16-03-2023

COMPUTER VISION

→ multicourse interdisciplinary
→ not from scratch → Policy Perspective
→ Assignment 3 - Topic Modelling
→ Assignment 4 - in class Essay
in class Coding.

Def:
It is field in AI that derive meaningful information from the digital image.

- Computer vision task
  → image recognition
  → Object detection
  → boundary image segmentation
  → semantic segmentations.
  → Captionisation

→ meaning from the images Geometrically, image Processing.
before ML.

→ Tools → open CV & MATLAB.

Timline

1970's
→ digital image processing
→ Pictorial structure
→ Generalized cylinders
→ optimal flows.

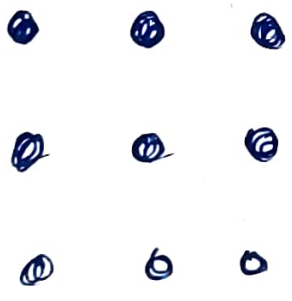
1980's
- Markov random fields.
- texture & focus.
- structure from motion.
- Physically based Modelling.

1990's
- graph cut
- Physics based vision
- factorization

2000's
- inference algorithm
- learning. (ML)

Standard task
- image processing
- feature extractions.
- segmentation
- Captioning.

Edge detections → Edge is where pixel values suddenly changes.



$$\begin{matrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{matrix}$$

Sir's filter

Canny Edge detection

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Horizontal

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Verticle.

Resources

- → google Colab
- → open CV
- → Gimp, etc.

Things to do in coding in Next class.

1. google Colab usecase
2. open CV use case
3. MNIST Better.

Zoom out — Easy
Zoom in — Hard

Problem → In Photograph find the tree Eges.

Salt and pepper noise → Cause by sharp and sudden disturbance in image.

Simple computer vision problem MNSIT Dataset

In [2]:
```
# To plot graoh
import matplotlib.pyplot as plt

# use svm
from sklearn import datasets,svm ,metrics

# to split dataset into train and test
from sklearn.model_selection import train_test_split
```

In [3]:
```
# for confusion matrix and classification report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

In [4]:
```
# load MNSIT() dataset
data =  datasets.load_digits()
```

In [5]:
```
# resize data matric
n_samples = len(data.images)
d = data.images.reshape((n_samples,-1))
```

In [6]:
```
target = data.target
```

In [7]:
```
# splitting into train and test
X_train,X_test,y_train,y_test = train_test_split(d,target,test_size=0.3)
```

In [8]:
```
# train model
model = svm.SVC(gamma = 0.001)
model.fit(X_train,y_train)
```

Out[8]:
```
▾ SVC
SVC(gamma=0.001)
```

In [9]:
```
y_predict=model.predict(X_test)
```

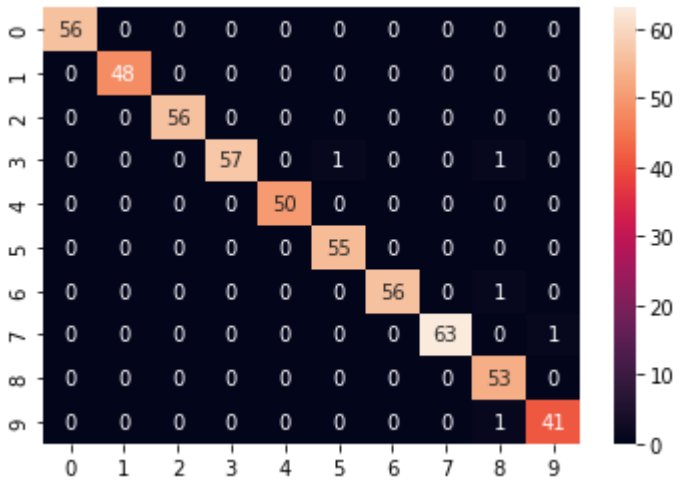
In [10]:
```
# accuracy and F1 score
print(classification_report(y_test,y_predict))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	56
1	1.00	1.00	1.00	48
2	1.00	1.00	1.00	56
3	1.00	0.97	0.98	59
4	1.00	1.00	1.00	50
5	0.98	1.00	0.99	55
6	1.00	0.98	0.99	57
7	1.00	0.98	0.99	64
8	0.95	1.00	0.97	53
9	0.98	0.98	0.98	42
accuracy			0.99	540
macro avg	0.99	0.99	0.99	540
weighted avg	0.99	0.99	0.99	540

In [11]:
```
cf_matrix=confusion_matrix(y_test,y_predict)
```

In [12]:
```
# for confusion matrix
import seaborn as sns
sns.heatmap(cf_matrix, annot=True)
```

Out[12]: <AxesSubplot:>



In [ ]:
```

```