Forest Dataset

**Notes/Observations**

-Good dataset to do both shallow and deep learning. Shallow methods got about 70% accuracy whereas the neural networks get about 80% accuracy.

-I’ve run a variety of MLPs with differing layers and units. Does that count as multiple deep learning methods or is MLP regarded as a single method?

-The training curve for optimising ‘C’ the regularisation parameter is strange. The training results get worse for higher values of ‘C’ which isn’t expected. I can’t figure that one out.

Results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Input | Train Accuracy | Test Accuracy | Notes |
| Logistic Regression | Raw Features | 0.682 | 0.680 |  |
| Logistic Regression | 12 Principal Components |  |  |  |
| Random Forest | Raw Features | 0.687 | 0.687 |  |
| SVC | Raw Features | 0.111 | 0.109 |  |
| MLP (100,100) |  | 0.7626 | 0.7777 |  |
| MLP (200,200) |  | 0.810 | 0.808 |  |
| MLP (300,300) |  | 0.835 | 0.833 |  |
| MLP (100,100,100) |  | 0.754 | 0.753 |  |
| MLP (100,100,100,100) |  | 0.7588 | 0.7535 |  |
| **MLP (shallow 2x300 + deep 4x100)** |  | **0.8430** | **0.8682** | **Best Model** |
| 1D CNN (10 x 5 Features, 512 Dense) |  | 0.8041 | 0.8372 |  |

***do we need use github for our project?***

**Tommy:**

5. german\_traffic\_signs:   
 shallow : Random Forest & Softmax Regression

deep: CNN & ANN & RNN

7. pulsar\_stars:   
 shallow : Logistic regression & Decision Tree

deep: ANN & CNN

**SVHM Dataset**

Results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Inputs** | **Train Accuracy** | **Test Accuracy** | **Notes** |
| SVC | Raw Data |  |  | Very difficult to train on entire dataset |
| Random Forest | Raw Data | .229 | 0.235 | Best shallow model |
| Logistic Regression | Raw Data | .235 | .185 | Overfitting (can reduce C parameter) |
| Logistic Regression | Shallow Encoder | .495 | .490 |  |
| Random Forest | Shallow Encoder | .248 | .257 |  |
| Logistic Regression | Deep Encoder | .407 | .419 |  |
| Random Forest | Deep Encoder | .227 | .240 |  |
| Logistic Regression | Convoluted Encoder | .737 | .711 | Best Hybrid Model |
| Random Forest | Convoluted Encoder | .278 | .273 |  |
| SVM | Convoluted Encoder |  |  | Slow to run! |
| CNN (32,32,64,64,512) - CIFAR config |  | 0.8974 | 0.911 |  |
| CNN (20,20,40,40 - 600) |  | .8970 | 0.913 |  |
| CNN (20,40 - 600) |  | 0.8563 | 0.8814 |  |
| CNN (20,20,20,40,40,40 - 600) |  | .9049 | 0.9212 |  |
| CNN (20,20,20,40,40,40 - 500) |  | 0.8989 | 0.9098 |  |
| CNN (20,20,40,40,80,80,1024) |  | 0.9009 | 0.9307 | Best Deep Model |
| CNN (40,40,80,80 - 1024) |  | 0.9074 | 0.9192 |  |
| MLP (100,100,100) |  | .499 | .471 | Deeper with less neurons appears better |

* Tried to generate PCA to utilise in shallow but difficult to run.
* SVC are very difficult and time-consuming to train.
* Many of the models take a long time to train (c30 mins for each model with epochs)
* Shallow methods have not been tune, ie optimal value of ‘C’ has not been determined.

Brett: Questions/Comments.

**Pump Sensor Data**

* Logistic regression and SVM look good.
* MLP. Final layer is only one linear unit. This is producing predicted probabilities greater than 1 which isn’t best practice. The activation could be changes to ‘sigmoid’ to give logistic regression type outcome or possibly softmax (but you have to input binary labels. The loss would also need to change to binary or categorical cross entropy I think. (if it replicated Point Cloud MNIST approach I think it would be better).
* CNN looks good I think.
* A RNN could be worth a try as RNNs are supposed to be good with time series data. I’ll see if I can try something.

**Point Cloud MNIST**

* I’m not sure coeffcient of determination is the right measure for a classification problem? Accuracy is probably the simplest and most recognisable measure.
* Shallow learning looks okay.
* MLP and CNN look good. Are there any better model configurations?

**Pulsar Stars**

● Looks good.

● Any idea why CNN comes out best? Given the dataset doesn’t have any sequence structure, I wouldn’t expect it to do very well but it does? I’m getting good (but not the best) results on the forest dataset too which has no sequence structure.

// In my opinion, the dataset doesn’t have too many features. The structure is to simple to fit any model.

**Traffic Signs**

● I can run the ‘import CV2’ bit which means I can’t run most of the file. Any ideas what I have to do?

//To processing images, Opencv is the best option. Which part can you not run properly?

<https://docs.opencv.org/master/d7/d9f/tutorial_linux_install.html>

● Thanks for the plot model code. I saw something like that on the net but couldn’t make it work but yours does.

// It is not necessary. It just helps me to understand the model easily. I've covered them already.

● Good to see CNN performing best. Any idea if there are better configurations?

Pulsar stars

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Input | Train Accuracy | Test Accuracy | Notes |
| Logistic Regression | Raw Features | 0.977 | 0.979 |  |
| DecisionTreeRegressor | Raw Features | 1.00 | 0.967 |  |
| ANN() | Raw Features | 0.979 | 0.979 |  |
| CNN() | Raw Features | 0.979 | 0.981 |  |

Traffic signs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Input | Train Accuracy | Test Accuracy | Notes |
| Random Forest | 2 dimension | 0.677 | 0.625 | slow |
| Softmax Regression | 2 dimension | 0.91 | 0.802 | Fast |
| ANN() | Raw Features | 0.901 | 0.839 |  |
| CNN() | Raw Features | 0.973 | 0.973 |  |
| RNN |  | 0.759 | 0.672 |  |

**Allen:**

3. Pump\_sensor\_data dataset:   
 shallow : Logistic Regression, SVM, Random Forest

deep: ANN & CNN

4. Point Cloud MNIST 2D dataset:   
 shallow : K-nearest neighbors, Extra trees, Random Forest

deep: ANN & CNN

-- for the deep learning, the y\_test or y\_training data can use to\_categorical

function or in the model.compile method, make loss=’sparse\_categorical\_crossentropy

‘. It both can work.

**Pump\_sensor\_data**

Results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Inputs** | **Train Accuracy** | **Test Accuracy** | **Notes** |
| Logistic Regression | Raw Data | 0.9980 | 0.9979 |  |
| SVM | Raw Data | 0.9998 | 0.9996 |  |
| Random Forest | Raw Data | 1.0000 | 0.9998 | Best model. And the broken samples are too small |
| MLP (100,100) | Raw Data | 0.9949 | 0.9950 |  |
| MLP (200,200) | Raw Data | 0.9971 | 0.9968 | best model with 2 hidden layers and 200 units |
| MLP (300,300) | Raw Data | 0.9965 | 0.9962 |  |
| MLP (100,100,100) | Raw Data | 0.9968 | 0.9967 |  |
| MLP (100,100,100,100) | Raw Data | 0.9958 | 0.9951 | deeper is not better in this dataset |
| CNN  (32, 6, 32, 6)  (1 layer, 100 units) | Raw Data | 0.9915 | 0.9933 | CNN is more stable that the test accuracy is always better than training accuracy |
| CNN  (32, 6, 32, 6)  (2 layer, 200 units) | Raw Data | 0.9900 | 0.9919 |  |
| CNN  (64, 6, 64, 6)  (1 layer, 100 units) | Raw Data | 0.9927 | 0.9946 | more filters have better accuracy |
|  |  |  |  |  |

**Point Cloud MNIST 2D dataset**

Results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Inputs** | **Train Accuracy** | **Test Accuracy** | **Notes** |
| Logistic Regression |  |  |  | Running long time without coming out result |
| K-nearest neighbors | Flatten the 28 x 28 as 1D 784 features | 0.9819 | 0.9688 |  |
| Extra trees | Flatten the 28 x 28 as 1D 784 features | 1.0000 | 0.9706 |  |
| Random Forest | Flatten the 28 x 28 as 1D 784 features | 1.0000 | 0.9700 |  |
| MLP (100,100) | Flatten the 28 x 28 as 1D 784 features | 0.9932 | 0.9768 |  |
| MLP (300,300) | Flatten the 28 x 28 as 1D 784 features | 0.9961 | 0.9799 |  |
| MLP (512,512) | Flatten the 28 x 28 as 1D 784 features | 0.9946 | 0.9811 | training accuracy is better than test accuracy from 2nd epoth |
|  |  |  |  |  |
|  |  |  |  |  |
| CNN  (32, 3x3, 32, 3x3)  (64, 3x3, 64, 3x3)  (1 layer, 512 units) | 28 x 28 2d arrays | 0.994 | 0.9949 | model is more stable that MLP that test accuracy is better that training accuracy |
| LSTM | 28 x 28 2d arrays | 0.9776 | 0.9857 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |