Using convolutional networks in practice

Raúl Ramos,
Universidad Industrial de Santander

Universidad Industrial de Santander

CAGE

Computo Avanzado y a Gran Escala
Advanced and Large Scale Computing
Research group

outline

1.machine learning

2. convolutional networks

3. working tools

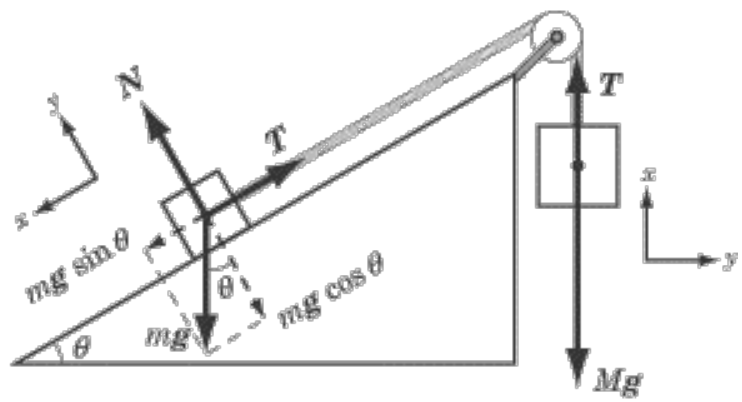
4. CNN training strategies

5. CNN interpretability and customization

6. CNN embeddings and semmatics

7. time series with CNNs

analytic models

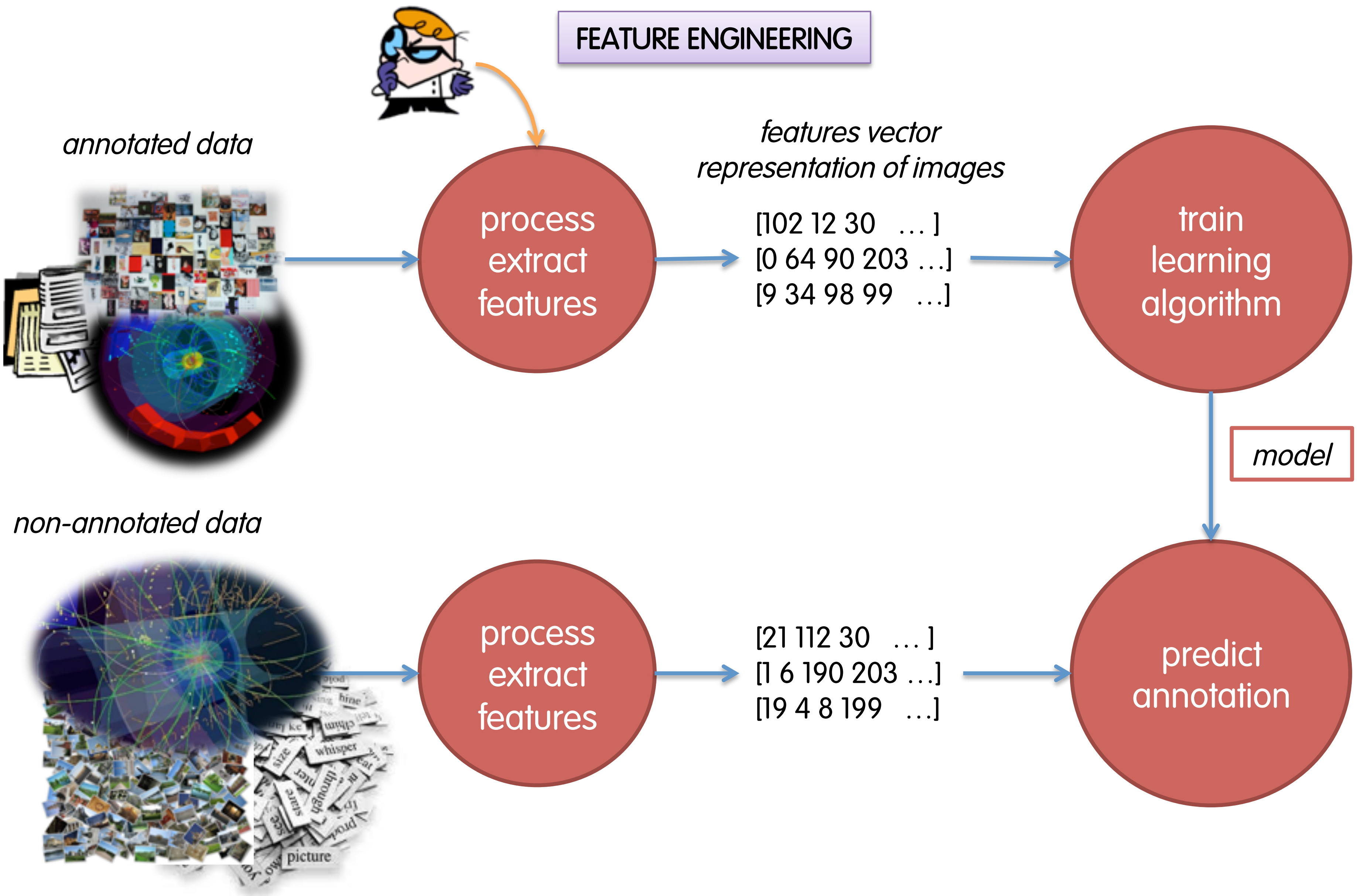


$$\frac{1}{2}Mv^2 + \frac{1}{2}mv^2 - Mgh + mgh\sin\theta = 0$$

$$\frac{1}{2}(m+M)v^2 = gh(M - m\sin\theta)$$

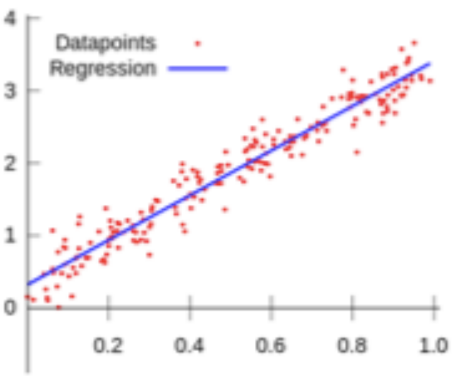
$$v = \sqrt{\frac{2gh(M - m\sin\theta)}{m+M}}$$

ML functional workflow



ML as mathematical optimization

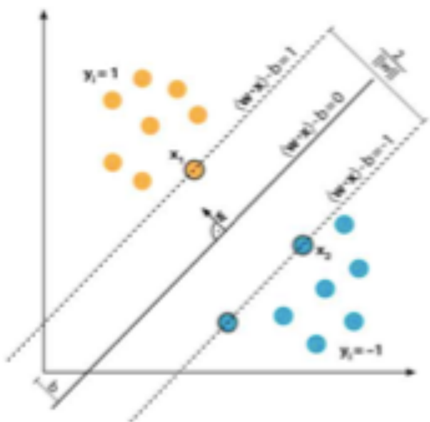
linear regression



$$\underset{\boldsymbol{w}}{arg\,min} \left\| \boldsymbol{y} - \boldsymbol{X}\boldsymbol{w} \right\|^2$$

$$\underset{\boldsymbol{w}}{arg\,min} \left\| \boldsymbol{y} - \boldsymbol{X}\boldsymbol{w} \right\|^2 + \lambda \left\| \boldsymbol{w} \right\|^2$$

SVM



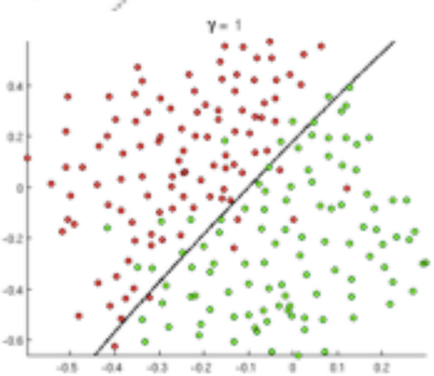
$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j K(x_i, x_j) \alpha_i \alpha_j,$$

subject to:

$$0 \leq \alpha_i \leq C, \quad \text{for } i = 1, 2, \ldots, n,$$

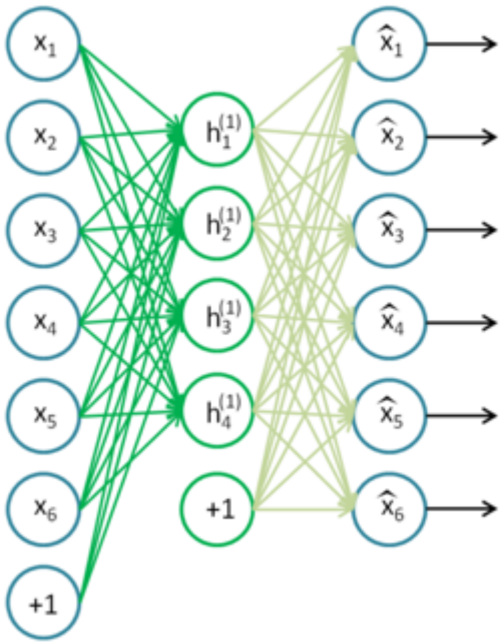
$$\sum_{i=1}^{n} y_i \alpha_i = 0$$

logistic regression



arg min $J(\theta) = -\left[\sum_{i=1}^{m} \sum_{k=1}^{K} 1\left\{ y^{(i)} = k \right\} \log \frac{\exp(\theta^{(k)\top} x^{(i)})}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} x^{(i)})} \right]$

ML as mathematical optimization



$$
\begin{aligned}
J(W,b) &= \left[\frac{1}{m}\sum_{i=1}^{m} J(W,b;x^{(i)},y^{(i)})\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2 \\
&= \left[\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\left\|h_{W,b}(x^{(i)}) - y^{(i)}\right\|^2\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2
\end{aligned}
$$

arg min $\quad J_{\text{sparse}}(W,b) = J(W,b) + \beta\sum_{j=1}^{s_2}\text{KL}(\rho||\hat\rho_j),$

$$\text{KL}(\rho||\hat\rho_j) = \rho\log\frac{\rho}{\hat\rho_j} + (1-\rho)\log\frac{1-\rho}{1-\hat\rho_j}$$

optimization through gradient descent and backpropagation

ML implementation workflow

formulate cost

$$J(W) = \sum (Wx_i - \hat{y}_i)^2$$

add regularization terms

$$J(W) = \sum (Wx_i - \hat{y}_i)^2 + ||W^2||$$

derive gradient expression

$$\frac{\partial J}{\partial W}$$

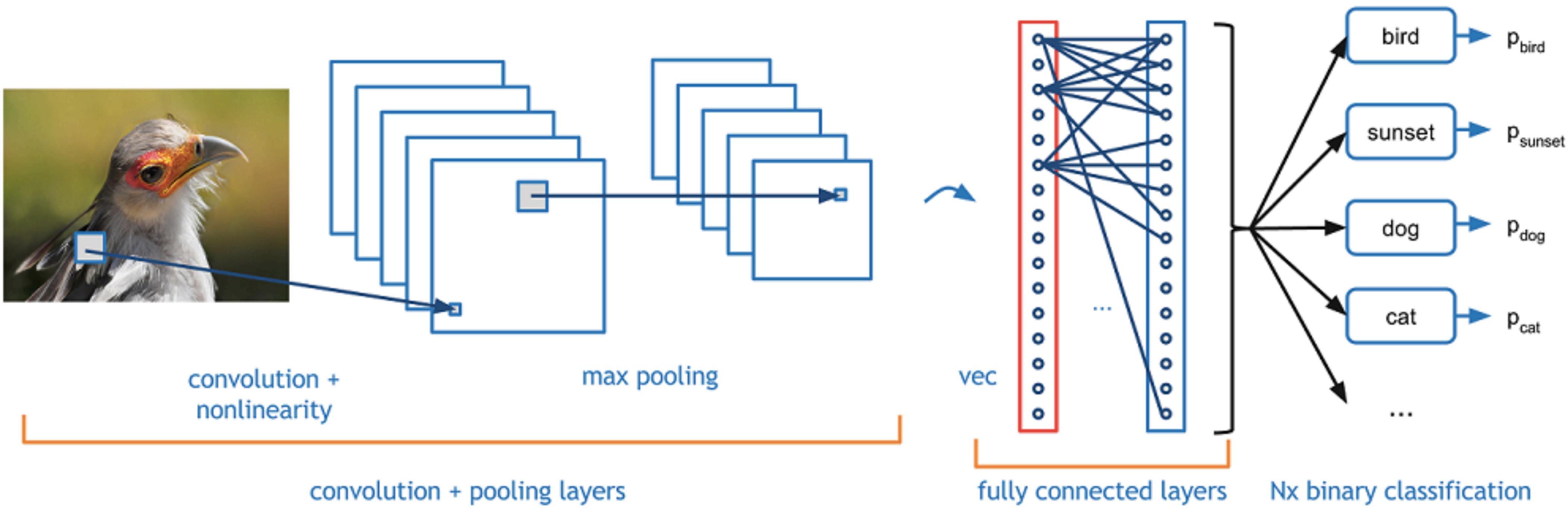
implement

<code>

optimization algorithm

$$\underset{W}{arg\ min} = J(W)$$

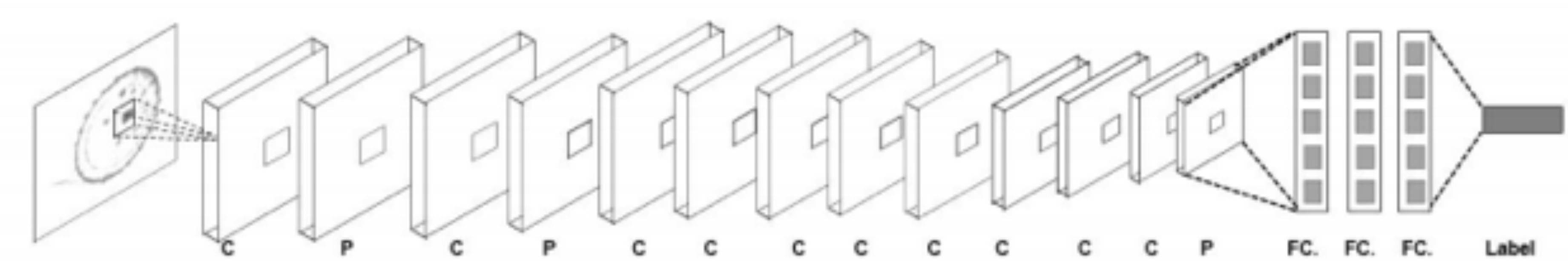
convolutional networks



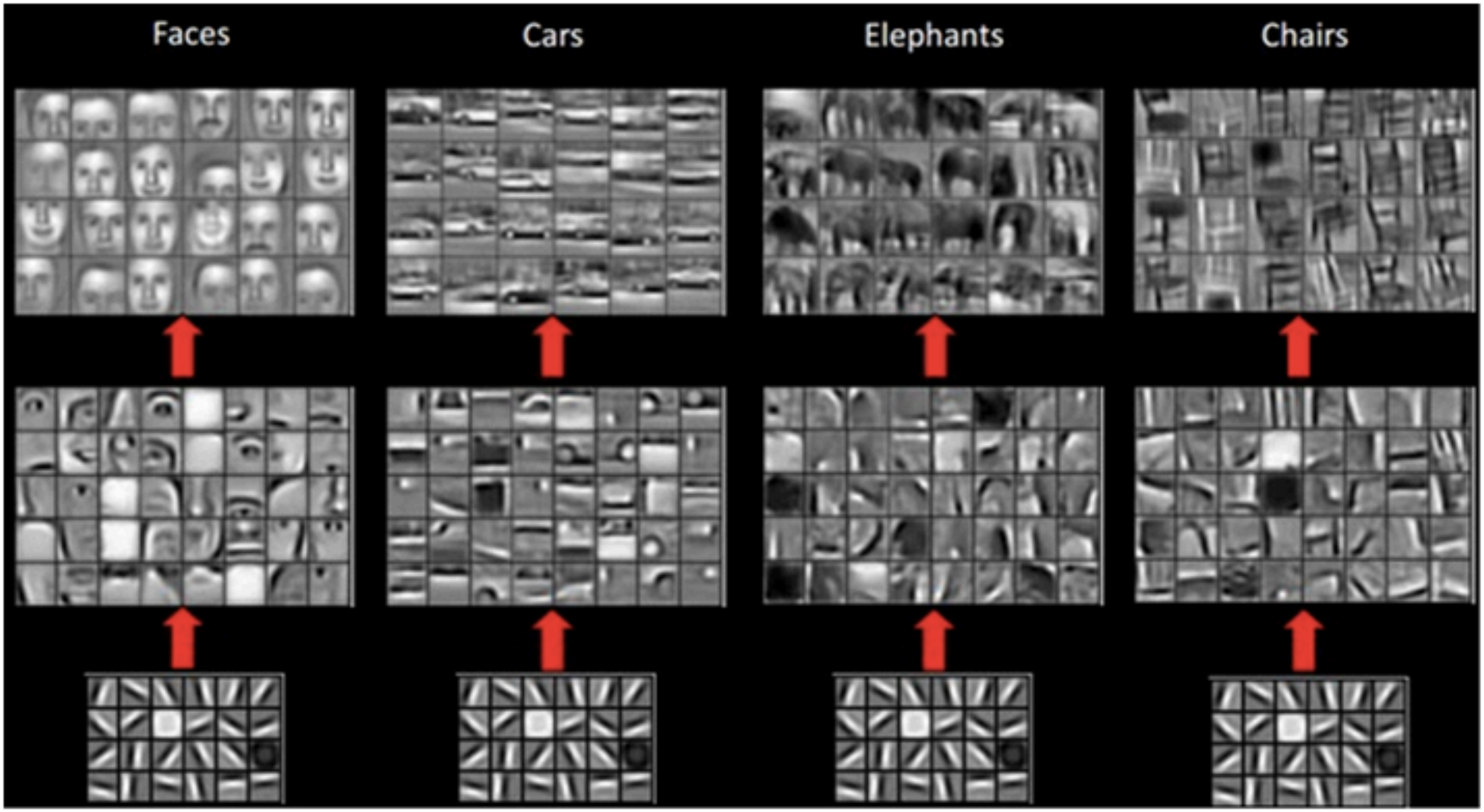
ml4a.github.io/dev/demos/demo_convolution.html

cs231n.github.io/convolutional-networks/

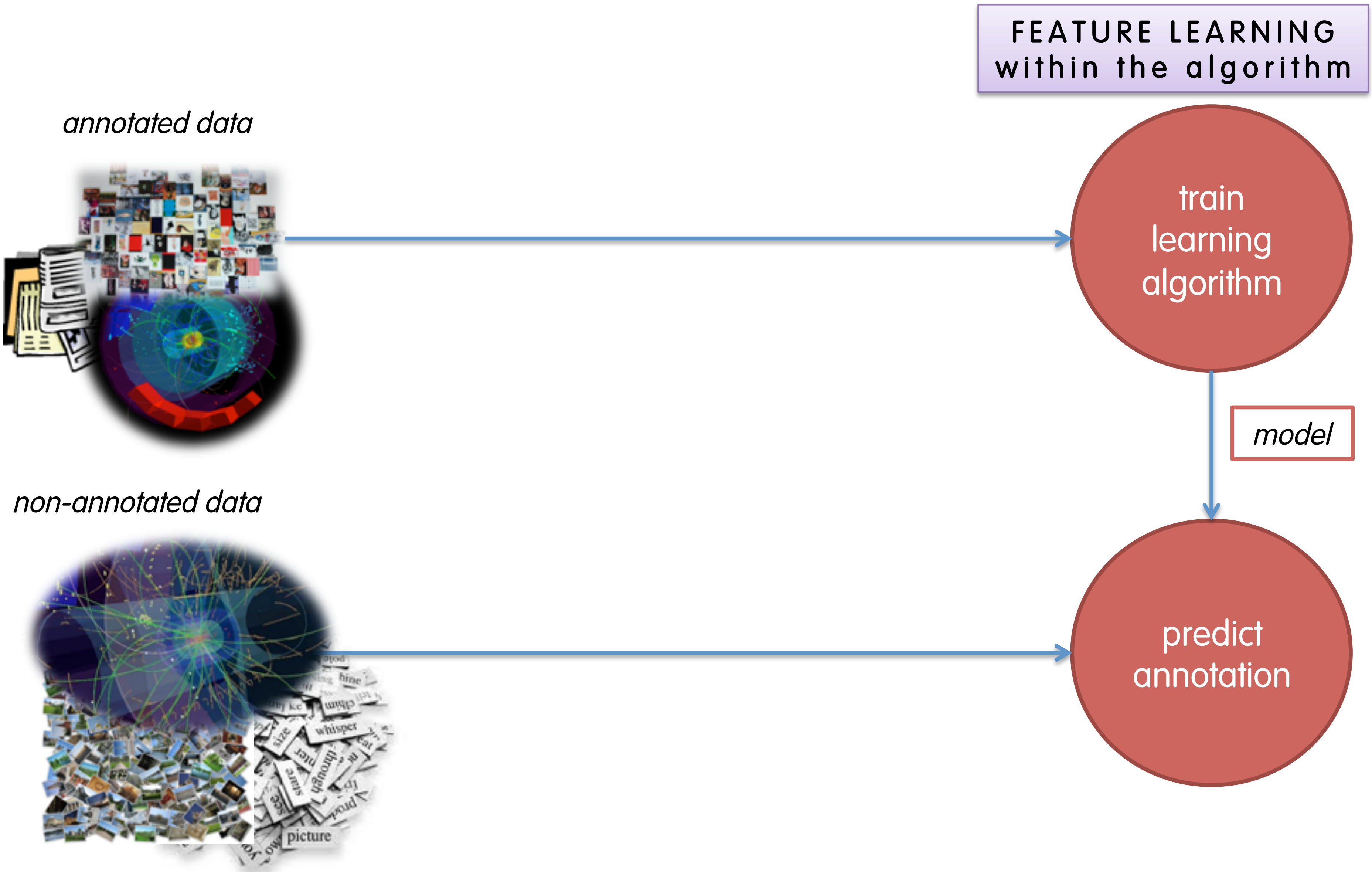
convolutional networks



convolutional networks



CNN's learn features



neural networks evolution

perceptron 1960's
multi layer perceptron 1990's

2005's:

- vanishing gradient

- activation saturation

- GPUs

- training strategies for large dataset (redudant data, gradients):
    - mini-batches
    - stochastic
    - dropouts for regularization
    - gradient momentum
    - weight initialization

- statistical learning approaches

2010: GPUs tehchnologies

deep learning

multi layered structures:

- RBMs, probabilistic → general tasks
- RNN, recurrent → time series
- CNN, convolutional → image recognition
- etc

state of the art

image net LSVRC contests (object localization, detection, etc.)

15M images, 1K categories

state of the art CNNs

image net LSVRC contests (object localization, detection, etc.)
15M images, 1K categories

2015 winner ResNet 152 layers, 1M params

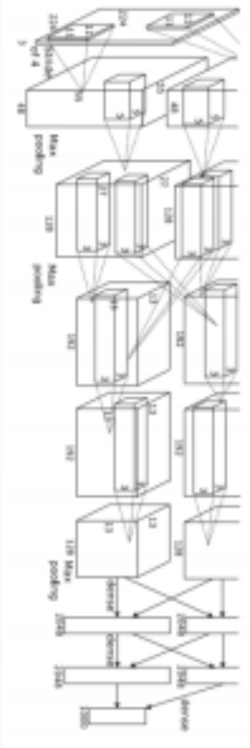
- Uses Residual Units (forces a particular mapping)
- ResNet 1K, 10M params

2017 winner Attention ResNet, 8M params

clarifai.ai

state of the art CNNs

"AlexNet"



[Krizhevsky et al. NIPS 2012]

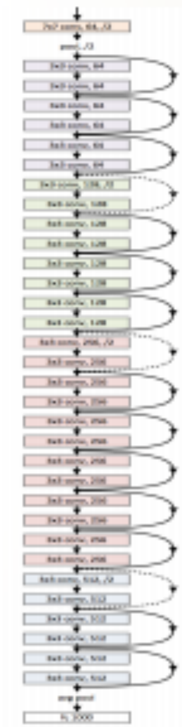
"GoogLeNet"

[Szegedy et al. CVPR 2015]

"VGG Net"

[Simonyan & Zisserman, ICLR 2015]

"ResNet"



[He et al. CVPR 2016]

An Explosion of Datasets

kaggle™

1627 Hosted Datasets	**276** Commercial Competitions	**1919** Student Competitions	**1MM** Data Scientists	**4MM** ML Models Submitted

how CNNs are built and trained

formulate cost

add regularization terms

derive gradient expression

implement

optimization algorithm

$$J(W,b) = \left[\frac{1}{m}\sum_{i=1}^{m} J(W,b;x^{(i)},y^{(i)})\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2$$
$$= \left[\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\left\|h_{W,b}(x^{(i)}) - y^{(i)}\right\|^2\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2$$

how CNNs are built and trained

formulate cost

↓

add regularization terms

↓

derive gradient expression

↓

implement

↓

optimization algorithm

TensorFlow

theano

torch

Theano

this is symbolic!!!!!

```
>>> X = theano.matrix()
>>> cost = (T.dot(X,w))^2 - y + lambda *  w^2
>>> gradient = T.grad(cost)
>>> fgrad = theano.function ([X,y,w], gradient)
>>> optimize (fgrad)
```

produces native code according to conf

~/.theanorc

```
[global]
device = cpu
floatX = float32
```

~/.theanorc

```
[global]
device = cuda
floatX = float32
```

MUST KNOW WHAT MEMORY

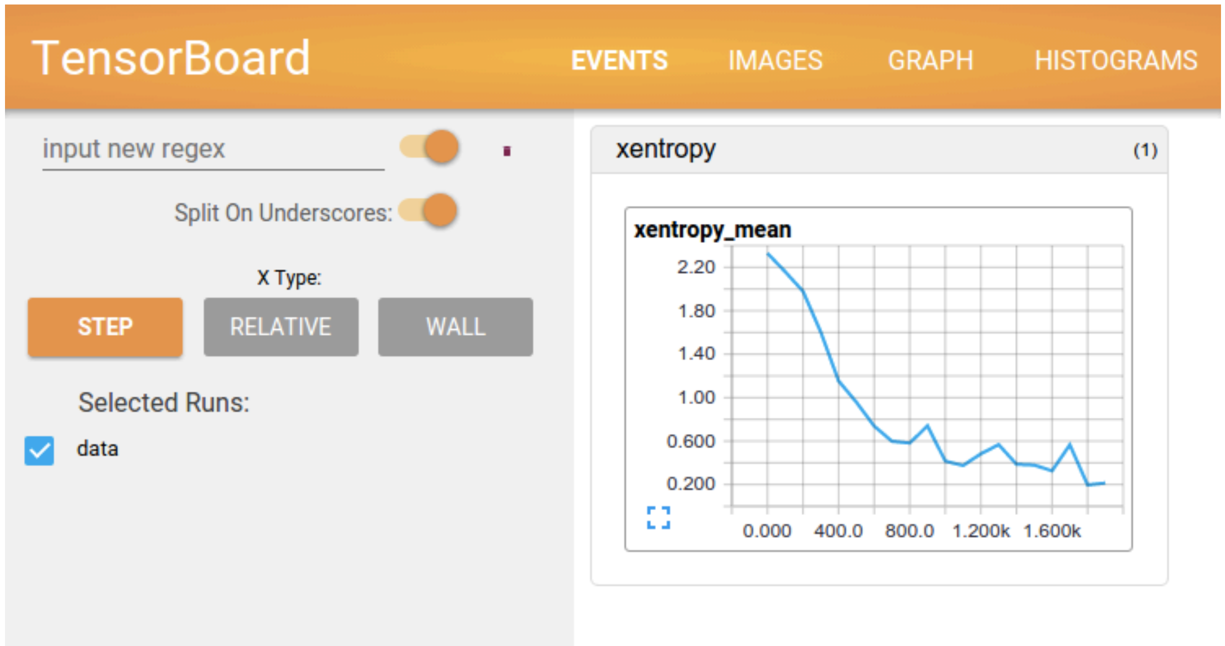
OUR DATA USES

Tensor-flow high level API

```
# Convolution Layer with 32 filters and a kernel size of 5
conv1 = tf.layers.conv2d(x, 32, 5, activation=tf.nn.relu)
# Max Pooling (down-sampling) with strides of 2 and kernel size of 2
conv1 = tf.layers.max_pooling2d(conv1, 2, 2)

# Convolution Layer with 64 filters and a kernel size of 3
conv2 = tf.layers.conv2d(conv1, 64, 3, activation=tf.nn.relu)
# Max Pooling (down-sampling) with strides of 2 and kernel size of 2
conv2 = tf.layers.max_pooling2d(conv2, 2, 2)

# Flatten the data to a 1-D vector for the fully connected layer
fc1 = tf.contrib.layers.flatten(conv2)

# Fully connected layer (in tf contrib folder for now)
fc1 = tf.layers.dense(fc1, 1024)
# Apply Dropout (if is_training is False, dropout is not applied)
fc1 = tf.layers.dropout(fc1, rate=dropout, training=is_training)

# Output layer, class prediction
out = tf.layers.dense(fc1, n_classes)
```

Tensor-board

optimization evolution



computation graph visualization

https://www.tensorflow.org/versions/r0.12/how_tos/graph_viz/

research on CNNs for the mortals

training with small datasets

feature learning

interpretability

customization

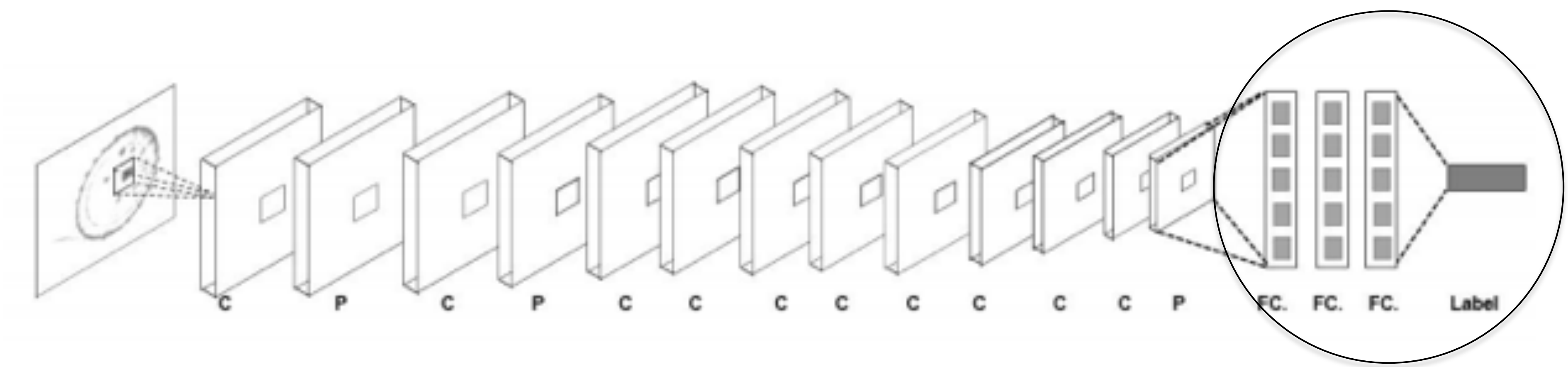
semantic embeddings

time series analysis

training with small datasets

finetuning:

- take CNN trained with large dataset
- reconfigure last layers for your class set
- adapt your data (resolution, include transforms, etc.)
- train with your data

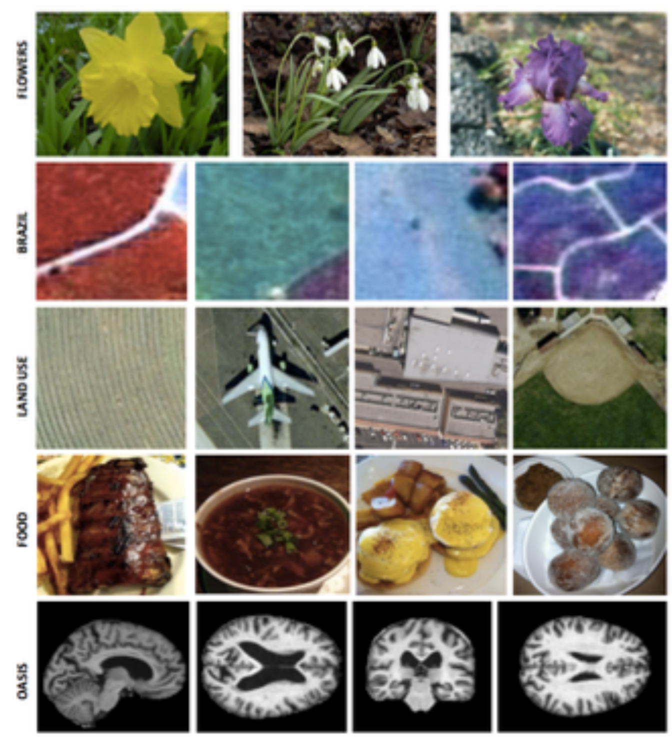


training with small datasets

issues with finetuning:

- your dataset might be too different from pretrain
- features learnt on pretrained models

Dataset	number of images / classes	images per class	avg image size	Literature
Brazil	2876 / 2	1400	64x64	87.03% [17]
Flowers	8187 / 102	40-258	750x500	82.50% [16]
Land Use	2100 / 21	100	256x256	93.42% [21]
OASIS	9600 / 2	4800	176x176	80.26% [10]
Food	10287* /101	100	512x512	56.40%[3]*



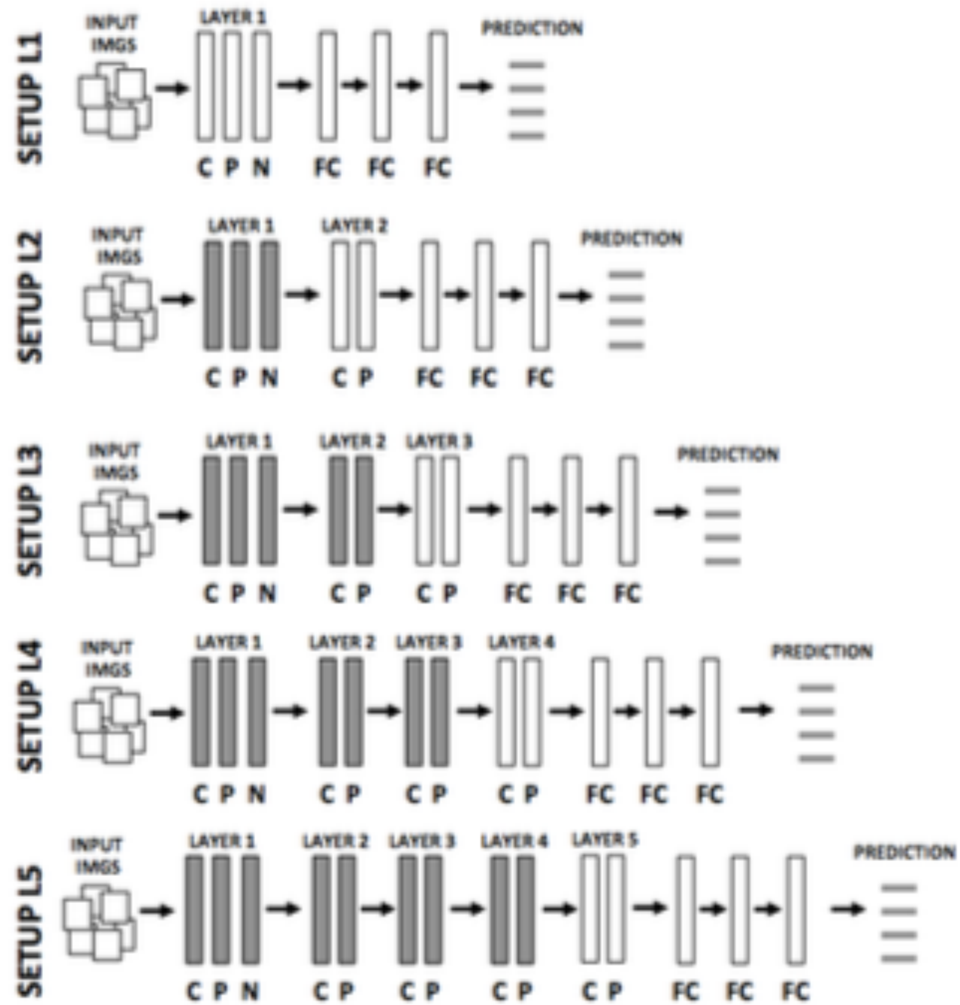
training with small datasets

CNN architecture

Layer	Input	Filter Stride Pad	Output	Connections × neurons	Total connections
conv1 rn	227x227x3	11x11/4/0	55x55	363x96	34,848
pool1	55x55x96	3x3/2/0	27x27		
conv2 rn	27x27x96	5x5/1/2	27x27	2400x256	614,400
pool2	27x27x256	3x3/2/0	13x13		
conv3 r	13x13x256	3x3/1/1	13x13	2304x384	884,736
conv4 r	13x13x384	3x3/1/1	13x13	3456x384	1,327,104
conv5 r	13x13x384	3x3/1/1	13x13	3456x256	884,736
pool5	13x13x256	3x3/2/0	6x6		
FC1 rd	6x6x256		4096	9216	37,748,736
FC2 r	1x1x4096		4096	4096	16,777,216
FC3	1x1x4096		4096	102	417,792
TOTALS					**58,689,568**

Table 2

greedy layerwise training

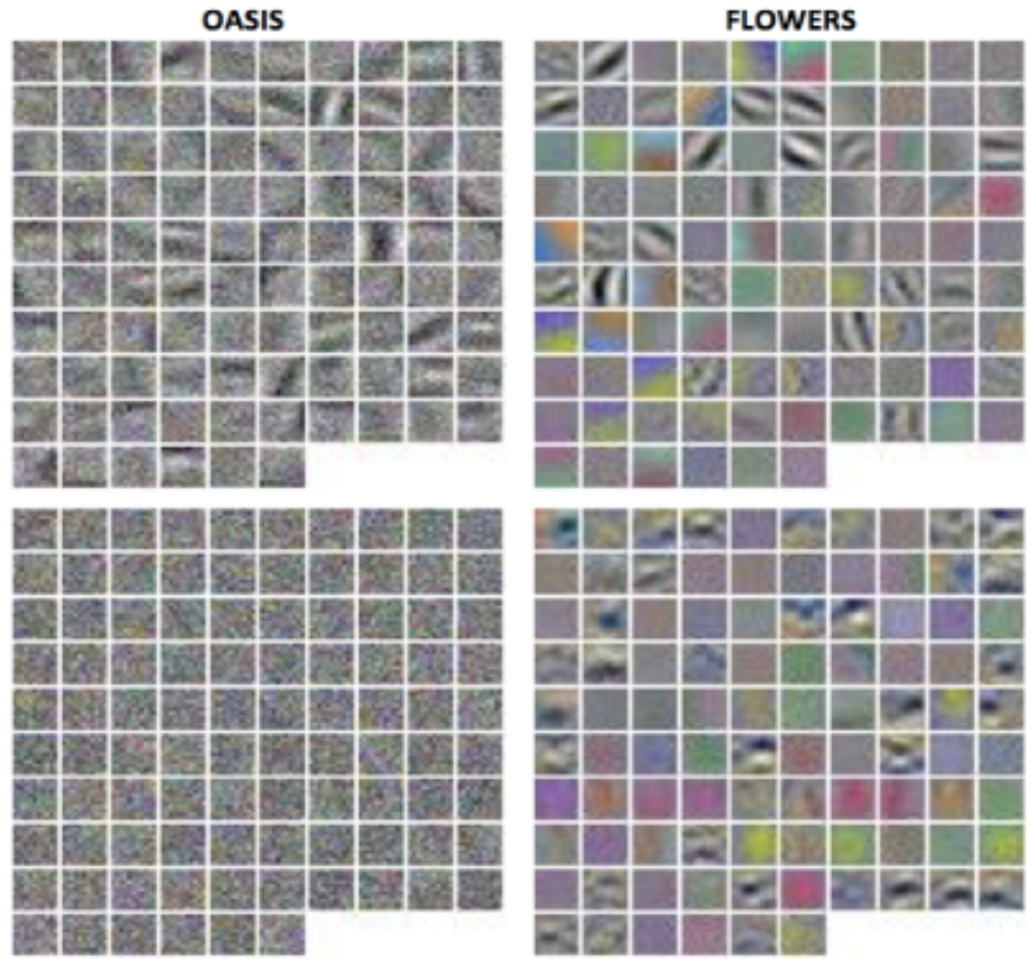


training with small datasets

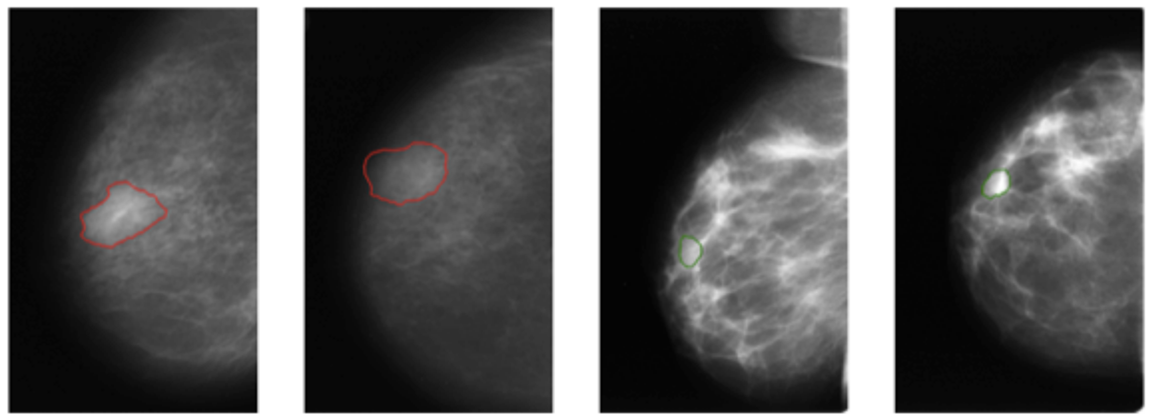
results

	Brazil	Flowers	Land Use	OASIS	Food
AlexNet.Simple Architecture					
L1	84.6%	49.6%	68.0%	65.3%	10.3%
L2	89.5%	65.9%	84.1%	67.9%	16.0%
L3	90.8%	75.1%	**87.5**%	**69.8**%	21.2%
L4	**91.7**%	**79.7**%	84.6%	68.3%	**22.8**%
L5	91.3%	79.5%	81.6%	68.8%	20.6%
Full L3		61.3%	79.6%		15.2%
Full L5	81.8%	<u>3.51%</u>	<u>4.5%</u>	68.5%	<u>1.3%</u>
	+12.0%	+30.4%	+9.9%	+10.27%	+49.8%
AlexNet Architecture					
L1	79.5%	57.9%	74.1%	64.5%	12.3%
L2	89.7%	72.1%	84.7%	68.5%	16.5%
L3	90.6%	76.5%	87.9%	69.1%	19.4%
L4	91.3%	77.9%	**88.3**%	**69.6**%	20.8%
L5	**91.9**%	**80.5**%	87.9%	68.9%	**23.3**%
Full L5	91.2%	70.6%	55.8%	67.5%	12.1%
	+0.8%	+14.0%	+57.7%	+3.1%	+92.4%

first layer features



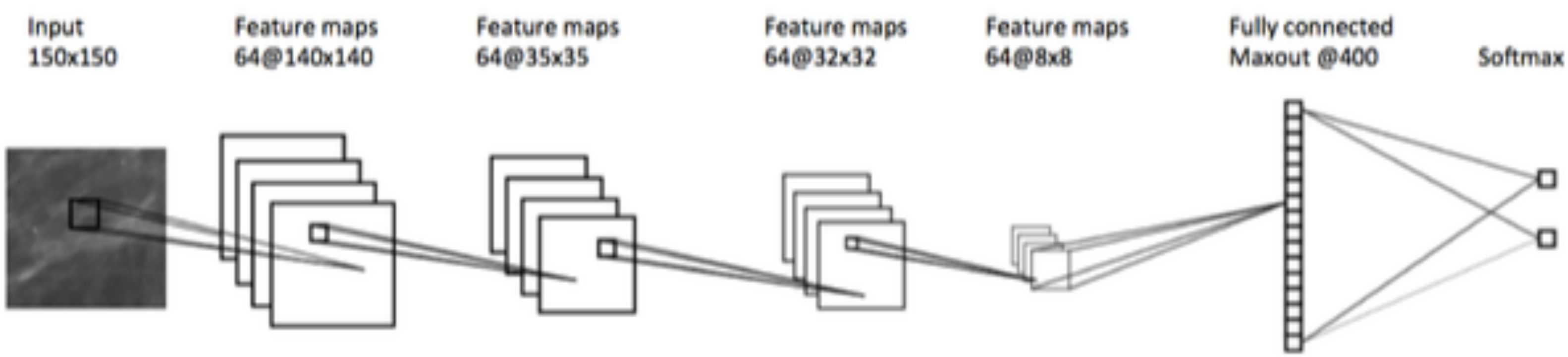
training with small datasets



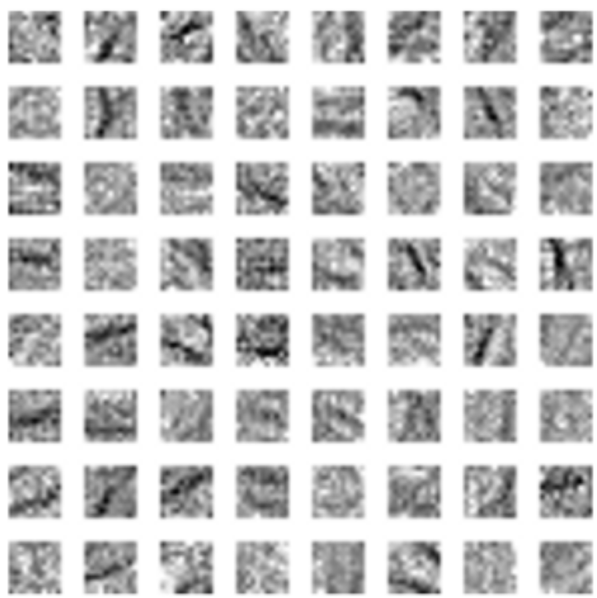
(a) (b) (c) (d)

feature learning:

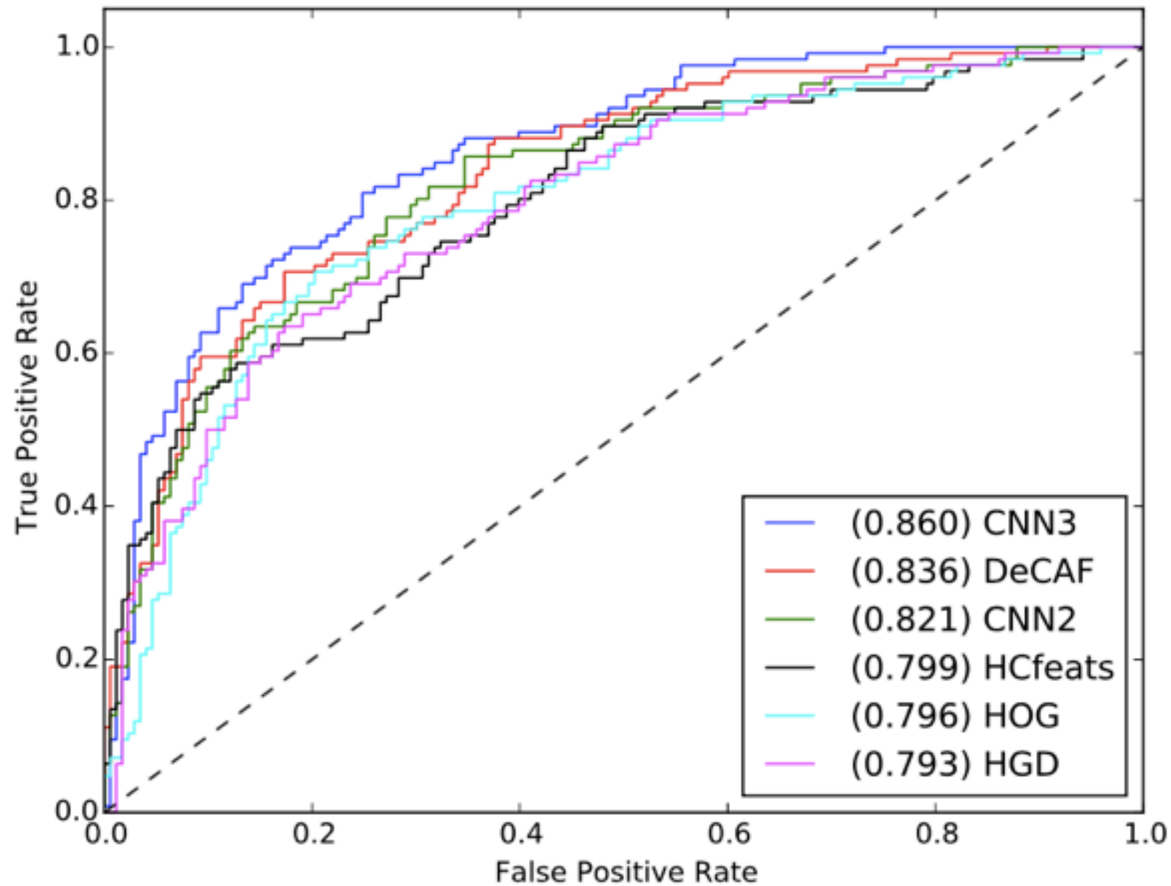
- explore different CNN architectures
- use CNN activations as features in a traditional ML setup
- compare performance with ENGINEERED features



training with small datasets



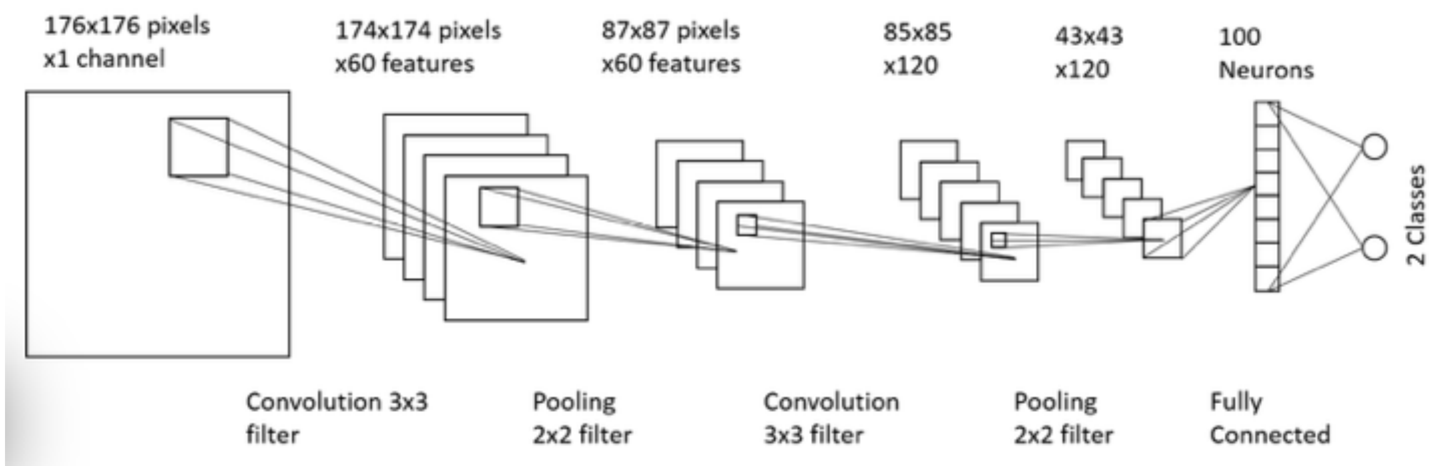
first layer filters



DeCAF: pretrained with Imagenet
HCFeats: 17 shape, texture and statistical features
HOG: histogram of oriented gradients
HGD: histogram of gradient divergence

interpretability

no black box in many domains. CNN for alzheimers' detection



	Matrix (std-dev)				CAFFE (std-dev)				Torch (std-dev)			
	acc	auc	tpr	fpr	acc	auc	tpr	fpr	acc	auc	tpr	fpr
Full	0.725 (0.093)	0.875 (0.040)	0.800 (0.112)	0.400 (0.335)	**0.825** (0.061)	0.875 (0.119)	**0.950** (0.112)	0.300 (0.112)	**0.825** (0.061)	**0.900** (0.085)	**0.950** (0.112)	0.300 (0.112)
LSC	0.750 (0.125)	0.838 (0.106)	0.800 (0.112)	0.250 (0.306)	0.725 (0.056)	**0.887** (0.073)	0.600 (0.285)	**0.150** (0.224)	**0.875** (0.088)	0.881 (0.072)	**0.900** (0.137)	**0.150** (0.224)
LA	0.800 (0.143)	0.856 (0.134)	0.900 (0.137)	0.300 (0.209)	**0.850** (0.137)	**0.950** (0.047)	0.900 (0.223)	0.300 (0.209)	**0.850** (0.104)	0.831 (0.160)	**0.950** (0.112)	**0.250** (0.250)
LT	0.725 (0.104)	0.663 (0.116)	0.850 (0.137)	0.400 (0.224)	0.775 (0.105)	0.806 (0.137)	**0.950** (0.112)	0.400 (0.224)	**0.800** (0.112)	**0.831** (0.067)	0.900 (0.137)	**0.300** (0.274)

inspect filter activation difference

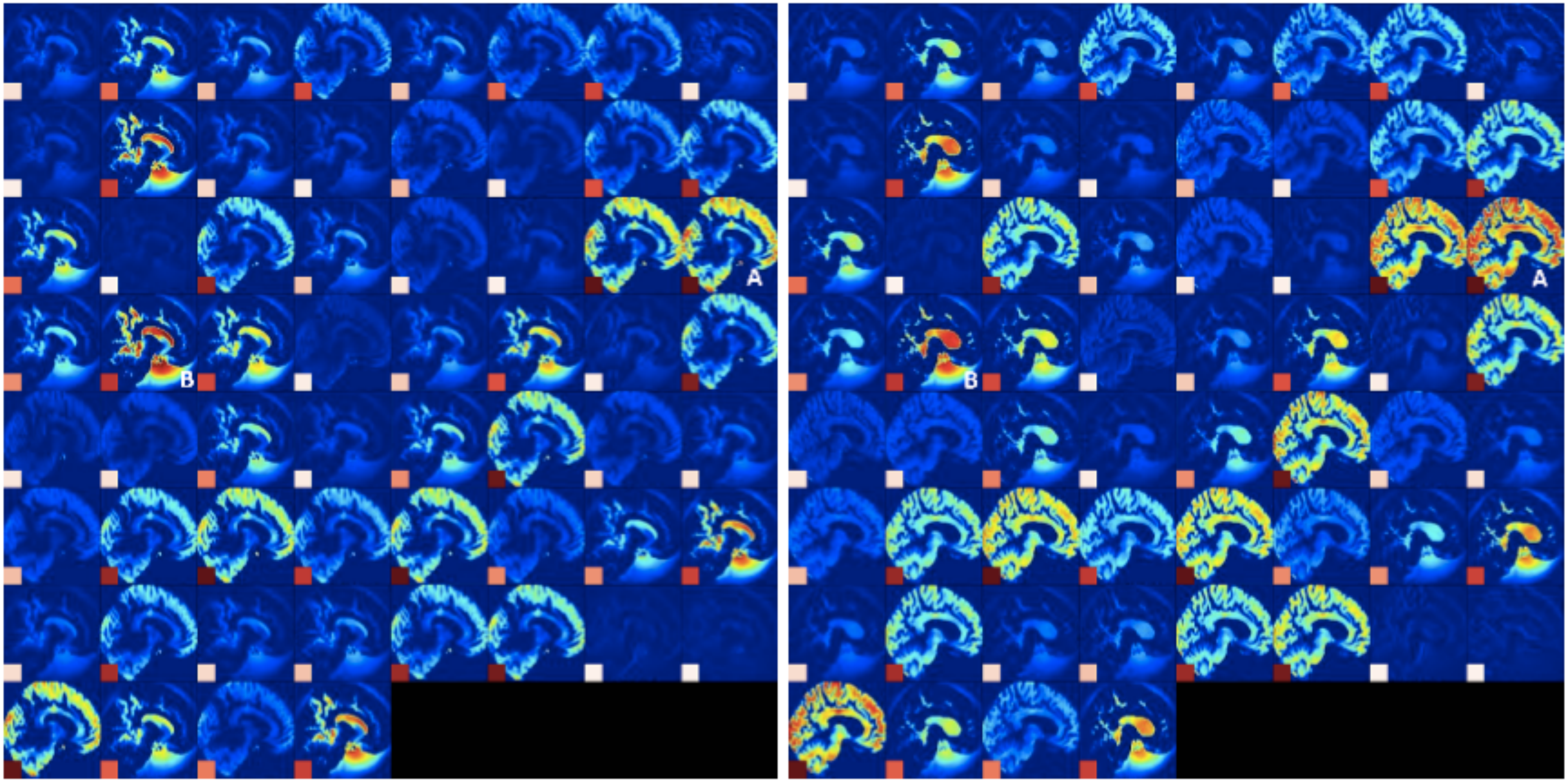


Figure 2. Activations of the 60 first layer filters of the selected CNN for a sagital image cut of a CDR 0 patient (left) and the same cut of a CDR1 patient (right). Colored squares show the dBMS of each filter (thus, a global measure, not for this particular image cut). Dark red represents higher dBMS (higher discrimination between both classes), while light pink represents low dBMS.

build a brain model

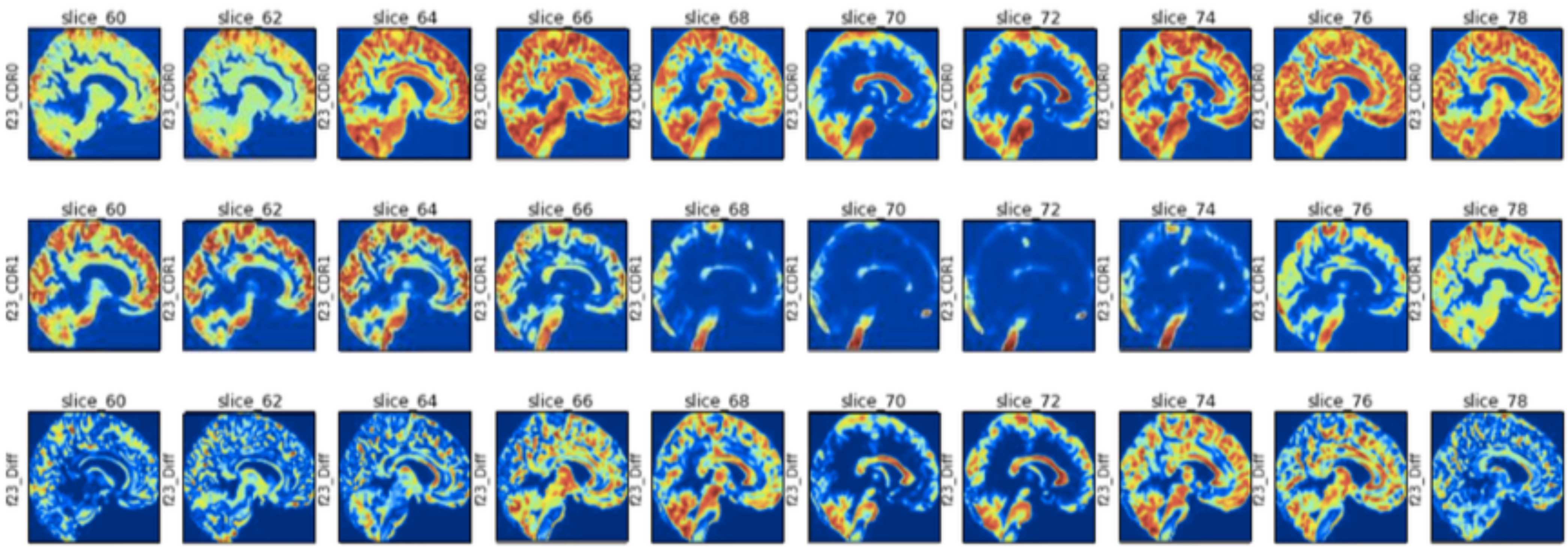


Figure 3. Sagital cuts 60 to 68 for brain models of filter A for CDR 0 (top), CDR 1 (middle) and the differential model (bottom). Only even cuts are included to show a larger range of the brain.

a differential model for selected filter per class. need to align brains

rank brain substructures

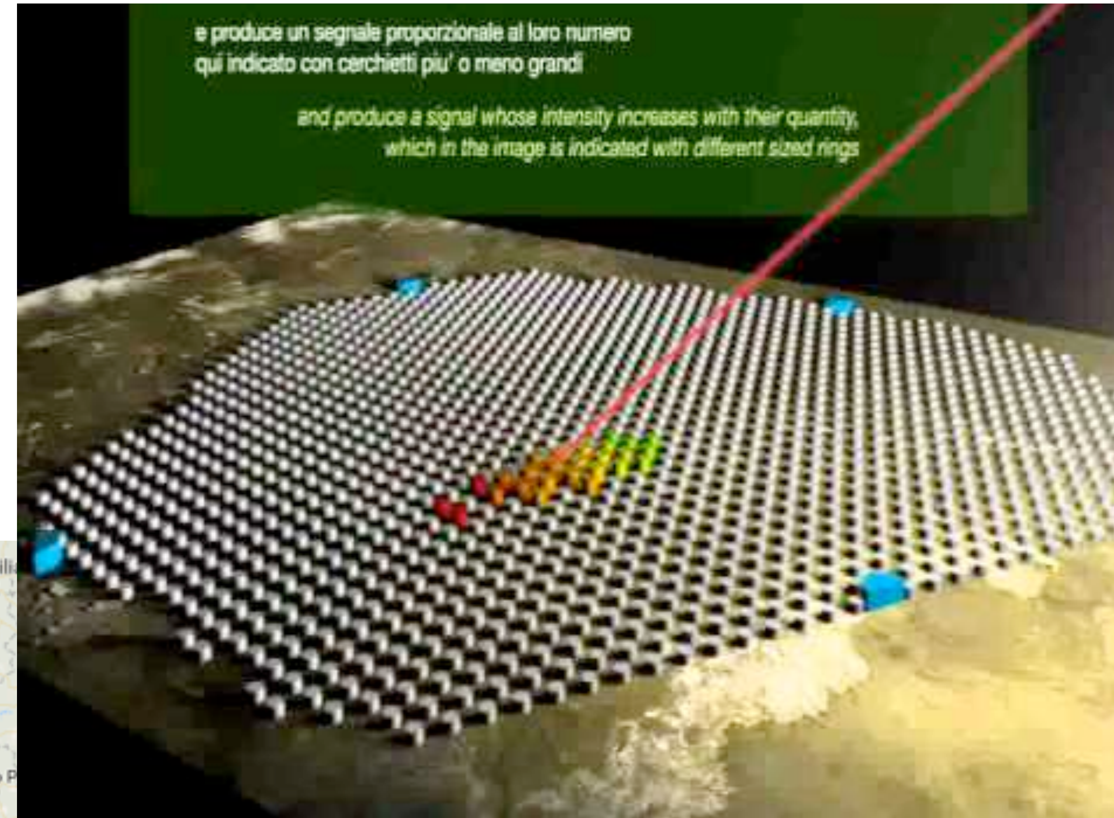
Table 7. Brain substructures ranked according to dBMS (differential brain model summary) when using Filter A of the selected CNN. Substructures with higher dBMS contribute most to differentiate both classes. * shows the substructured identified in the literature where Alzheimer's is mostly located.

	Brain substructure	dBMS		Brain substructure	dBMS
1	Frontal pole *	130.4	9	Left thalamus	123.9
2	Superior frontal gyrus	126.8	10	Right thalamus	123.9
3	Occipital pole	126.2	11	Left hippocampus *	123.7
4	Temporal pole	125.6	12	Right hippocampus *	123.7
5	Superior paletal lobule	125.4	13	Left caudate	123.6
6	Brain stem	125.3	14	Right caudate	123.6
7	Left lateral ventricle	124.2	15	Left amygdala *	123.5
8	Right lateral ventricle	124.1	16	Right amygdala *	123.5

filter identifies typical disease location

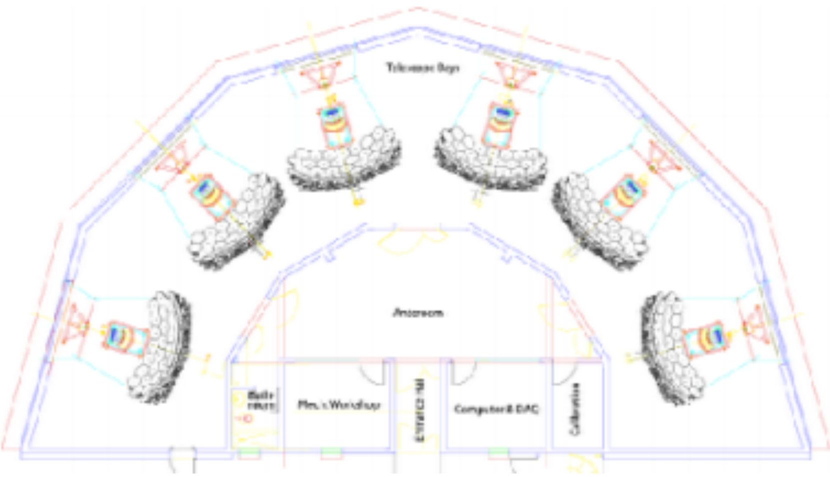
customization

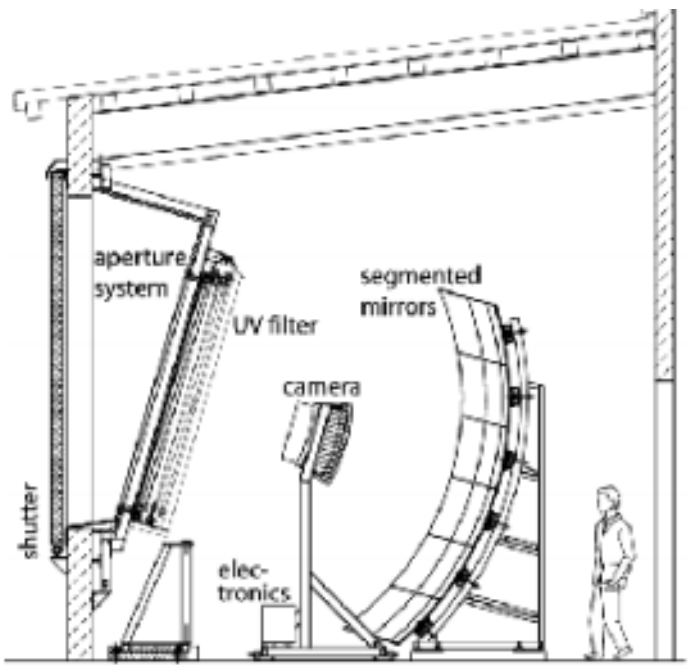
pierre auger
cosmic ray
observatory

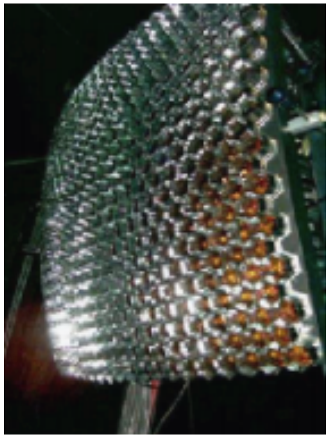


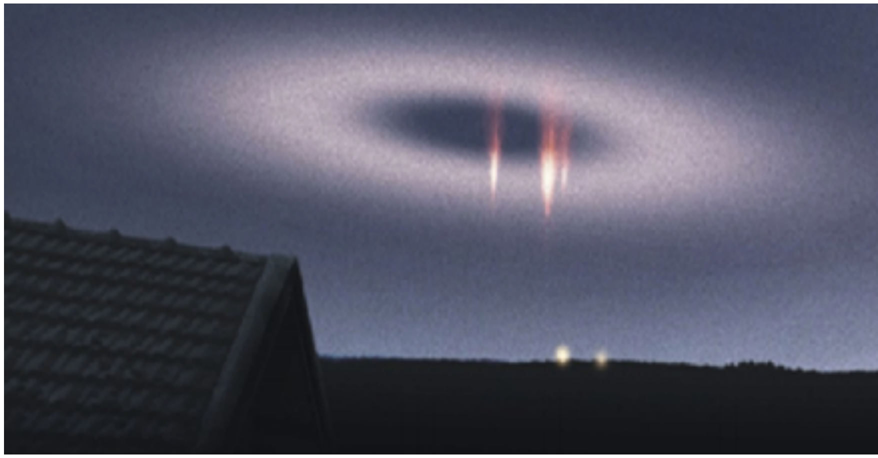
elves detection

transient luminous events



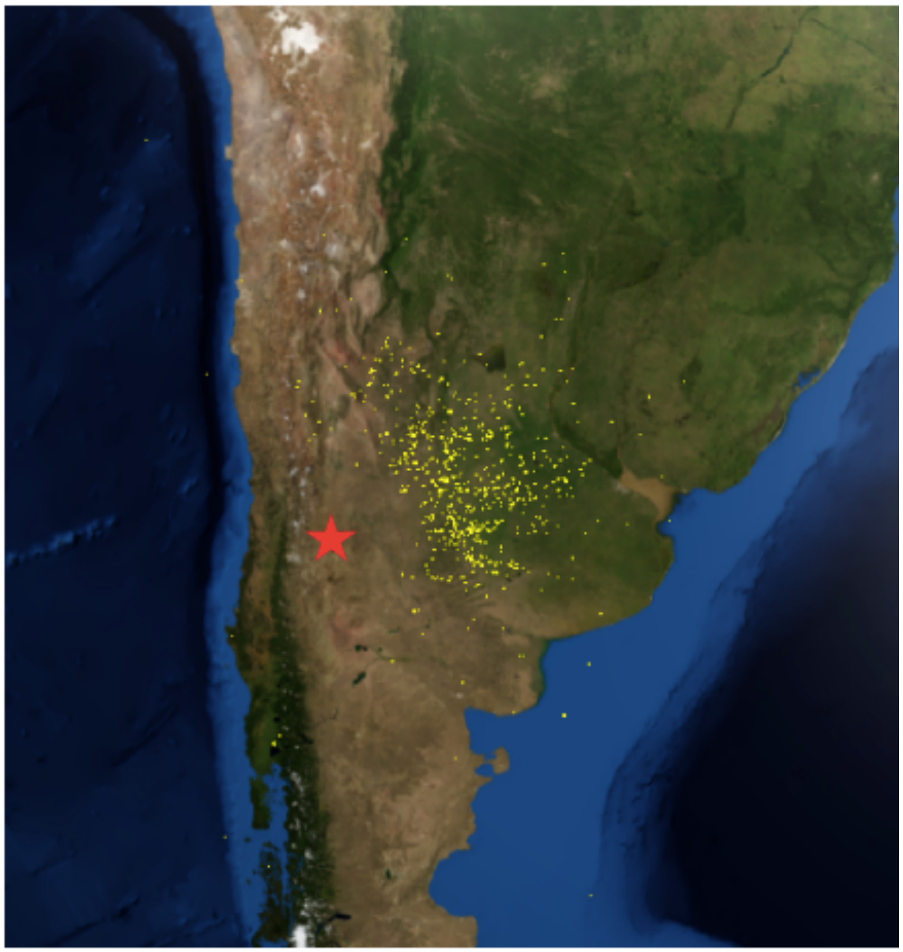


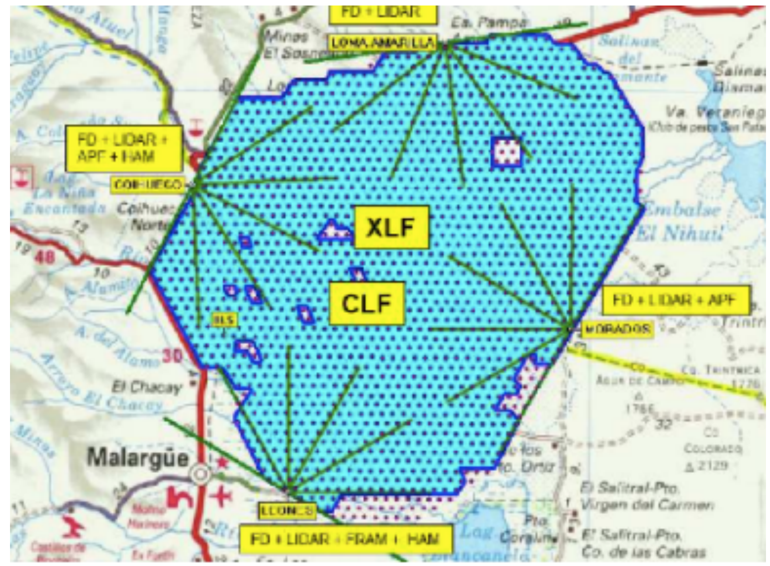




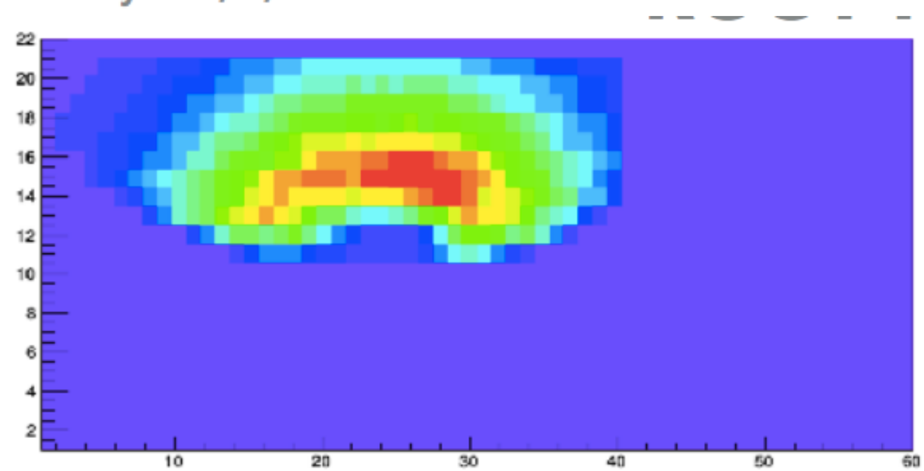
elves detection

transient luminous events – CAN SIMULATE THEM!!!! – IMAGE TIME SERIES


Courtesy of David Grisham and Cody Doyle, CSM, Colorado, USA

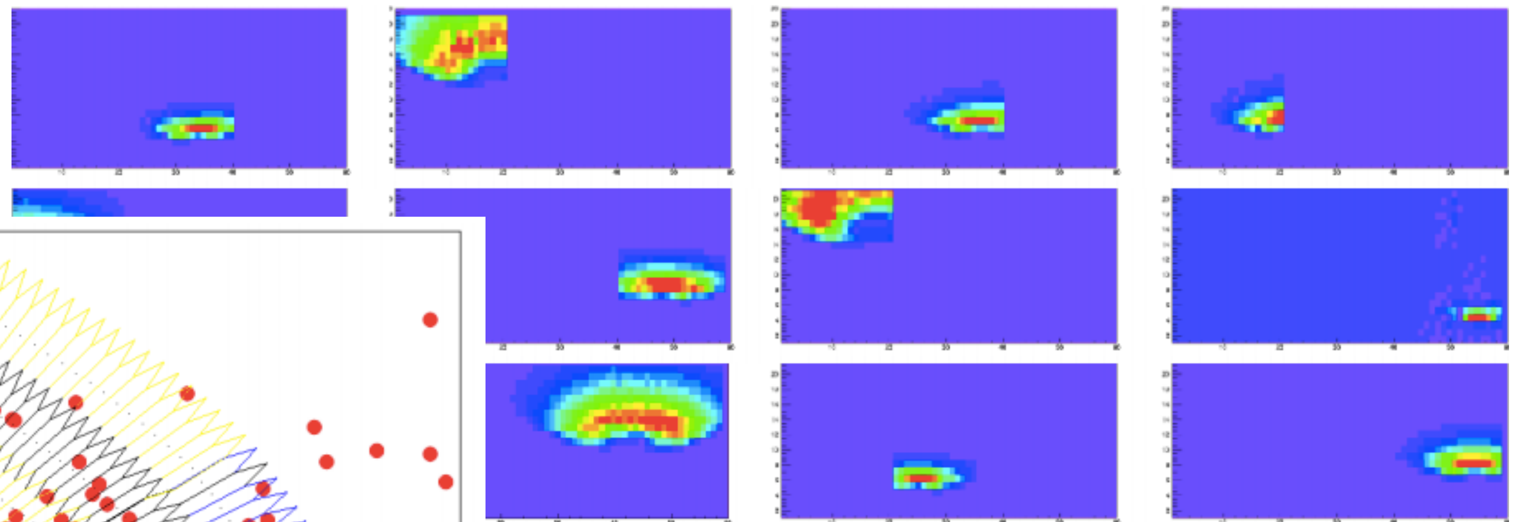


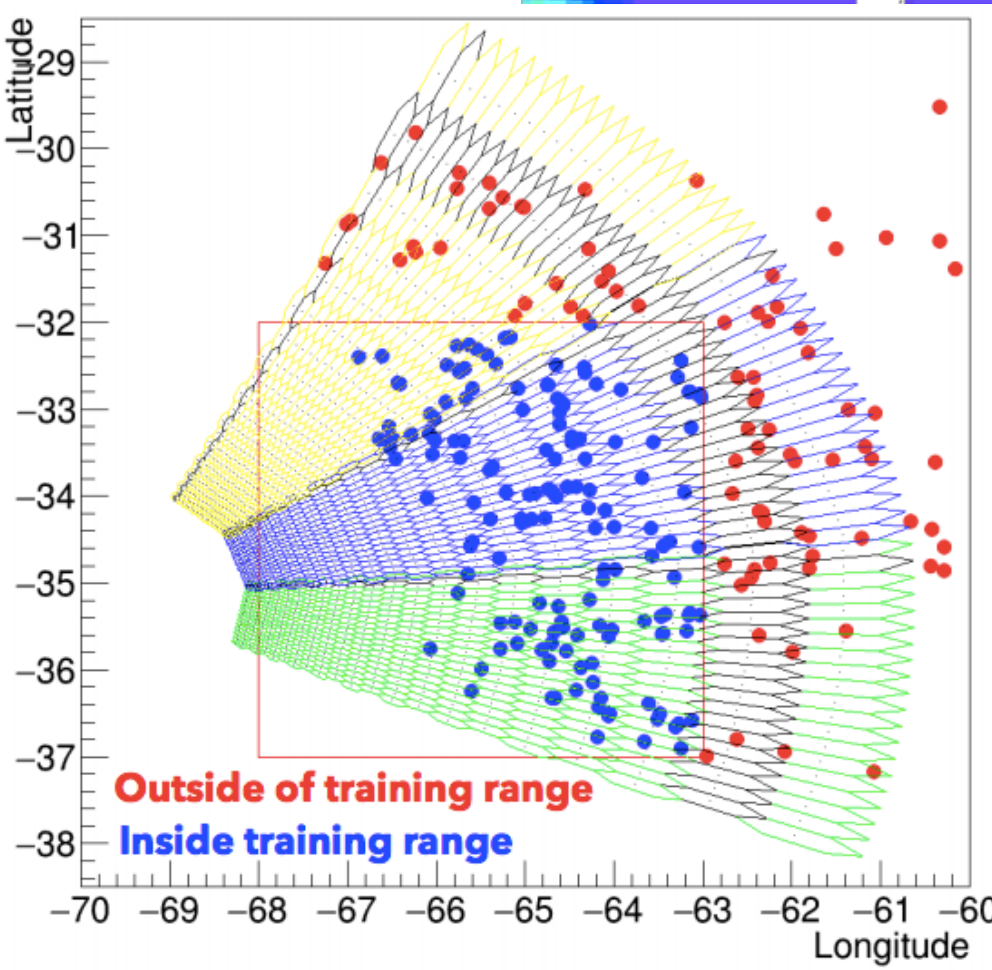
- Eyes 1, 3, and 4 see the most data!



elves detection transient luminous events

goal: reconstruct lightning properties (location, intensity)





initial tests ½ degree accuracy in predicting location (target is 1/10)

want to include geometry in the CNN
loss functions for penalizing camera borders

semantic embeddings

1. use pretrained GoogleNet + VGG
2. extract activations for 50K images + 1000 classes (Imagenet)
3. compute averaged activation for each class
4. spectral clustering of classes (2,3,4…,19 clusters)
5. map cluster to wordnet hypernym/hyponym lexical taxonomy
6. obtain taxonomy INDUCED by the CNN
7. measure layers representativeness

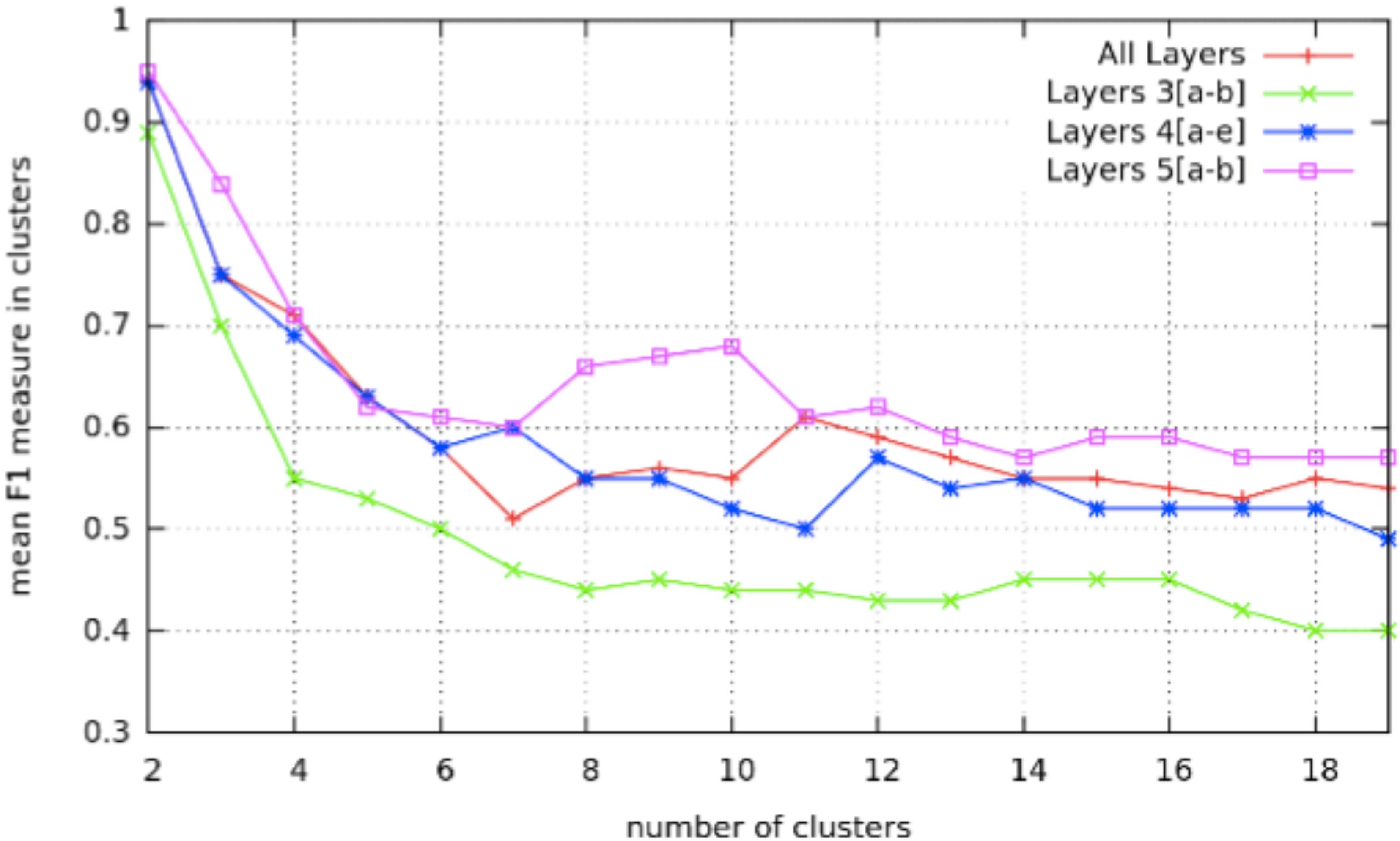
semantic embeddings

induced concept hierarchy

2			Living thing	Artifact				
3			Living thing	Artifact	Conveyance			
4		Mammal	Living thing	Artifact	Conveyance			
5		Mammal	Living thing	Artifact	Conveyance	Artifact (clothing)		
6	Bird	Mammal	Matter (reptile)	Artifact	Conveyance	Artifact (clothing)		
7	Bird	Mammal	Matter (reptile)	Instrumentality	Wheeled vehicle	Clothing	Craft	
8	Bird	Mammal	Matter (reptile)	Instrumentality	Wheeled vehicle	Clothing	Craft	Structure

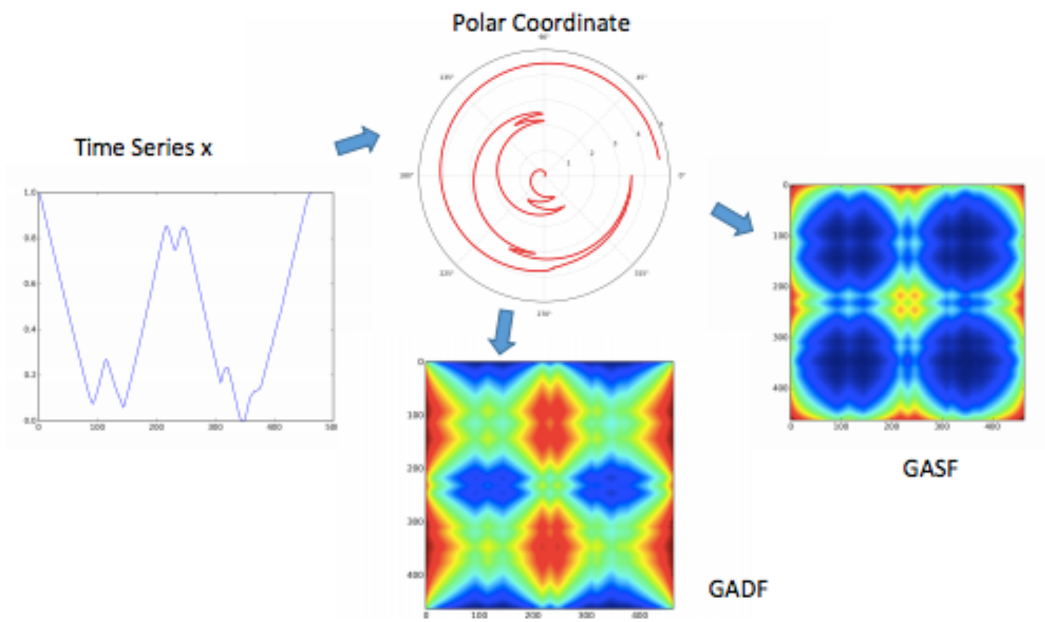
semantic embeddings

last layers (5ab) offer greater distinguishability



1d signal analysis with CNNs

1. convert 1D signal to 2D image



2. use pretrained CNN for classification
3. use standard LSTM for classification

1D signal analysis with CNNs

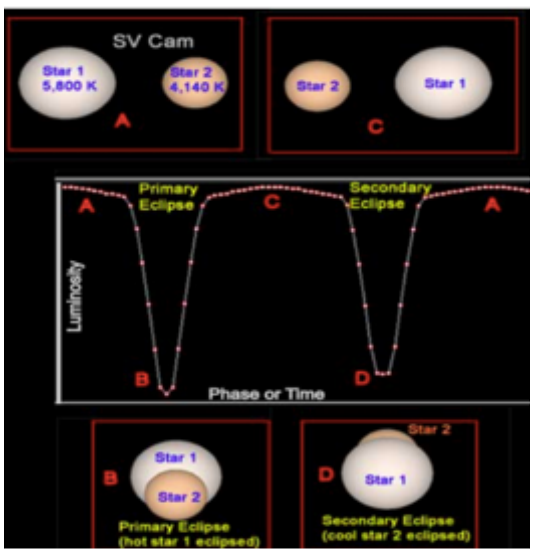
Datasets

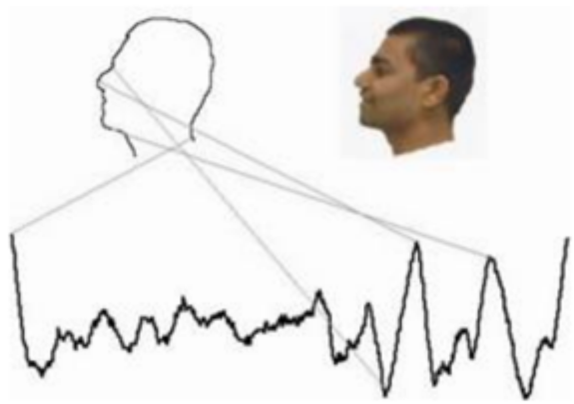
Startlight: brightness of a celestial object as a function of time

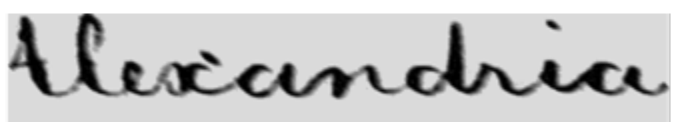
Face: facial outlines

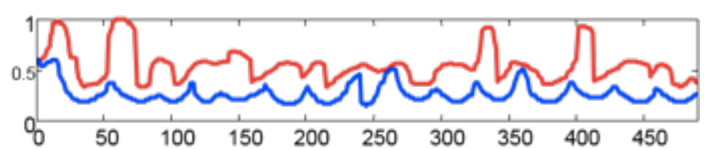
Earthquake: event about to occur based on recent readings

50words: Handwritten words outline









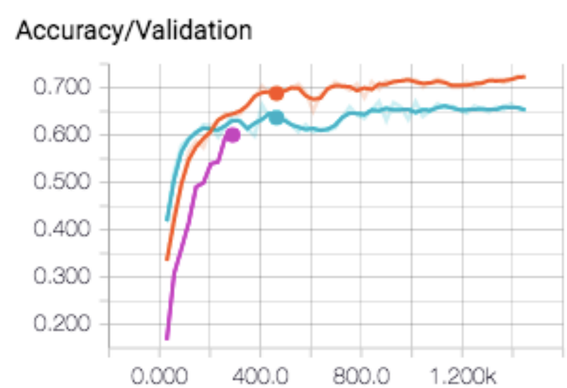
1D signal analysis with CNNs

RNNs are endowed with store/forget gates

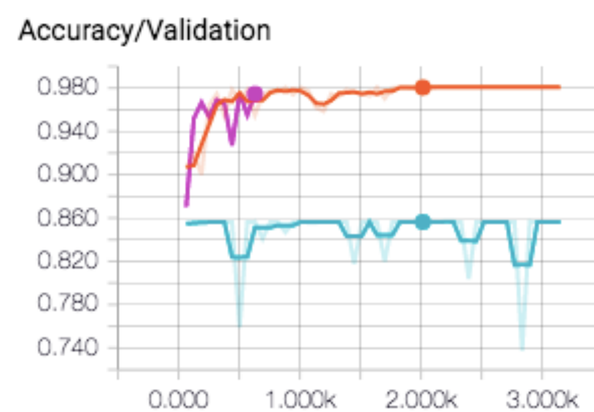
CNNs detect patterns on Euclidean localities

explore 1D to 2D transformations

understand better what tasks are better suited for each



Name	Smoothed	Value	Step	Time	Relative
50w_cnn	0.6879	0.6857	464.0	Fri Oct 20, 14:57:04	25s
50w_lstm	0.6366	0.6352	464.0	Fri Oct 20, 14:58:20	16s
50w_vgg	0.6000	0.6000	290.0	Fri Oct 20, 14:56:36	1m 12s

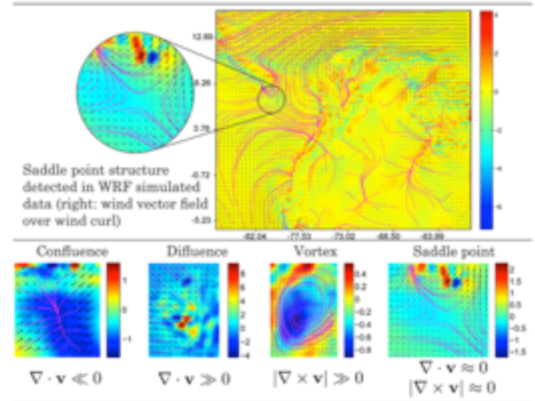


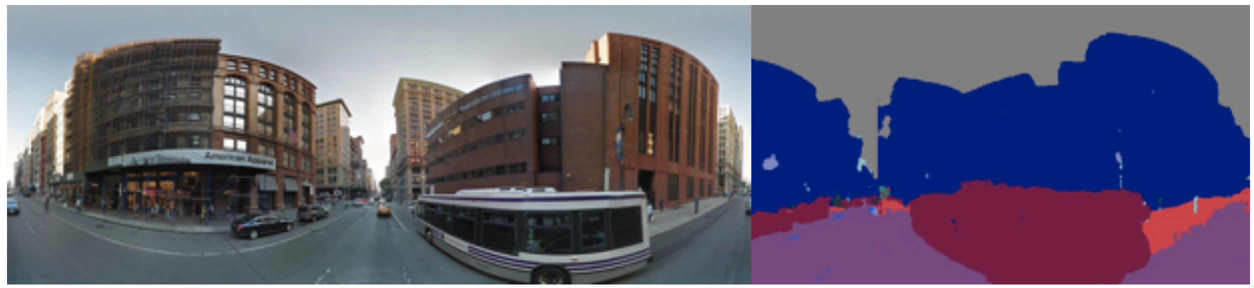
Name	Smoothed	Value	Step	Time	Relative
star_cnn	0.9805	0.9808	2.016k	Fri Oct 20, 14:04:11	3m 31s
star_lstm	0.8563	0.8562	2.016k	Fri Oct 20, 14:07:03	48s
star_vgg	0.9745	0.9745	630.0	Fri Oct 20, 13:53:11	6m 53s

research projects

deep learning

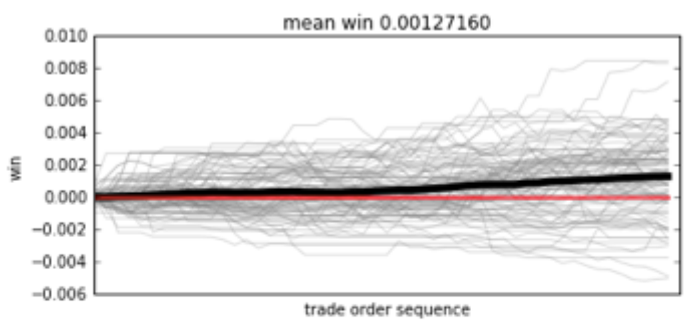
image processing (biomedical, climate, object detection)
time series (finance, KPIs, text mining)

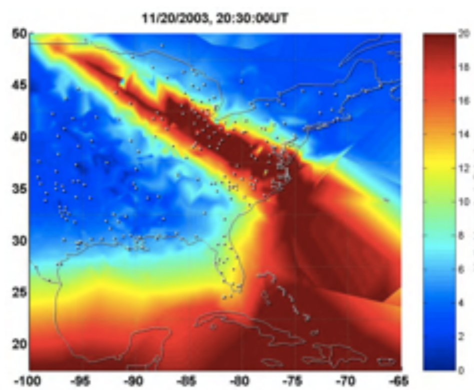


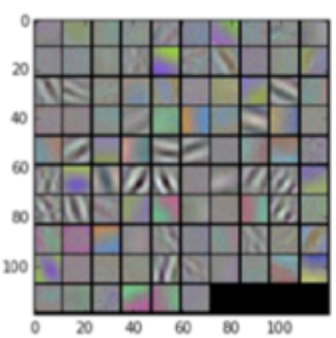


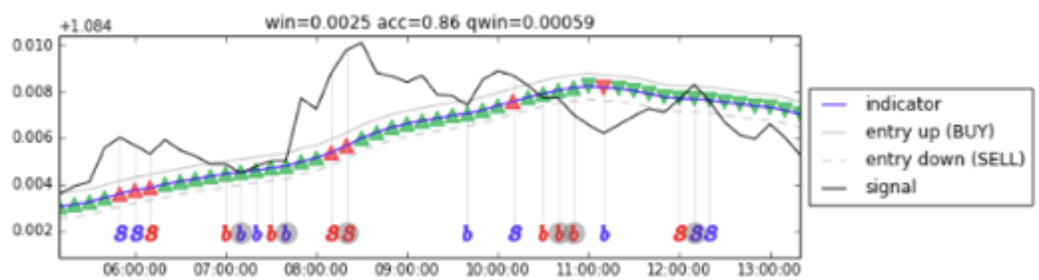
gnss

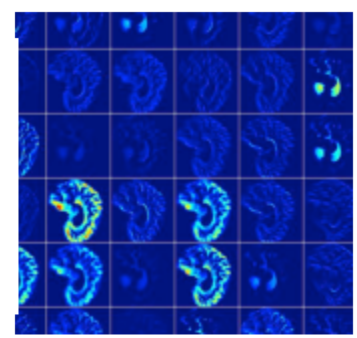
ionospheric modelling
precision positioning
intelligent transport systems











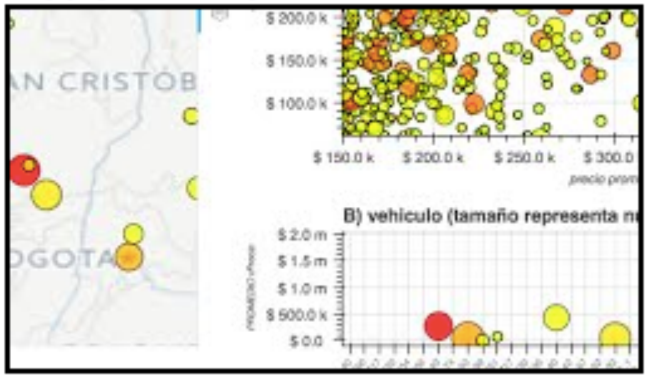
projects with industry

frontier x analytics

www.frontierx.co

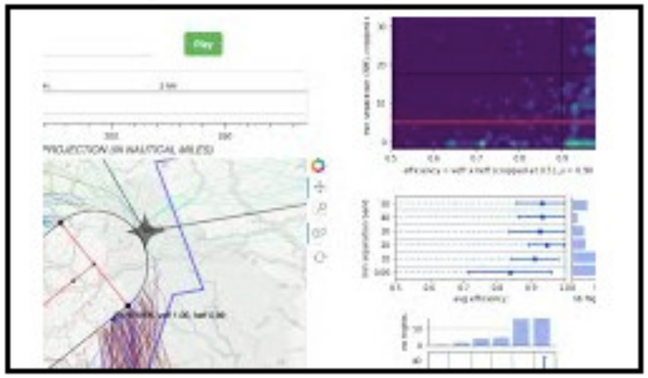
transportation

supply chain optimization



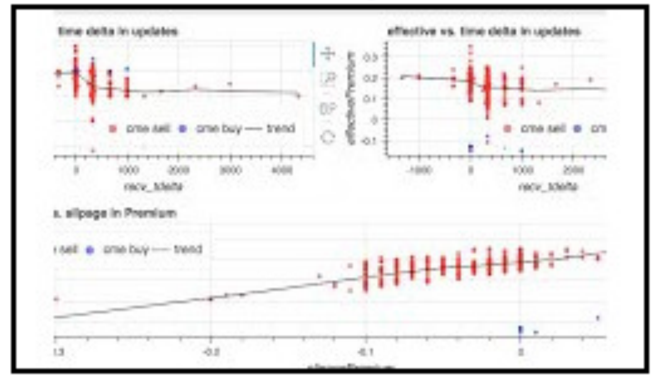
aeronautics

efficiency of descent operations



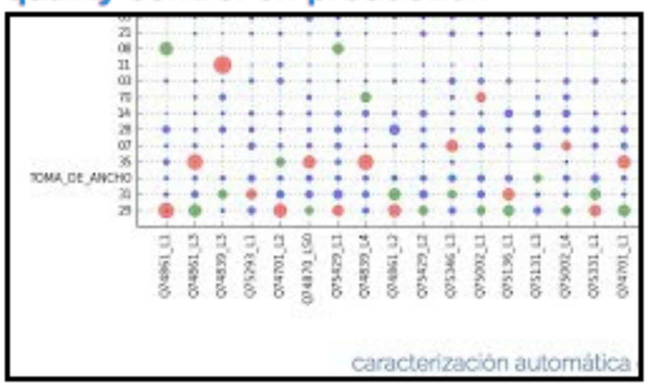
finance

brokers - markets - operations



textile

quality control on production

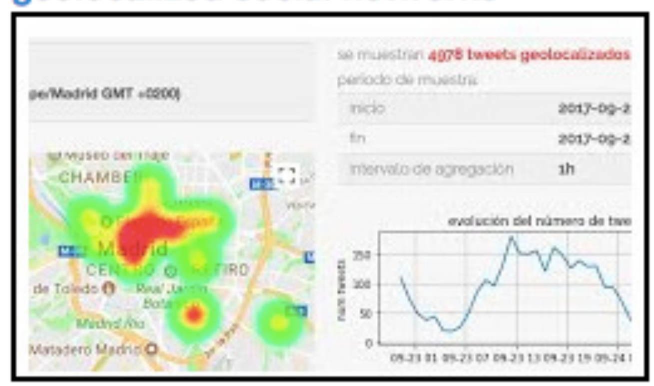


tourism

online reputation

mobility

geolocalized social networks



CERN

Sun microsystems

PildoLabs move smart

CENTRO EXTREMEÑO DE TECNOLOGÍAS AVANZADAS
CETA Ciemat

PLANETARIO CETA·CIEMAT

mindLAB

UNIVERSIDAD NACIONAL DE COLOMBIA

BIOS
CENTRO DE BIOINFORMÁTICA Y BIOLOGÍA COMPUTACIONAL

Universidad Industrial de Santander

UIS

CAGE

Computo Avanzado y a Gran Escala
Advanced and Large Scale Computing
Research group


thnx