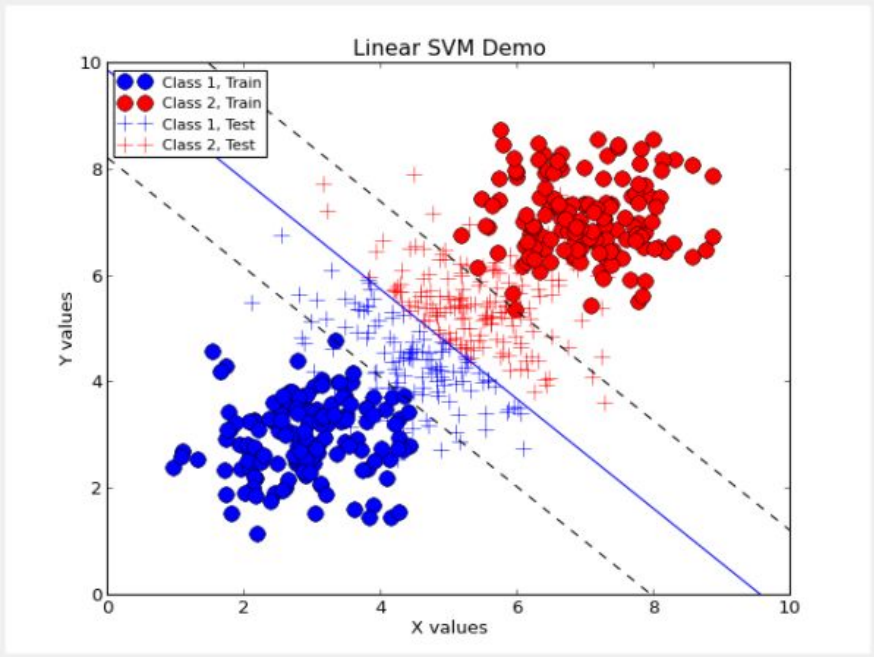
- Develop an image recognition model
  - EMNIST

- Explore different ML algorithms
  - Support Vector Machine
  - Feed Forward Neural Network
  - Convolutional Neural Network.

Evaluation method

- **Cross Validation**
  - **Training set:** dataset that is used to train the model.
  - **Validation set:** dataset that is used to tune our model.
  - **Testing set:** dataset to estimate the accuracy of the model overall.
- Training set vs validation set
- 1. General performance of the model - how high is the accuracy score
- 2. Overfitting - validation accuracy
  - NN, CNN: Dropout
  - SVM: finding the right kernel function

SVM



https://randomforests.wordpress.com/2014/01/29/a-linear-support-vector-machine/

Pros and Cons

Advantages

- Effective in high dimensional spaces
- Algorithm utilizes a so-called “support vector”, that is a subset of the training dataset
  - Memory efficient
  - Strong against overfitting
- Allows us to use different kernel functions including linear, polynomial, radial basis function (RBF), etc.

Disadvantages

- algorithm is sensitive to our choice of **a kernel function ->** difficulty in finding an appropriate kernel function

Construction of the model

- Used SVM provided by scikit-learn to build a model.
- Mainly focused on evaluating different kernel functions and deciding which kernel function is appropriate for the particular problem
  - We thought that the performance of the model mostly depended on our choice of Kernel functions.
  - Tested total of 3 different kernel functions: **radial basis, linear, and sigmoid**

Evaluating the model

Trained with 10,000 datapoints

- **Radial Basis Function: ~70 percent accuracy**
- **Linear: ~71 percent accuracy**
- **Sigmoid: ~65 percent accuracy**
- Linear function > radial basis function **BUT**

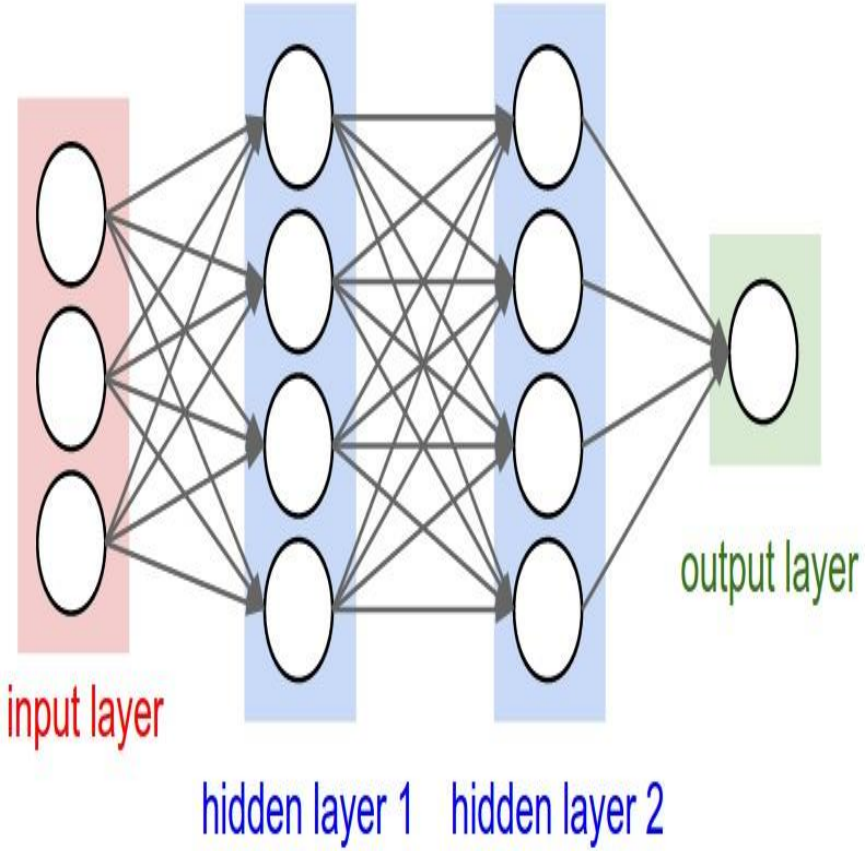
accuracy score = 0.753798076923
report =

	precision	recall	f1-score	support
1	0.61	0.68	0.65	800
2	0.78	0.80	0.79	800
3	0.83	0.83	0.83	800
4	0.73	0.70	0.72	800
5	0.82	0.79	0.81	800
6	0.79	0.75	0.77	800
7	0.71	0.49	0.58	800
8	0.67	0.75	0.71	800
9	0.61	0.63	0.62	800
10	0.73	0.82	0.77	800
11	0.72	0.74	0.73	800
12	0.54	0.67	0.60	800
13	0.85	0.91	0.88	800
14	0.71	0.73	0.72	800
15	0.82	0.90	0.86	800
16	0.81	0.85	0.83	800
17	0.68	0.65	0.66	800
18	0.76	0.66	0.71	800
19	0.93	0.83	0.87	800
20	0.66	0.65	0.65	800
21	0.80	0.82	0.81	800
22	0.77	0.81	0.79	800
23	0.88	0.88	0.88	800
24	0.83	0.75	0.79	800
25	0.75	0.71	0.73	800
26	0.91	0.79	0.84	800
avg / total	0.76	0.75	0.75	20800

accuracy score = 0.985835694051
report =

	precision	recall	f1-score	support
1	1.00	1.00	1.00	235
2	1.00	1.00	1.00	216
3	1.00	1.00	1.00	246
4	1.00	1.00	1.00	230
5	1.00	1.00	1.00	215
6	1.00	1.00	1.00	206
7	1.00	0.99	0.99	222
8	1.00	1.00	1.00	254
9	0.82	0.85	0.83	231
10	1.00	0.99	0.99	255
11	1.00	1.00	1.00	221
12	0.85	0.83	0.84	238
13	1.00	1.00	1.00	240
14	1.00	1.00	1.00	221
15	1.00	1.00	1.00	229
16	1.00	1.00	1.00	236
17	0.99	1.00	0.99	215
18	1.00	1.00	1.00	219
19	1.00	1.00	1.00	248
20	1.00	1.00	1.00	240
21	1.00	1.00	1.00	254
22	1.00	1.00	1.00	254
23	1.00	1.00	1.00	230
24	1.00	1.00	1.00	202
25	1.00	1.00	1.00	228
26	1.00	1.00	1.00	216
avg / total	0.99	0.99	0.99	6001

Neural Network



https://www.pyimagesearch.com/2016/09/26/a-simple-neural-network-with-python-and-keras/

Pros and Cons

Advantages

- Neural networks have shown successes in many image recognition tasks.
- Good for nonlinear data with large number of inputs
  - images are a good example of such data.

Disadvantages

- Disadvantages of neural networks include its empirical nature of model development and proneness to overfitting.
- Training a multi-layer neural network is difficult in that it requires a lot of hyperparameter tuning.

Construction of the model

Baseline model
→ evaluation
→ optimization
→ evaluation
→ optimization
→ and so on...

Baseline model

- 1 hidden layer with the same number of neurons as input.
- Activation functions used: ReLu for hidden layers, Softmax for output layer.
- Overfitting is reduced by using the Dropout feature of keras. (Dropout=0.2)
- Loss function: categorical_crossentropy(good for predicting multiple mutually-exclusive classes)
- Optimizing method: Adam (a form of gradient descent)
- Trained over 20 epochs with weight updates for every image input.

Evaluating the model

Validation Set

- We selected 10% of our training dataset (validation data) to see after each epoch to see the performance of the model against dataset that it hasn't seen during training.
- We decided there has been an overfitting of data when the validation accuracy flattens out.

Test Set

- We used a separate set of 20800 samples (test data) to test how well our final model predicts on an unseen dataset.

Optimization

- Number of Epochs
  - Stop training when the validation accuracy flattens out (manifestation of overfitting)
- Number of neurons in hidden layer(s)
  - Too few neurons: underfitting, too many: overfitting
- Number of hidden layers
  - two layers are said to be sufficient for almost any application since one layer is supposed to approximate most functions
- Dropout rate
  - Reduces overfitting but the rate being too high may result in underfitting

Final Model

Model outline

- Baseline model + 2 hidden layers with 784 neurons

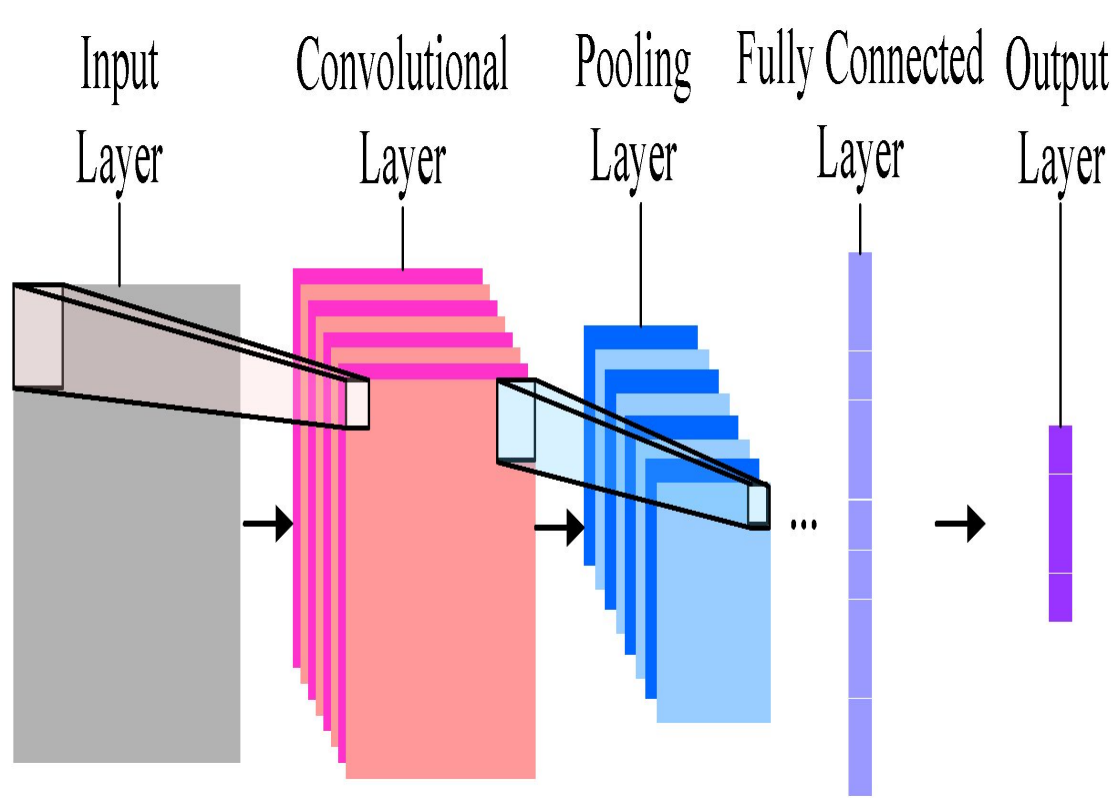
Test

- Accuracy of 0.91 and a loss of 0.30

Validation

- Validation loss kept decreasing until it reached 0.3142, while validation accuracy kept increasing to 0.9010 both around epoch 14.

CNN



http://www.mdpi.com/2078-2489/7/4/61

Pros and Cons

Advantage

- Shows a very high accuracy rate in image recognition tasks in particular
  - CNN is designed to assume that input data is an image.

Disadvantage

- High computational cost -> slow to train

Understanding and implementing CNN

- We came up with the following base line model
  - **Convolutional layer - two 3 by 3 convolutional filters**
  - **Pooling Layer - one 2 by 2 pooling layer**
  - **Fully Connected Layer - 1 fully connected layer with 128 nodes**
- We tried changing different parameters
  - number of nodes in the hidden layers
  - changing the dimension of the filter
  - etc.

Model Evaluation

- Improved accuracy and loss (91 % accuracy with 30 % loss)

- High computation cost (each epoch: 300~600 sec)
    - we believe that carefully calibrating each layers, nodes, number of epochs and other hyperparameters for this particular problem would be another project on its own.

Conclusion

- Overall Accuracy of the model: ~ 90% accuracy
  - Reason for a relatively low accuracy?
- Things we could have done better?
- What we have learned

THANK YOU