Introduction to Capsule Networks

Keywords: Machine Learning, Deep Learning, Artificial Intelligence, Computational Intelligence, Capsule Networks

— Piotr Chlebek
https://www.linkedin.com/in/piotrr/

Date Jan 15, 2018
ML Gdańsk
http://mlgdansk.pl/

Version 0.1

Before we start… (1)

Credits to

- Dynamic Routing Between Capsules (PDF)
  by **Sara Sabour**, **Nicholas Frosst**, **Geoffrey E Hinton**, Oct 26, 2017
- Capsule Networks (CapsNets) – Tutorial (Video)
  by **Aurélien Géron**, Nov 21, 2017
- What is a CapsNet or Capsule Network? (Article with source code)
  by **Debarko De**, Oct 31, 2017
- Understanding Hinton's Capsule Networks (Article)
  by **Max Pechyonkin**, Nov 3, 2017
- CapsuleNet on MNIST (Notebook with source code)
  by **Kevin Mader**
- ML Gdańsk http://www.mlgdansk.pl/
  - SOM for MNIST (Presentation)
- MNIST http://yann.lecun.com/exdb/mnist/
- This slide background image (source)
- More sources inline

Opinions and views expressed in this presentation are solely my own or quoted. They are not related to any company for which I work / worked.

In Polish: Opinie i poglądy wyrażone w tej prezentacji są wyłącznie moje własne lub cytowane.
Nie są powiązane z żadną firmą, dla której pracuję / pracowałem.

Feedback or questions ?
I encourage You to give me feedback or ask a questions.
Yes, You can interrupt me during the presentation.

Before we start… (2)

Voluntary data collection.

Collected data will be published under a **Public Domain** license.

Ręczność: Prawo [X] / Lewo [ ] | Płeć: Mężczyzna [X] / Kobieta [ ] | Wiek: 48 | Data: 20.06.2017

Styl	8	1	2	3	4	5	6	7	8	9	0	Rezerwa		
Starannie	8	1	2	3	4	5	6	7	8	9	0			
	8	1	2	3	4	5	6	7	8	9	0			
Niedbale	8	1	2	3	4	5	6	7	8	9	0			
	8	1	2	3	4	5	6	7	8	9	0			
Mniejsze	8	1	2	3	4	5	6	7	8	9	0			
Większe	8	1	2	3	4	5	6	7	8	9	0			
Powoli	8	1	2	3	4	5	6	7	8	9	0	6		
Pośpiesznie	8	1	2	3	4	5	6	7	8	9	0			
Pochylone	8	1	2	3	4	5	6	7	8	9	0			
Ciut z boku	8	1	2	3	4	5	6	7	8	9	0			
	8	1	2	3	4	5	6	7	8	9	0			

The Agenda

- Introduction / Theory
  - State of the art results without augmentation
  - Invariance and Equivariance
  - A Single Capsule
  - The Grid of Capsules
  - Squashing
  - Dynamic Routing
  - Routing by Agreement
  - Advantages and Disadvantages
- Case Study
  - Snake Detection
  - Capsules for Face Detection
  - MNIST Dataset
  - Capsules for MNIST by Hinton
- Source Code

State of the art results without augmentation

In the paper 'Dynamic Routing Between Capsules' (Oct 26, 2017)

Table 1: CapsNet classification test accuracy. The MNIST average and standard deviation results are reported from 3 trials.

Method	Routing	Reconstruction	MNIST (%)	MultiMNIST (%)
Baseline	-	-	0.39	8.1
CapsNet	1	no	$0.34_{\pm 0.032}$	-
CapsNet	1	yes	$0.29_{\pm 0.011}$	7.5
CapsNet	3	no	$0.35_{\pm 0.036}$	-
CapsNet	3	yes	$\mathbf{0.25}_{\pm 0.005}$	**5.2**

MNIST - 0.25% test error rate (close to 0.21% state-of-the-art with augmentation)

MultiMNIST - 5.2% test error rate (on-par with the state-of-the-art)

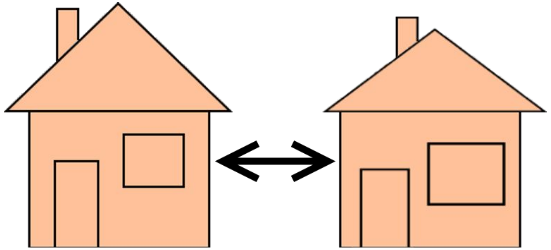
smallNORB - 2.7% test error rate (on-par with the state-of-the-art)

Invariance and Equivariance

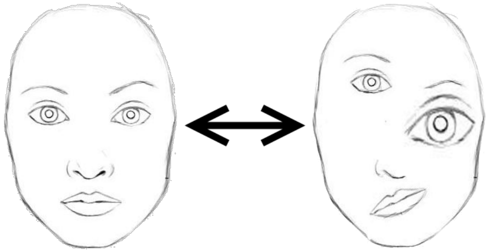
Invariance (PL: niezmienniczość) makes a classifier tolerant to small changes in the viewpoint. The idea of **pooling** is that it creates “summaries” of each sub-region. It also gives you a little bit of positional and translational invariance in object detection. This invariance also leads to triggering **false positive** for images which have the components of a recognized object but not in the correct order.

Equivariance (PL: równowaga) makes a classifier understand the rotation or proportion change and adapt itself accordingly so that the spatial positioning inside an image, including relationships with other components, is not lost.

Text source: [What is a CapsNet or Capsule Network?] with small modifications.



Useful Invariance



Problematic Invariance

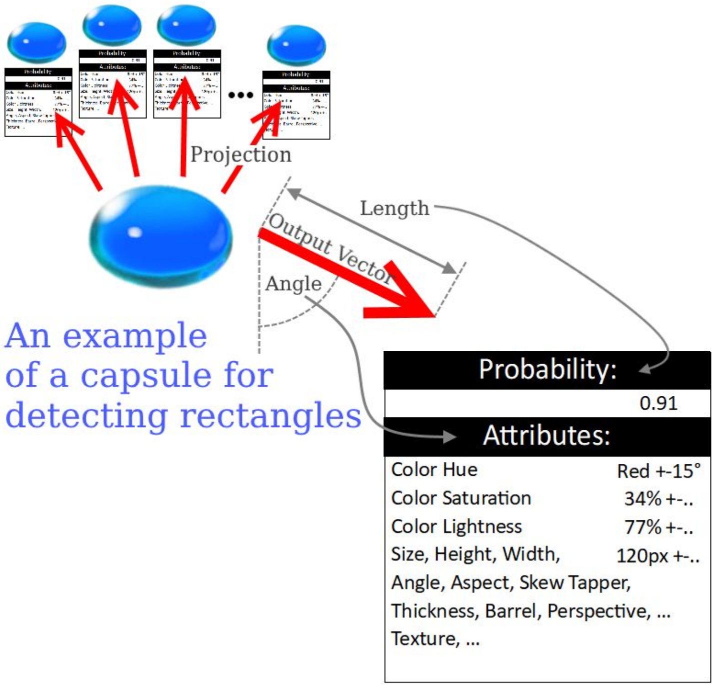
Image source ([original], [deformed])

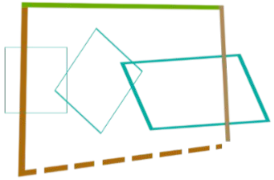
A Single Capsule

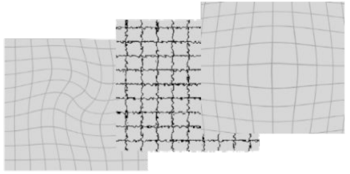
The **output** can be encoded using a **vector** whose **length is a probability**, and the multidimensional **orientation** depends on the **attributes**.

Objects exact **location** can also be an attribute. The approximate location can be deduced from the capsule index.

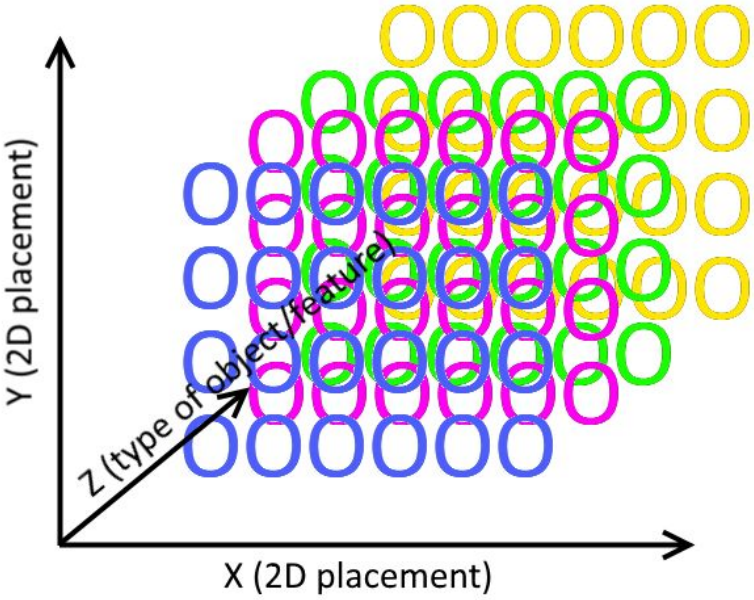
Each capsule **projects** the potential outputs of each connected capsule from the upper layer.







The Grid of Capsules (1)



Many capsules, many types of capsules (which detect a different object or feature).

In the case of image analysis (inverse rendering), each **placement** is a separate capsule.

We do not need another set of capsules for different **orientation**, **size**, **color** or **other attributes** of the detected object/feature. These attributes can be estimated by the capsule itself.

The good news: if several capsules are triggered with high probability, we can ignore the rest of the capsules.

The Grid of Capsules (2)

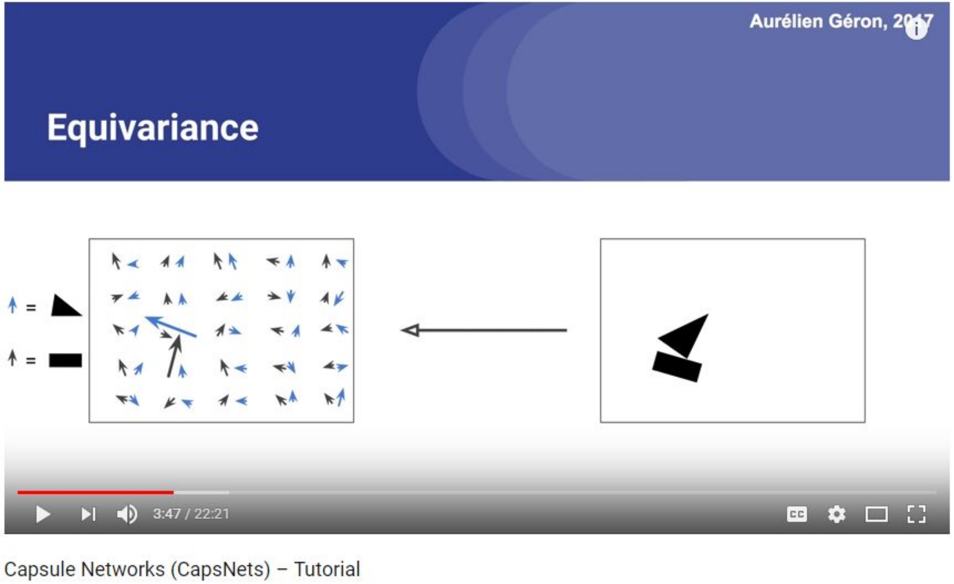
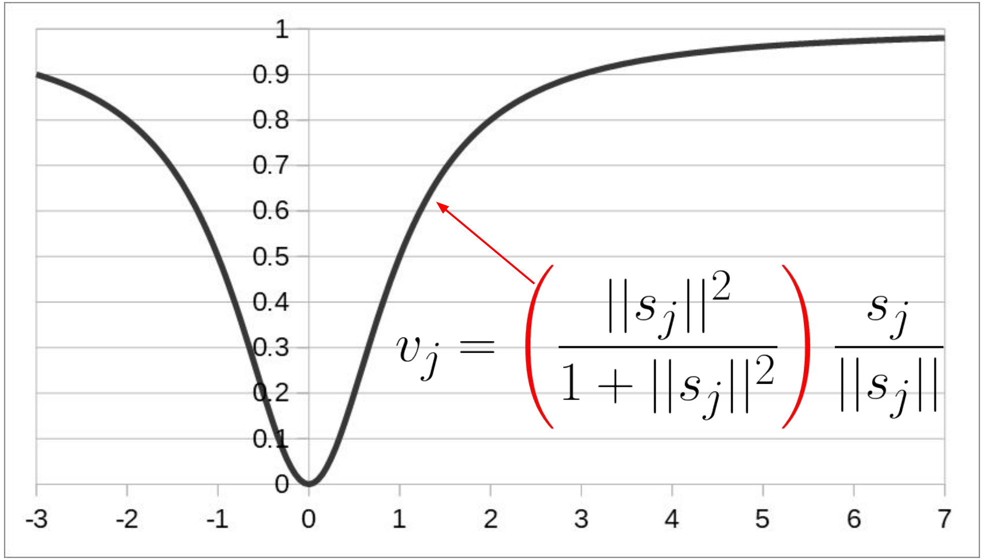


Image source: Capsule Networks (CapsNets) – Tutorial by Aurélien Géron

The **first layer** of capsules outputs for detecting a **triangle** and a **rectangle**.

In this case, **attributes** stores the **angle** of the detected object.

Squashing



Capsule Squashing Function

Dynamic Routing

The algorithm has been presented in the paper 'Dynamic Routing Between Capsules':

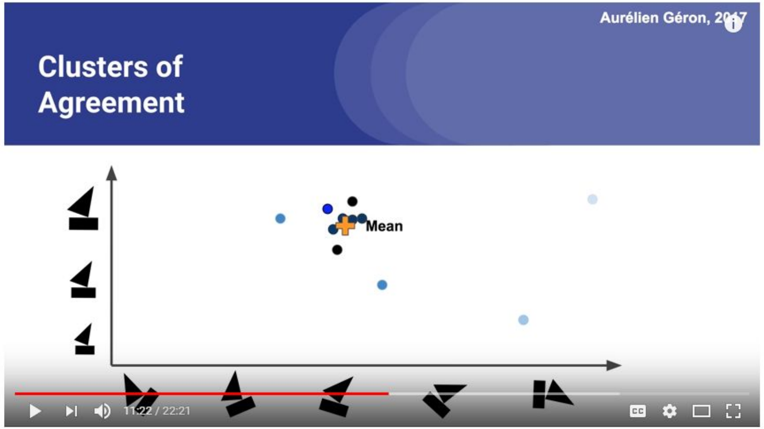
Procedure 1 Routing algorithm.

procedure ROUTING($\hat{\mathbf{u}}_{j|i}$, r, l)
    for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow 0$.
    for r iterations **do**
        for all capsule i in layer l: $\mathbf{c}_i \leftarrow \texttt{softmax}(\mathbf{b}_i)$ ▷ `softmax` computes Eq. 3
        for all capsule j in layer $(l+1)$: $\mathbf{s}_j \leftarrow \sum_i c_{ij} \hat{\mathbf{u}}_{j|i}$
        for all capsule j in layer $(l+1)$: $\mathbf{v}_j \leftarrow \texttt{squash}(\mathbf{s}_j)$ ▷ `squash` computes Eq. 1
        for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i}.\mathbf{v}_j$
    return $\mathbf{v}_j$

Routing by Agreement (1)

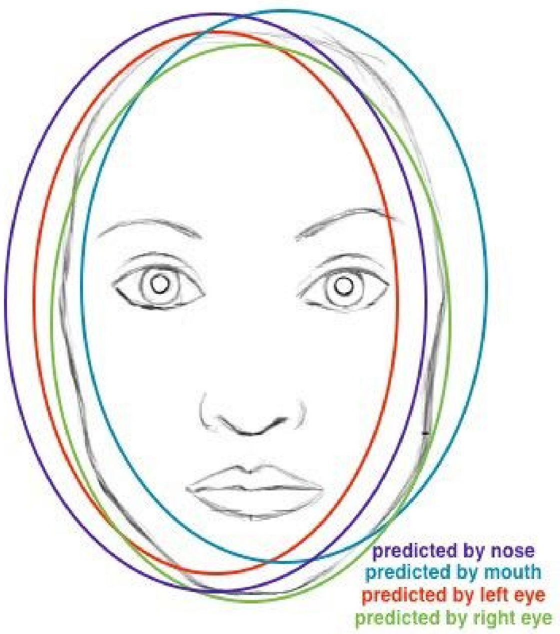
Severar projections of **second layer** capsule attributes for detecting a **boat**.

In this case, **attributes** stores the **boat orientation** (x) and the **boat size** (y).



Capsule Networks (CapsNets) – Tutorial

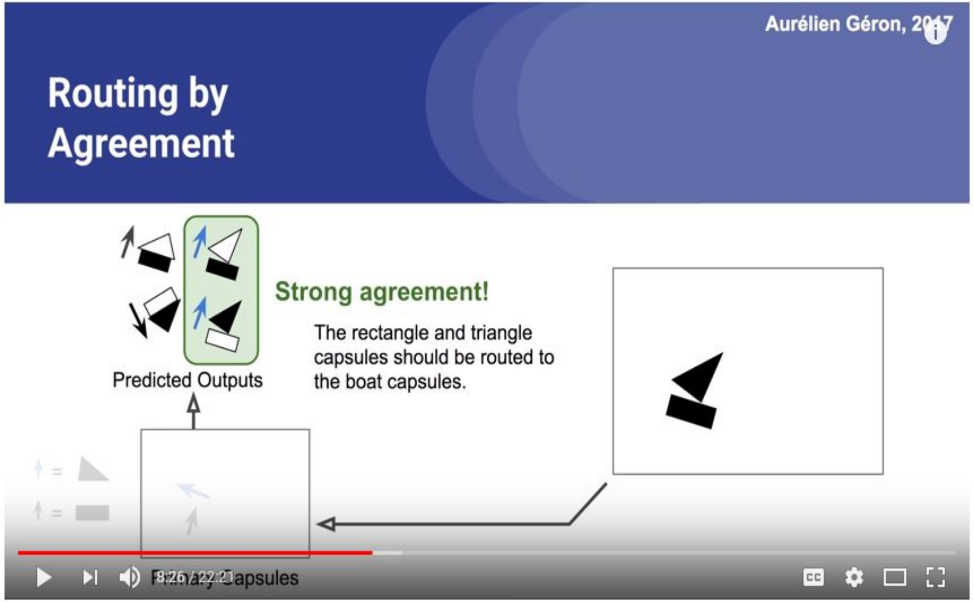
Image source



Prediction for the face location

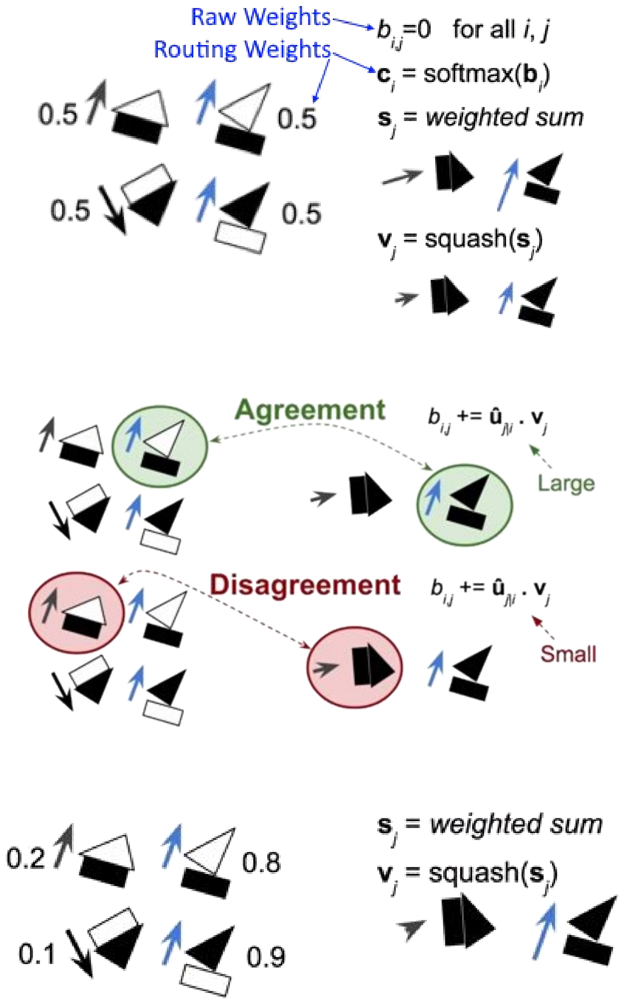
Image source (original, colored)

Routing by Agreement (2)



Capsule Networks (CapsNets) – Tutorial

Images source



Advantages and Disadvantages

+ :)

+ Good preliminary results (MNIST).
+ Requires less training data.
+ Works good with overlapping objects.
+ Potentially good on crowded scene.
+ Can detect partially visible objects.
+ Results are interpretable, components hierarchy can be mapped.
+ Equivariance (classifier adapt to small changes in input).

- :(

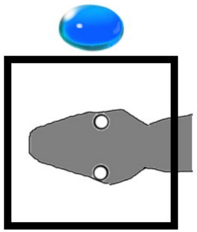
- No known yet accuracy on large images.
- Slow training time (so far).
- Non linear squashing may not reflect the probability nature.
- Cosine of an angle for measuring the agreement may not be the best choice (too flat around 0).

Case Study - Snake Detection

It is easy to imagine a capsule to detect a **snake head** or a **tail**. But these characteristic parts can be hidden and tangled.

Therefore, the capsule for detecting the **snake's body** is important. When such multiple discoveries occur, they should agree on the location of the snake, its color, texture and thickness.

These attributes are taught automatically, detailed labels are not necessary.



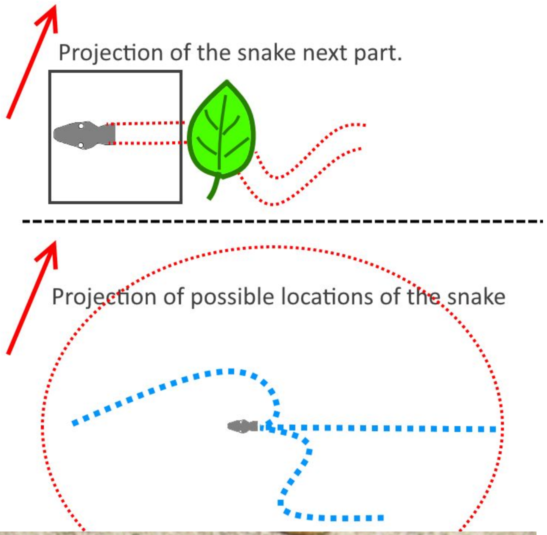


Foto by Glenn Bartolotti - Own work, CC BY-SA 3.0

Capsules for Face Detection (1)

Any ideas which objects or features can be detected?

….

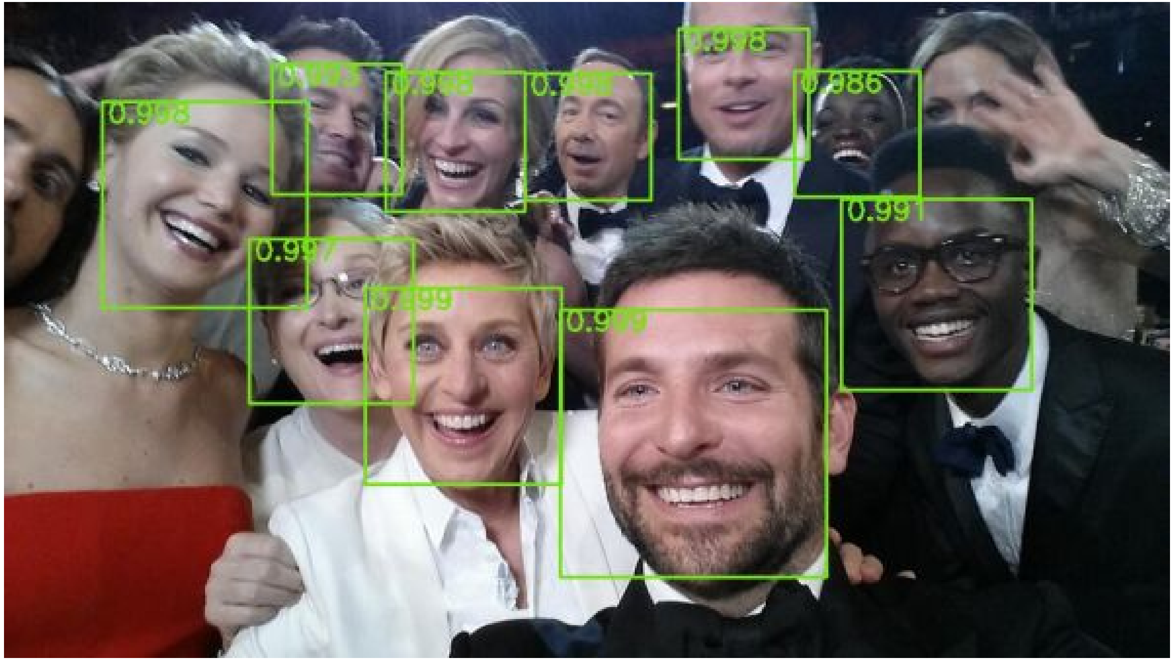


Image [origin](#) (Ellen DeGeneres).

Capsules for Face Detection (2)

Have You considered a partially visible face?

Can we still detect a face?

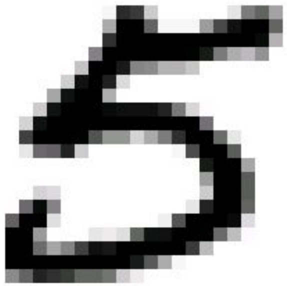
Capsules for Face Detection (3)

Human body:
eye, eyebrows, nose, mouth, face/head shape, chair, skin, cheek, chin, neck, teeth, tongue, beard, mustache, sideburns, ...

Addings:
makeup, glasses, necklace, hat, earrings, (head)scarf, collar, hood, brooch, burqa, mask, clown nose, cigarette, tobacco pipe, face wrap, crown, goggles, bandage, ...

MNIST Dataset

http://yann.lecun.com/exdb/mnist/

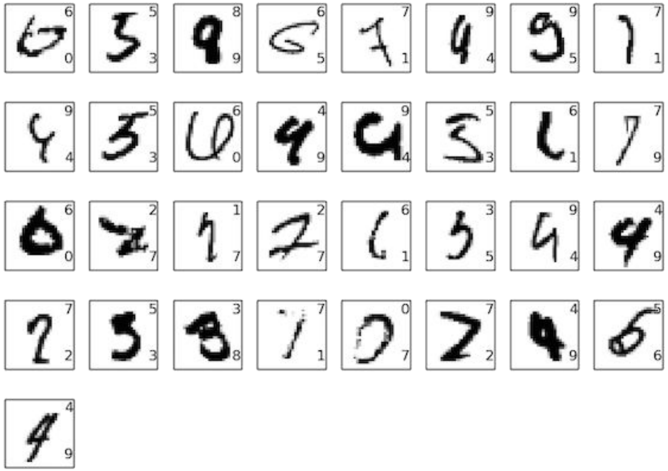


"The MNIST database of handwritten digits, available from this page, has a training set of **60,000** examples, and a test set of **10,000** examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image (**28x28**).

It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting."

Results:

- 0.23 - Ciresan et al. CVPR 2012
- 0.21 - http://cs.nyu.edu/~wanli/dropc/
- …
- 0.33 - http://neuralnetworksanddeeplearning.com/chap6.html ←(img. source)



Capsules for MNIST by Hinton (1)

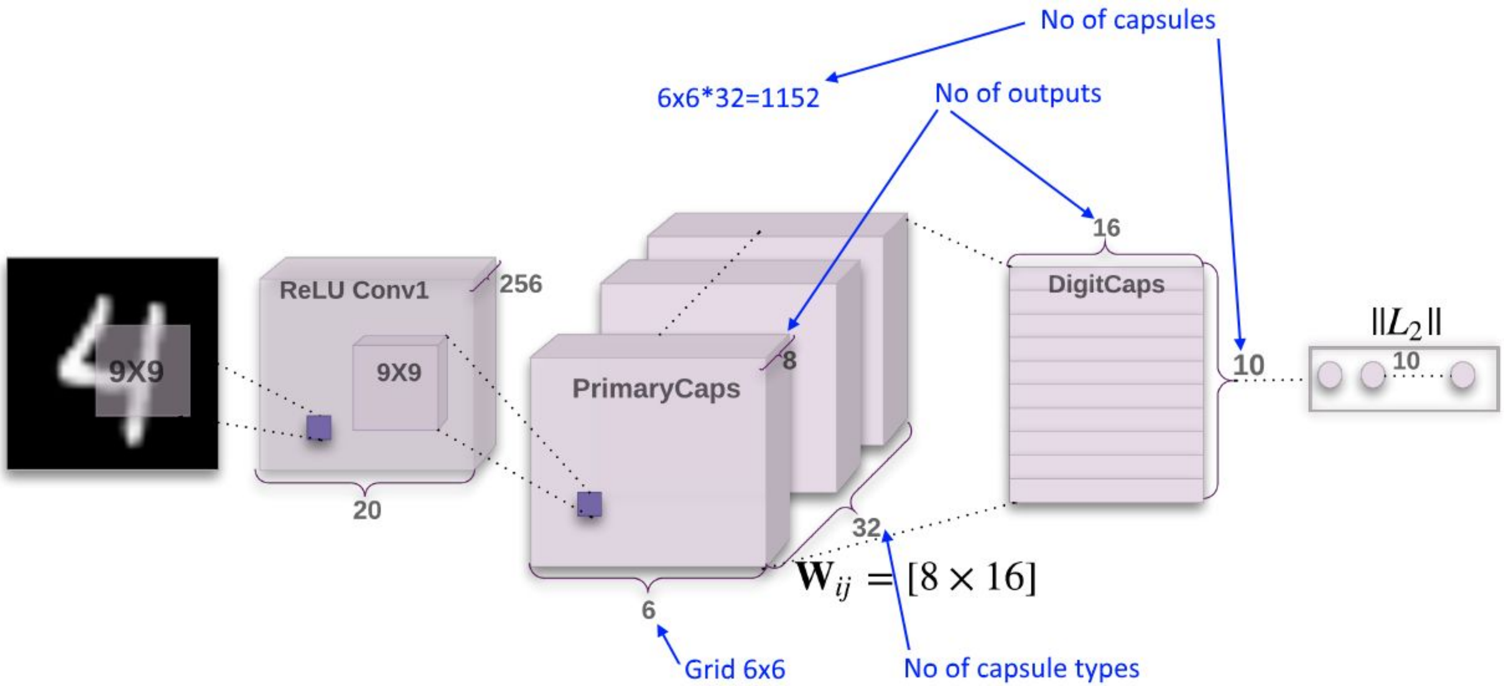


Image source: [Dynamic Routing Between Capsules]

Capsules for MNIST by Hinton (2)

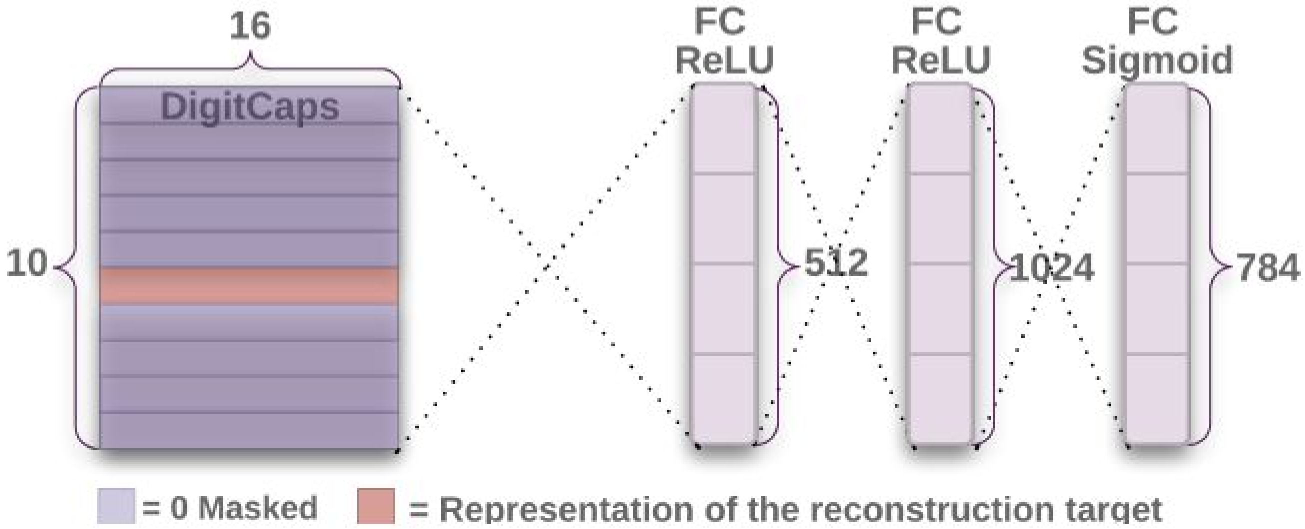
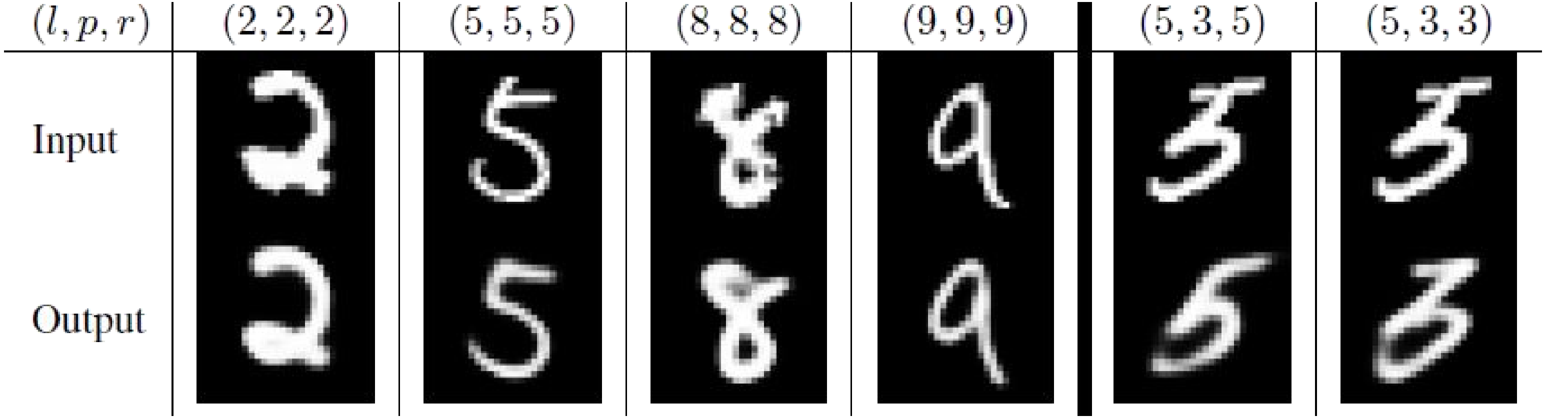


Image source: [Dynamic Routing Between Capsules]

Capsules for MNIST by Hinton (3)

Based on 16D CapsNet output.
(l, p, r) - the label, the prediction, reconstruction target.



Capsules for MNIST by Hinton (4)

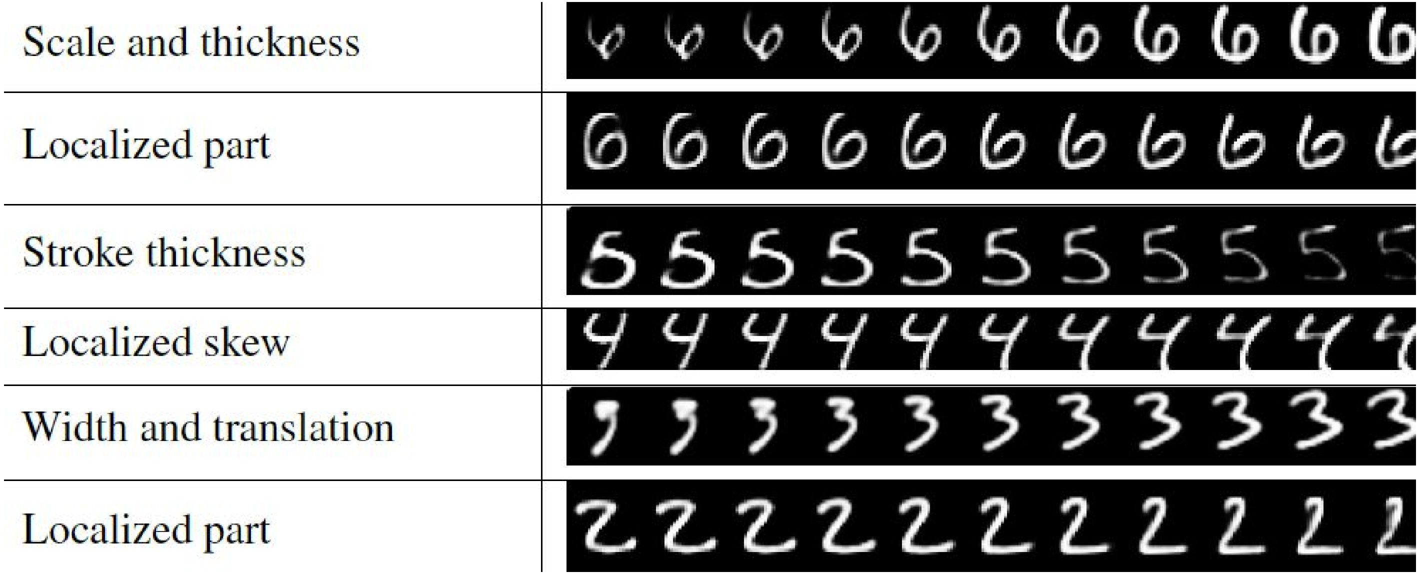


Image source: [Dynamic Routing Between Capsules]

Source Code

https://github.com/naturomics/

https://www.kaggle.com/kmader/capsulenet-on-mnist

A PyTorch implementation of the NIPS 2017 paper "Dynamic Routing Between Capsules".
https://github.com/gram-ai/capsule-networks

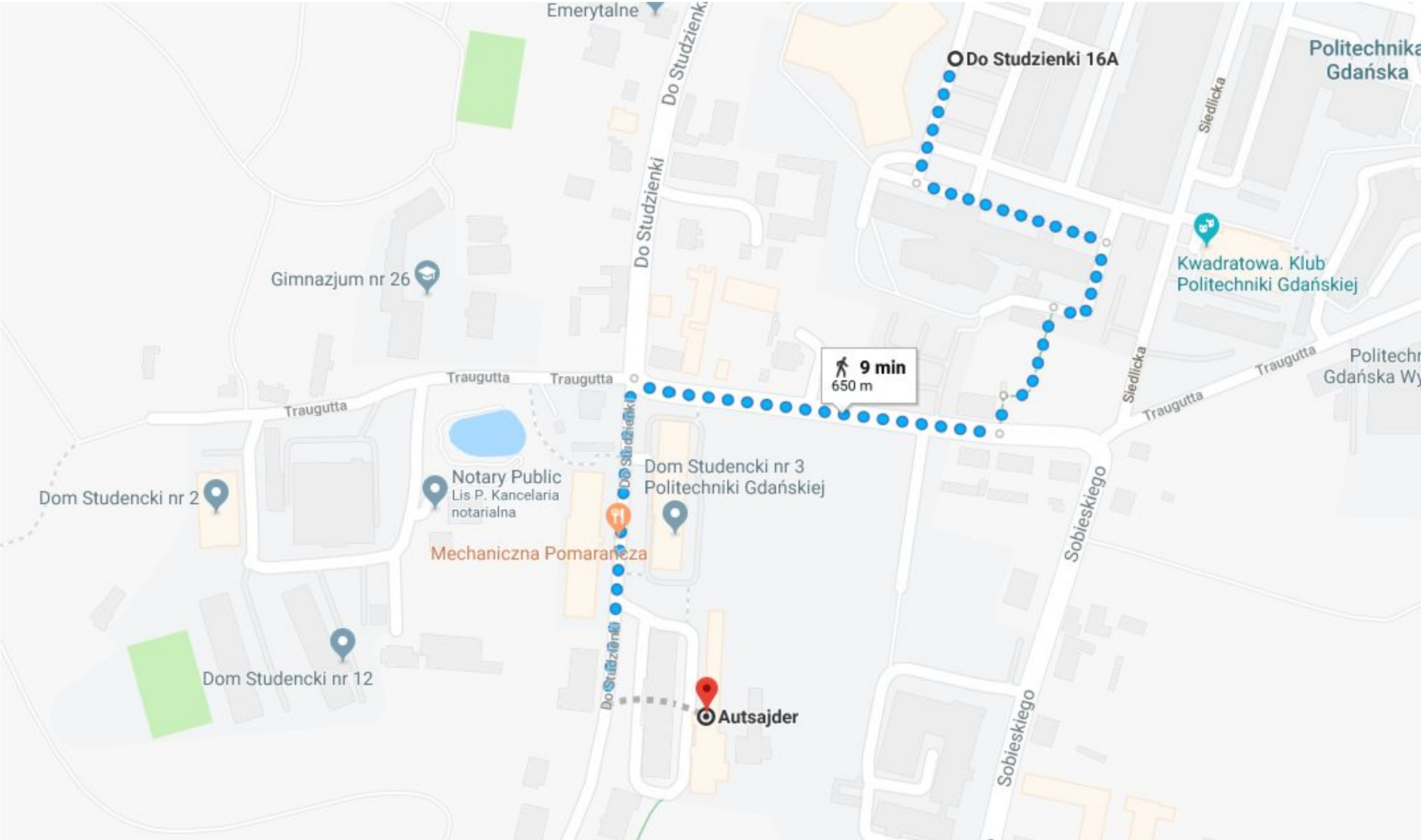
https://github.com/XifengGuo/CapsNet-Keras/blob/master/capsulenet.py

CapsNet implementations:
* TensorFlow implementation: https://github.com/ageron/handson-ml/...
  It is presented in video: https://youtu.be/2Kawrd5szHE
* Keras w/ TensorFlow backend: https://github.com/XifengGuo/CapsNet-...
* TensorFlow: https://github.com/naturomics/CapsNet...
* PyTorch: https://github.com/gram-ai/capsule-ne…

Questions?

Discussion...

;) Classic Routing



BACKUP

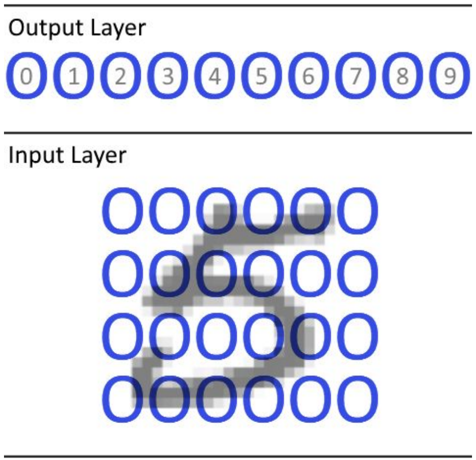
Capsules for MNIST (1) TBD

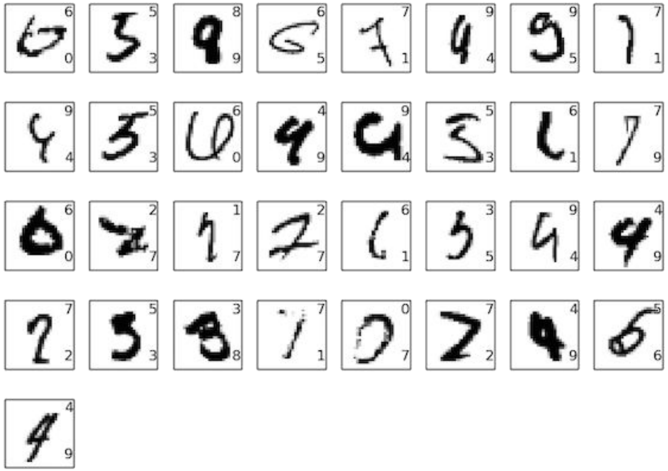
Let's build a **simple** capsule network for MNIST.

1. 2D grid of capsules in first layer.
2. Ten capsules for each digit classification.
3. 2 (or 3) fully connected layers.

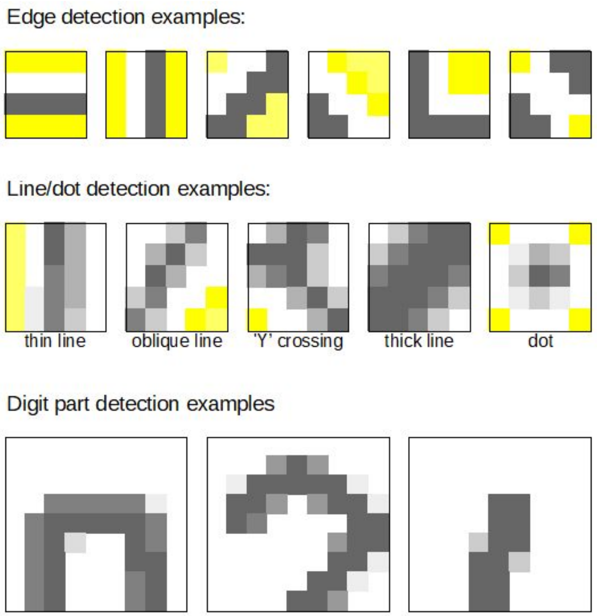
Let's assume:
The projection is orthogonal to capsule index.

We can use capsule index for placement and reuse the projection.





Capsules for MNIST (2) TBD



Types of input capsules (based on pixels):

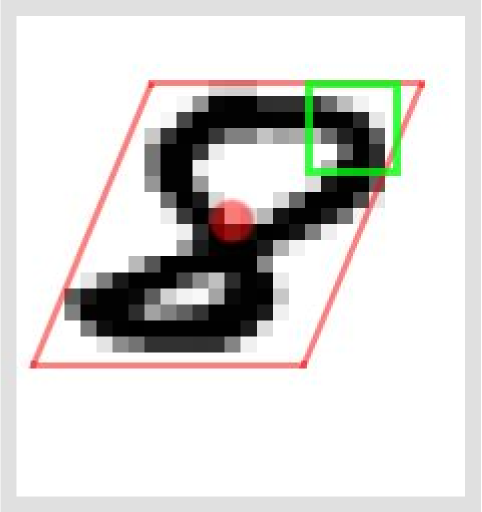
- edge detection + solid kernel (white or black)
- detection of straight line, curved line, crossing, a dot, ...

The more capsule types we have, the more projections we need.
And projections are costly.
Recommendation -> small input kernel and three layers of capsules.

For each digit attribute, input capsules will project (we have two options):

- a probability distribution (better for a small kernel)
- a exact attribute value (better for a large kernel or middle layer)

Capsules for MNIST (3) TBD



Output layer capsule attributes:

- where is a digit center (x,y)
- digit width, height & skew (we ignore rotation or upside-down)
- pen thickness (because it should be agreed)

Input Layer capsule attributes:

- Exact value or distribution of possible digit center (relative)
- Exact value or distribution of possible digit width, height & skew
- Estimation of pen thickness