

SIGN LANGUAGE RECOGNITION



-Hritik Rao

PROBLEM STATEMENT

How do we ensure that everyone has a voice? -- Even those who don't have one.

- There are approximately 70 million people, who are hearing and speech impaired.
- Static hand gestures are the most essential facets of gesture recognition
- Sign language is a natural way of interaction for such people, and is vital for their everyday functioning

- Hand Gesture Recognition, a person instructs the machine using his bare hands, whereas images of the persons hand gestures are captured and analyzed in order to determine the meaning of the hand gesture.
- Find and extract features that can be used

APPLICATIONS

- Apps that can facilitate direct feed conversion of sign language to letters/ sounds. (Human Computer Interface)
- Emergency First Responders
- Robot Control
- Sign Language and Terrorism.

Sign Language for Emergency Situations

ASL, English, & Spanish

*Bonus Sign Expressions
Language Mini Chart included*



Q SEARCH

NEW YORK POST




NEWS

ISIS using sign language to recruit deaf terrorists

By [Chris Perez](#)

March 9, 2015 | 10:59am

 [ISIS using sign language to recruit deaf terrorists](#)

In a new propaganda film, ISIS militants communicate with potential terrorists using sign language.

ANALYSIS

- This project was chosen keeping in mind the people with hearing and listening disabilities.
- We aim to make their life easier and help the world understand them better, as real development is the growth of society as one, not just the majority.
- On the business front, this is a promising technology for any company that interacts directly with hearing impaired people (eg. restaurant drive throughs,taxis,etc) and with any organization that seeks to include them as a common part of their workforce.

Challenges Faced:

- Time- The full potential of the project and its applications,by implementing it in day to day life using real time video feed, could not be explored.

DATA PREPARATION

- The Sign Language MNIST data came from greatly extending the small number (1704) of the color images included as not cropped around the hand region of interest.
- To create new data, an image pipeline was used based on ImageMagick and Pillow that included cropping to hands-only, gray-scaling, resizing, and then creating at least 50+ variations to enlarge the quantity.
- The modification and expansion strategy was filters ('Mitchell', 'Robidoux', 'Catrom', 'Spline', 'Hermite'), along with 5% random pixelation, +/- 15% brightness/contrast, and finally 3 degrees rotation.

	label	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781	pixel782	pixel783
0	3	107	118	127	134	139	143	146	150	153	...	207	207	207	207	206	206	206	204	203
1	6	155	157	156	156	156	157	156	158	158	...	69	149	128	87	94	163	175	103	135
2	2	187	188	188	187	187	186	187	188	187	...	202	201	200	199	198	199	198	195	194
3	2	211	211	212	212	211	210	211	210	210	...	235	234	233	231	230	226	225	222	229
4	13	164	167	170	172	176	179	180	184	185	...	92	105	105	108	133	163	157	163	164

EXPERIMENTS

This is a list of the models we have used to process and transform the data

- NAIVE BAYES
- SUPPORT VECTOR MACHINE
- K- NEAREST NEIGHBOUR
- LOGISTIC REGRESSION
- CONVOLUTION NEURAL NETWORKS

NAIVE BAYES

- In ML, Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong independence assumptions between the features. Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

The diagram shows the Naive Bayes formula with arrows pointing from descriptive labels to the corresponding parts of the equation:

$$P(c | x) = \frac{P(x | c)P(c)}{P(x)}$$

Labels and their corresponding terms in the formula:

- Likelihood** points to $P(x | c)$
- Class Prior Probability** points to $P(c)$
- Posterior Probability** points to $P(c | x)$
- Predictor Prior Probability** points to $P(x)$

$$P(c | X) = P(x_1 | c) \times P(x_2 | c) \times \cdots \times P(x_n | c) \times P(c)$$

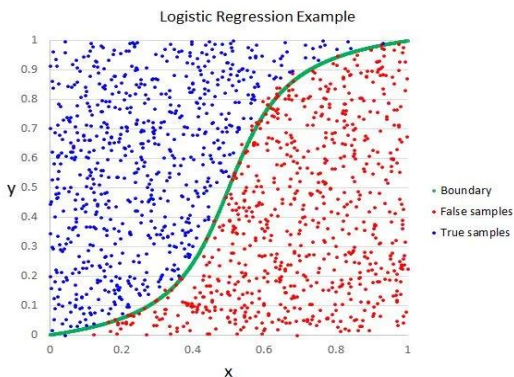


```
...: calc_accuracy("Naive Bayes",label_test,pred)
...:
...:
...: run_nb()
label  pixel1  pixel2  ...  pixel782  pixel783
pixel784
0      3      107      118  ...      204      203
202
1      6      155      157  ...      103      135
149
2      2      187      188  ...      195      194
195
3      2      211      211  ...      222      229
163
4      13     164      167  ...      163      164
179

[5 rows x 785 columns]
train table created
test table created
nb started
accuracy score for Naive Bayes 0.3898494143892917
```

LOGISTIC REGRESSION

- Logistic Regression measures the relationship between one variable (what we want to predict) and the other variables (our features), by estimating probabilities using its underlying logistic function. Basically, it finds the probability of a certain element belonging to a particular category.
- It is a form of supervised learning, and is very commonly used for classification. It has wide applications in Healthcare , Social Sciences & various ML for advanced research & analytics.





```
regression",label_test,pred)
```

```
....:
```

```
....:
```

```
....: run_lr()
```

	label	pixel1	pixel2	...	pixel782	pixel783
pixel784						
0	3	107	118	...	204	203
202						
1	6	155	157	...	103	135
149						
2	2	187	188	...	195	194
195						
3	2	211	211	...	222	229
163						
4	13	164	167	...	163	164
179						

```
[5 rows x 785 columns]
```

```
train table created
```

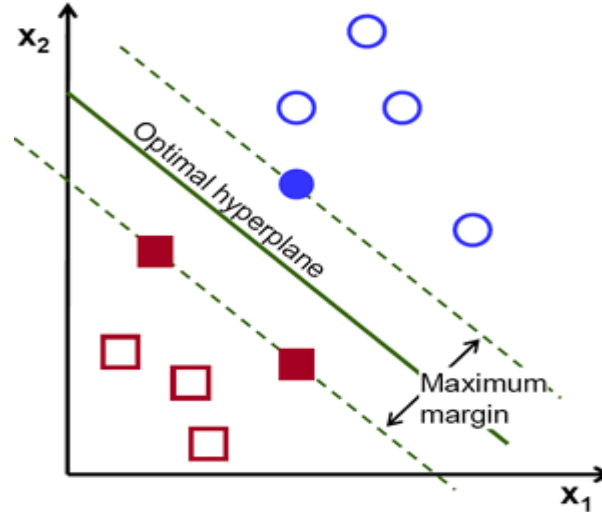
```
test table created
```

```
lr started
```

```
accuracy score for Logistic regression 0.6820970440602342
```

SUPPORT VECTOR MACHINE

- “Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges.
- SVM has a technique called the [kernel trick](#). These are functions which takes low dimensional input space and transform it to a higher dimensional space i.e. it converts not separable problem to separable problem, these functions are called kernels.



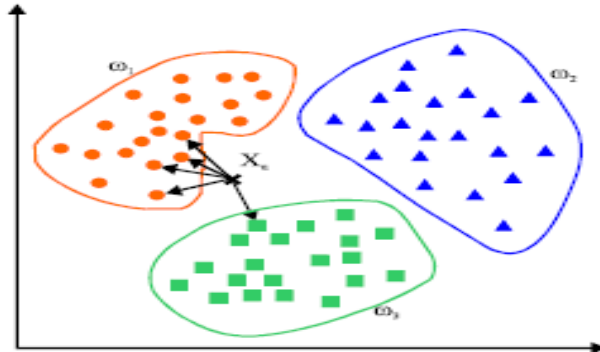


```
...: calc_accuracy("SVM",label_test,pred)
...:
...:
...: run_svm()
label  pixel1  pixel2  ...  pixel782  pixel783
pixel784
0      3      107    118  ...      204      203
202
1      6      155    157  ...      103      135
149
2      2      187    188  ...      195      194
195
3      2      211    211  ...      222      229
163
4     13      164    167  ...      163      164
179

[5 rows x 785 columns]
train table created
test table created
svm started
accuracy score for  SVM 0.6968767428890128
```

K- NEAREST NEIGHBORS

- **KNN** is a **non-parametric, lazy** learning algorithm. Its purpose is to use a database in which the data points are separated into several classes to predict the classification of a new sample point.
- KNN Algorithm is based on **feature similarity**: How closely out-of-sample features resemble our training set determines how we classify a given data point.



```
neighbours",label_test,pred)
```

```
....:
```

```
....:
```

```
....: run_knn()
```

	label	pixel1	pixel2	...	pixel782	pixel783
pixel784						
0	3	107	118	...	204	203
202						
1	6	155	157	...	103	135
149						
2	2	187	188	...	195	194
195						
3	2	211	211	...	222	229
163						
4	13	164	167	...	163	164
179						

```
[5 rows x 785 columns]
```

```
train table created
```

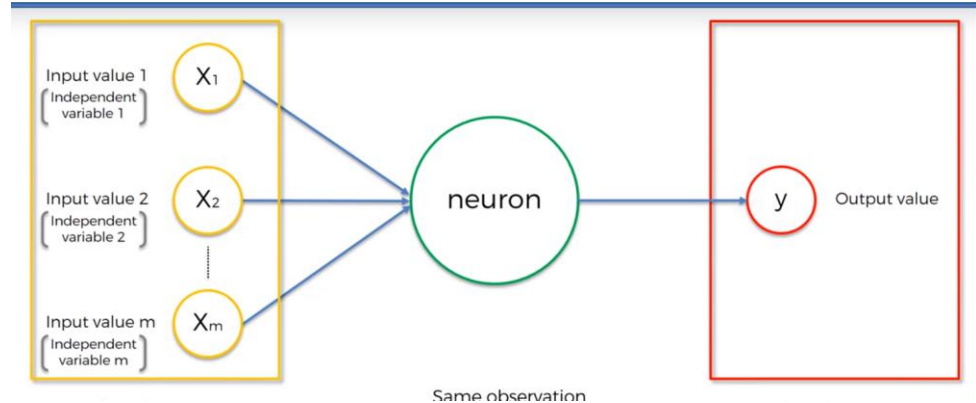
```
test table created
```

```
knn started
```

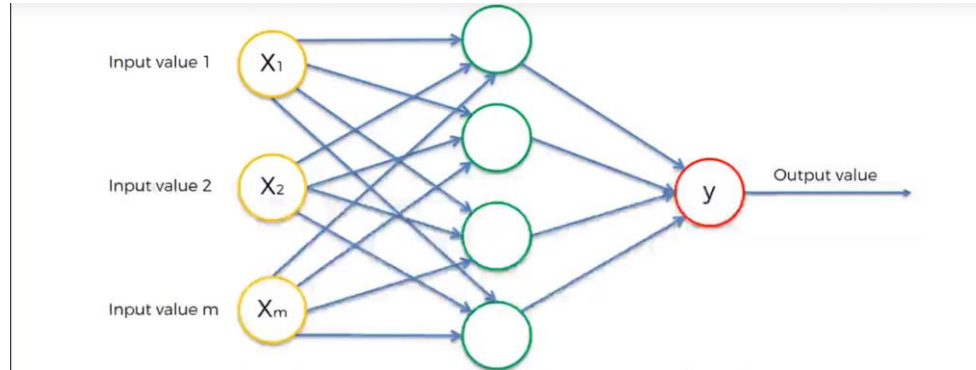
```
accuracy score for K nearest neighbours 0.8039598438371445
```


CNN

CNN are made up of neurons with learnable weights and biases. Each neuron receives several inputs, takes a weighted sum over them, pass it through an activation function and responds with an output.



The convolution layer comprises of a set of independent filters. Parameter sharing is sharing of weights by all neurons in a particular feature map.



Modules used:

Keras: The Python Deep Learning library



Keras

You have just found Keras.

Keras is a high-level neural networks API, written in Python and capable of running on top of [TensorFlow](#), [CNTK](#), or [Theano](#). It was developed with a focus on enabling fast experimentation

Python Data Analysis Library

pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the [Python](#) programming language.

pandas is a [NumFOCUS](#) sponsored project. This will help ensure the success of development of *pandas* as a world-class open-source project, and makes it possible to [donate](#) to the project.

sklearn.model_selection.train_test_split

```
sklearn.model_selection. train_test_split (*arrays, **options)
```

[\[source\]](#)

Split arrays or matrices into random train and test subsets

Quick utility that wraps input validation and `next(ShuffleSplit()).split(X, y)` and application to input data into a single call for splitting (and optionally subsampling) data in a oneliner.

```
#Cross validation technique (Splitting training set in 4:1 as train and validation set)
from sklearn.model_selection import train_test_split
X_train, X_val, y_train, y_val = train_test_split(X, Y, test_size=0.2, stratify=Y, shuffle=True, random_state=42)
```

sklearn.preprocessing.OneHotEncoder

Encode categorical integer features as a one-hot numeric array.

The input to this transformer should be an array-like of integers or strings, denoting the values taken on by categorical (discrete) features. The features are encoded using a one-hot (aka 'one-of-K' or 'dummy') encoding scheme. This creates a binary column for each category and returns a sparse matrix or dense array.

By default, the encoder derives the categories based on the unique values in each feature. Alternatively, you can also specify the categories manually. The OneHotEncoder previously assumed that the input features take on values in the range $[0, \max(\text{values}))$. This behaviour is deprecated.

This encoding is needed for feeding categorical data to many scikit-learn estimators, notably linear models and SVMs with the standard kernels.

Sample	Value	Numerical	Value_Red	Value_Blue	Value_Green
1	Red	1	1	0	0
2	Green	3	0	0	1
3	Blue	2	0	1	0
4	Green	3	0	0	1

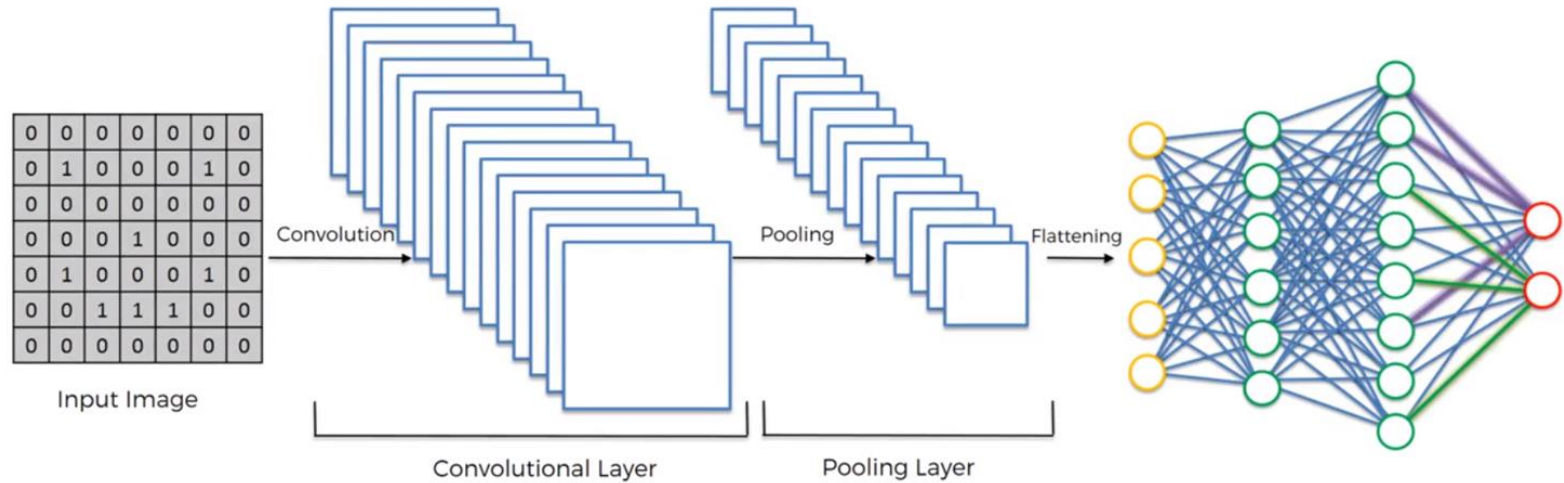
Conv2D

2D convolution layer (e.g. spatial convolution over images).

This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. If `use_bias` is True, a bias vector is created and added to the outputs. Finally, if `activation` is not `None`, it is applied to the outputs as well.

When using this layer as the first layer in a model, provide the keyword argument `input_shape` (tuple of integers, does not include the sample axis), e.g. `input_shape=(128, 128, 3)` for 128x128 RGB pictures in `data_format="channels_last"`.

CNNs, like neural networks, are made up of neurons with learnable weights and biases. Each neuron receives several inputs, takes a weighted sum over them, pass it through an activation function and responds with an output. The whole network has a loss function.



Modal Summary

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 28, 28, 1)	0
conv2d_1 (Conv2D)	(None, 28, 28, 64)	640
max_pooling2d_1 (MaxPooling2D)	(None, 14, 14, 64)	0
conv2d_2 (Conv2D)	(None, 14, 14, 64)	36928
max_pooling2d_2 (MaxPooling2D)	(None, 7, 7, 64)	0
conv2d_3 (Conv2D)	(None, 7, 7, 64)	36928
max_pooling2d_3 (MaxPooling2D)	(None, 3, 3, 64)	0
flatten_1 (Flatten)	(None, 576)	0
dense_1 (Dense)	(None, 128)	73856
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 24)	3096

Usage of callbacks

A callback is a set of functions to be applied at given stages of the training procedure. You can use callbacks to get a view on internal states and statistics of the model during training. You can pass a list of callbacks (as the keyword argument `callbacks`) to the `.fit()` method of the `Sequential` or `Model` classes. The relevant methods of the callbacks will then be called at each stage of the training.

```
Epoch 00019: val_loss improved from 0.00002 to 0.00002, saving model to model.h5
Epoch 20/50
21964/21964 [=====] - 59s 3ms/step - loss: 4.8541e-04 - acc: 0.9998 - val_loss: 2.0200e-05 - val_acc: 1.0000
Epoch 00020: ReduceLROnPlateau reducing learning rate to 1.0000000656873453e-06.
Epoch 00020: val_loss did not improve from 0.00002
Epoch 21/50
21964/21964 [=====] - 59s 3ms/step - loss: 3.3540e-04 - acc: 1.0000 - val_loss: 1.9757e-05 - val_acc: 1.0000
Epoch 00021: val_loss did not improve from 0.00002
Epoch 22/50
21964/21964 [=====] - 60s 3ms/step - loss: 9.0231e-04 - acc: 0.9998 - val_loss: 2.0023e-05 - val_acc: 1.0000
Epoch 00022: val_loss did not improve from 0.00002
Epoch 00022: early stopping
```

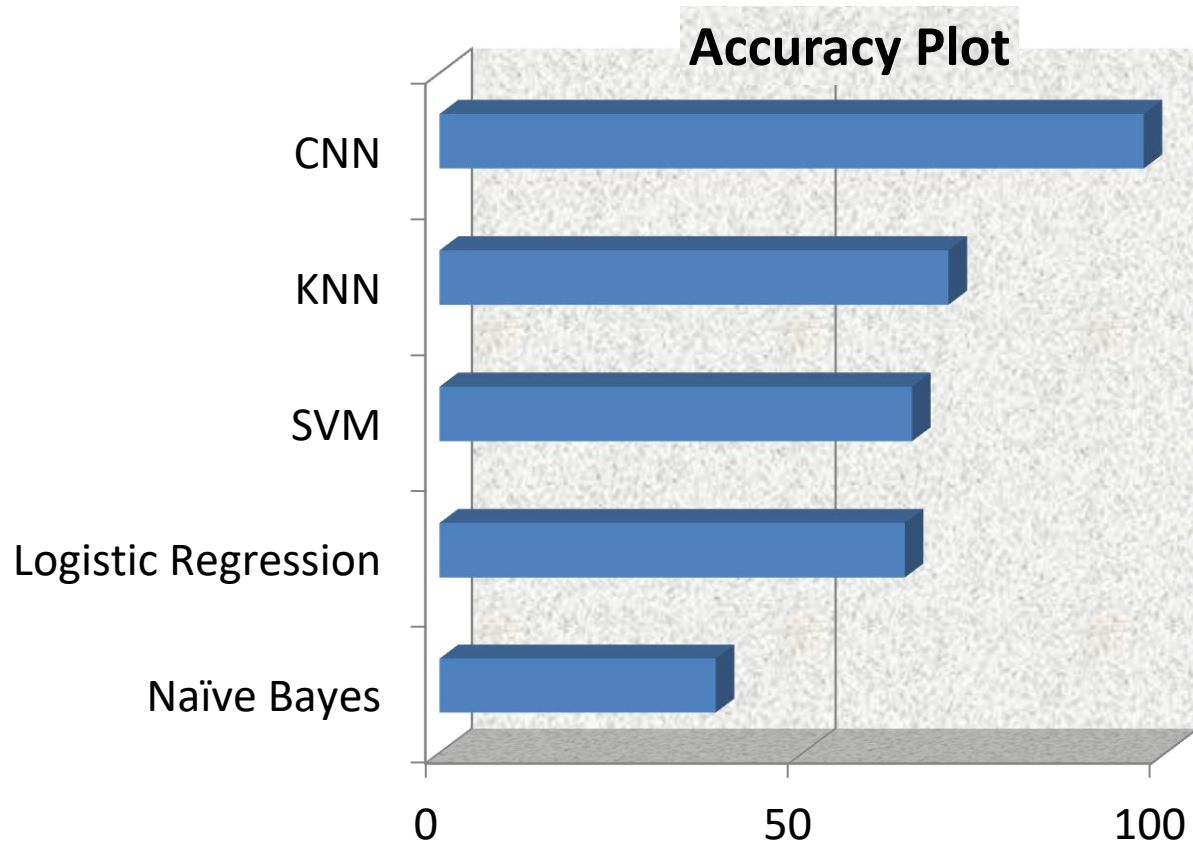


```
Tconfm = {'tp':0, 'fp':0}
for n in range(df2.shape[0]):
    if Y_test.argmax(axis=-1)[n]==Tpreds[n]:
        Tconfm['tp'] +=1
    if Y_test.argmax(axis=-1)[n]!=Tpreds[n]:
        Tconfm['fp'] +=1
print(Tconfm)
```

```
{'tp': 6844, 'fp': 328}
```

```
#Test accuracy
score, acc = model.evaluate(X_test, Y_test, batch_size=32)
#print('Test score:', score)
print('Test accuracy:', acc)
```

```
7172/7172 [=====] - 5s 762us/step
Test accuracy: 0.9542665923034022
```



CONCLUSION

- Sign Language Recognition can be quite useful and important, if it can be merged with various facets of society.
- While the current solution can only be used to recognize static hand gestures, there are many ways this project can be taken forward.
- We can improve this to include Live Feed Sign Language recognition, that can be made available for everyday use.
- We can also include various other forms of sign languages, and can also involve other hand gestures, that can be informative of their nature.
- From helping aid with a person's everyday activity, to enhancing security and helping save lives, Sign Language Recognition has quite the future.

THANK YOU!