Orchestrating a brighter world

NEC

Taking the Fifth:
Teaching Deep Networks when to ‘shut up’

Tsvi Lev
General Manager
Israel Research Center
NEC Corporation
Or.katz@necam.com

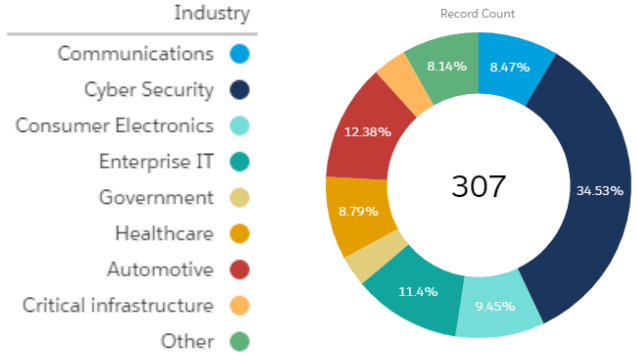
NEC Global Network

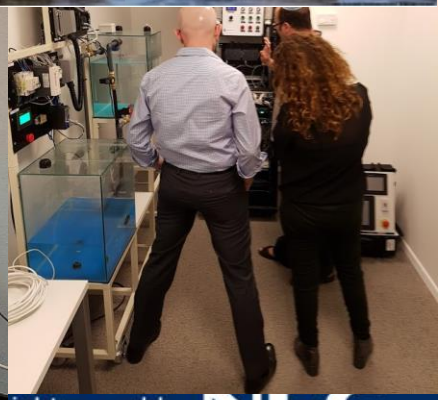
- Conducting Business In Over 160 Countries
- Network Of 9 Global Research Labs
- ~0.5% of Revenue Allocated to Research: ¼ Billion$

Established 1899

NEC's Israeli Research Center

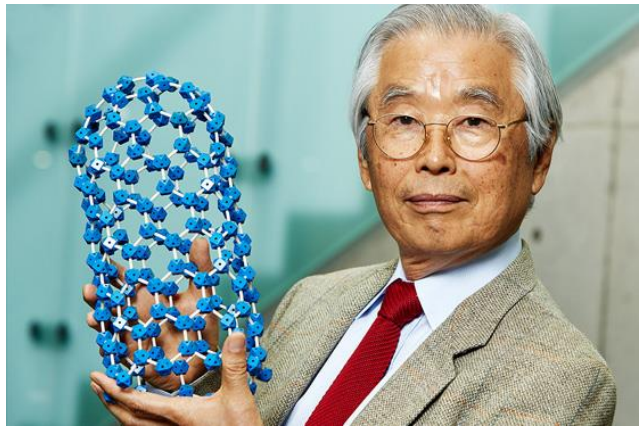
- **25 FTE: Cyber, Algorithms/DL, Outreach**
- **4 PhD, 7 MSc, rest BSc/BSEE.**
- Total investment of NEC in Israel ~10M$.
- Collaborations with **MIT, BIU, BGU, TAU.**
- HaGihon Plant Digital Shadow for training and testing
- Working on embedded DL systems
- Partnership with 2 HMOs for medical data, **installed in Assuta Radiology**
- Involved with 2 Accelerators – DRIVE and Alpha-C
- Work almost exclusively in Python ☺

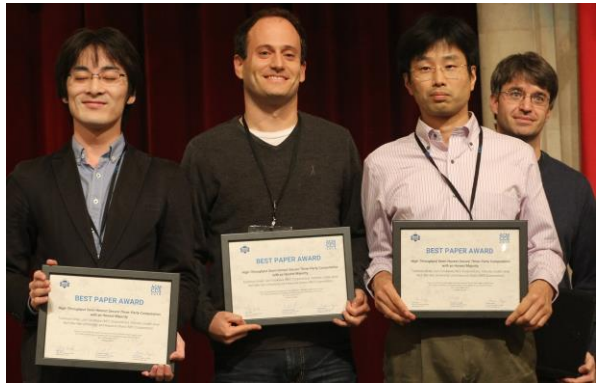


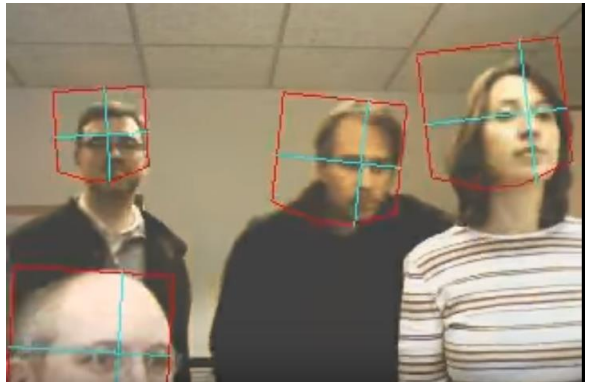


NEC research firsts..

- **Yann LeCun** – face detection using CNN (2003..), NEC Labs Princeton
- **Geoff Jiang** (ML for Cyber Security) – now VP AI at Ant Financial
- **Yuanqing Lin** – Director of the institute of DL, Baidu
- **Sumio Iijima** – Carbon Nanotubes (1991)
- **Furukawa, Chandraker -** best of ACM CCS 2016, CVPR 2014
- 4 Consecutive times #1 in **NIST Face Recognition**

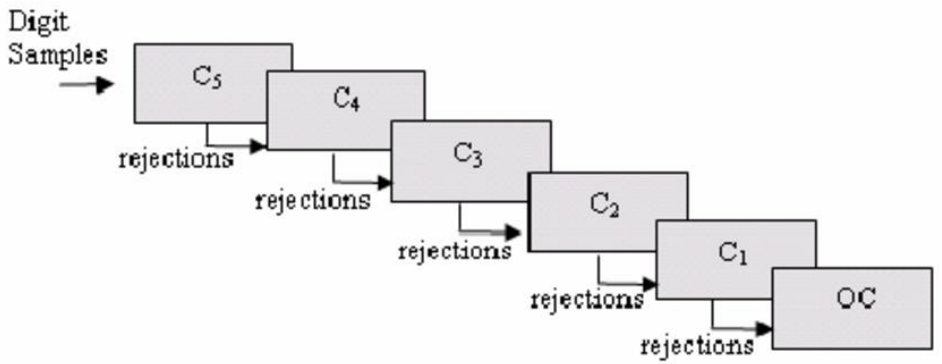


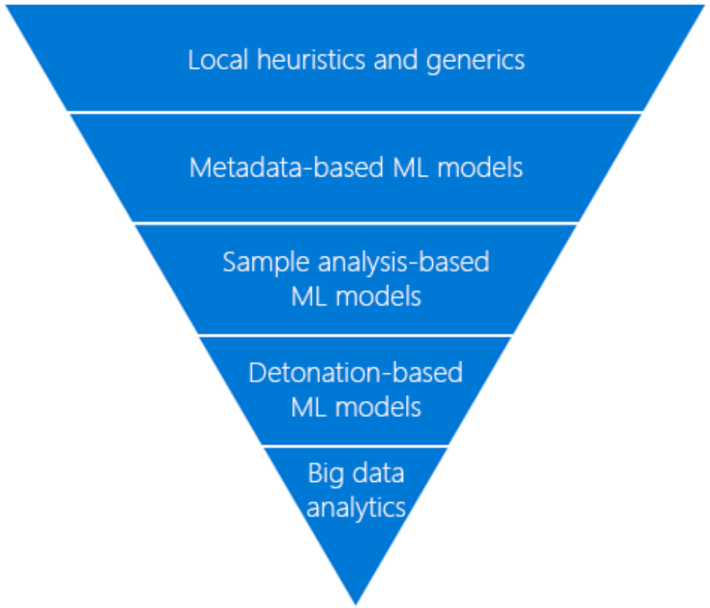




What Abstaining means in ML

- A classifier/predictor may have some **certainty measures** (e.g. Logit values)
- Research tends to focus on **improving performance** (TP/FP,AUROC curve, Top-1, Top-5) – but real world examples require higher performance or 'no-go'.
- Often, declaring the classifier/predictor as 'uncertain' is better than using a prediction
- Examples:
  - Medical Diagnostics – it is better to call upon a human expert to resolve than make a bad prediction
  - ADAS – alert the driver
  - Chain of command (e.g. Cyber) – pass 'uncertain' cases to OTHER classifiers.
  - Search for a 'more certain' sample –e.g. in fertilization.





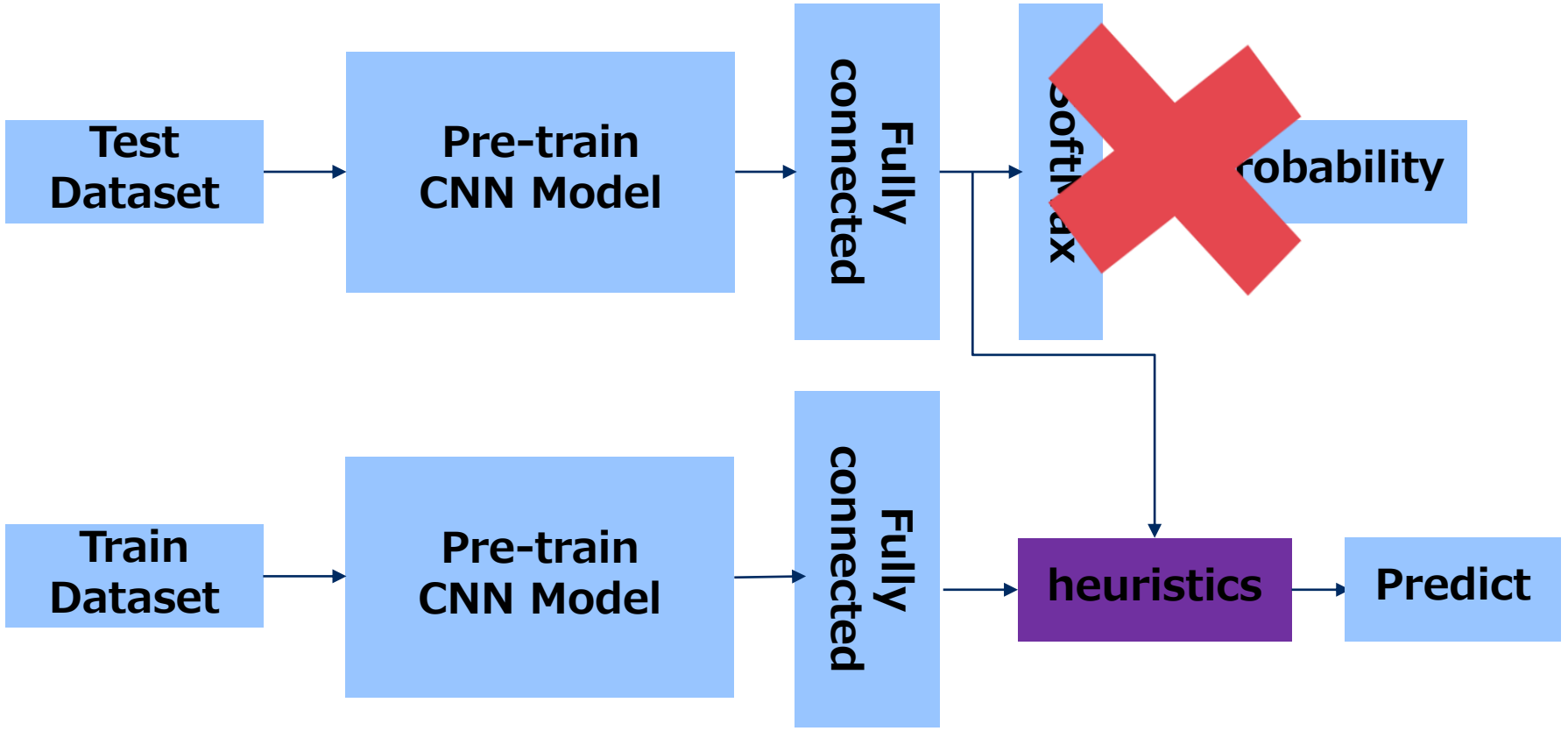
As ML/DL reaches the real world, research adapts

- Selective Classification for Deep Neural Networks – Geifman et al. (Technion): **use the Logit value (adaptive selection),**
- Learning to abstain via curve optimization - **use the Logit value and examine Precision-Recall Curve.**
- Dropout as a bayesian approximation: Representing model uncertainty in deep learning– Gal el al. - **Test time Dropout**
- **"Explanation" methods** – if we don't like the explanation we don't trust the result.
- Deep k-Nearest Neighbors: Towards Confident, Interpretable and Robust Deep Learning – Papernot et al. (Penn State) – **kNN applied to various DNN layers**



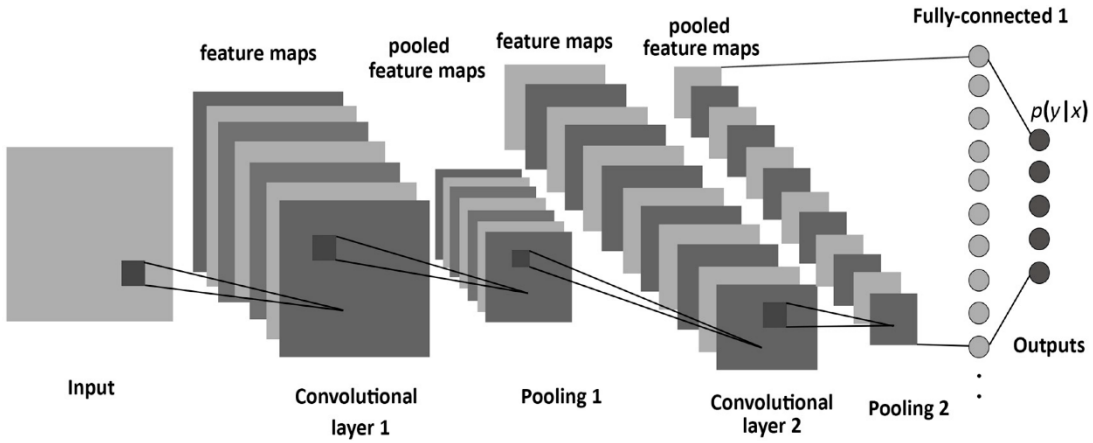

Our approach

Our starting point is a pre-trained model and we evaluate different heuristics as to when the network should abstain from classification.

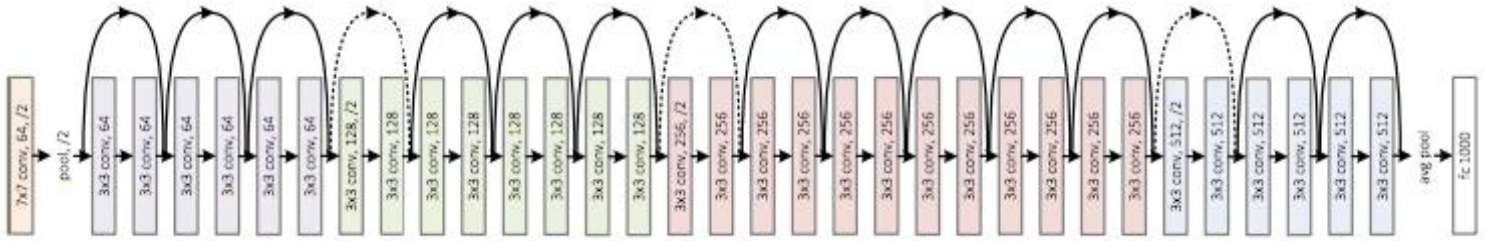


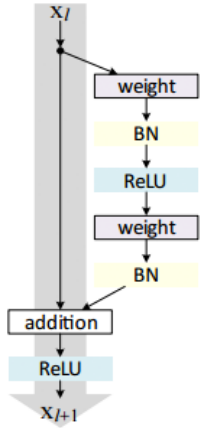
Network Architectures evaluated

Simple convolutional network



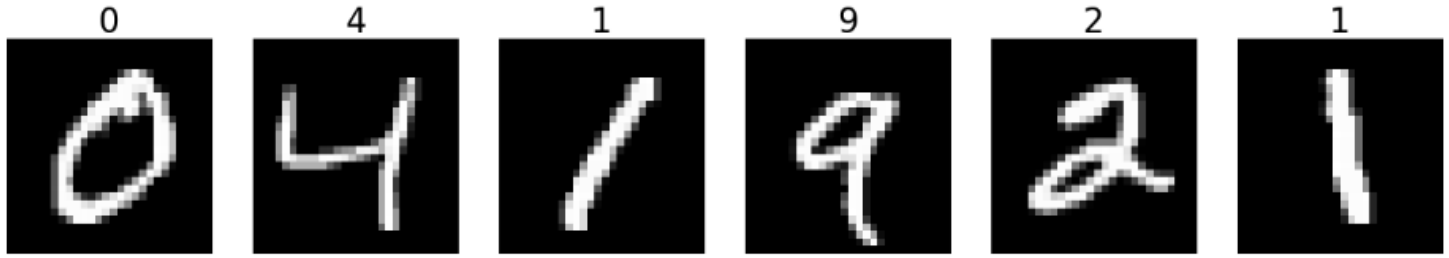
Deep residual network





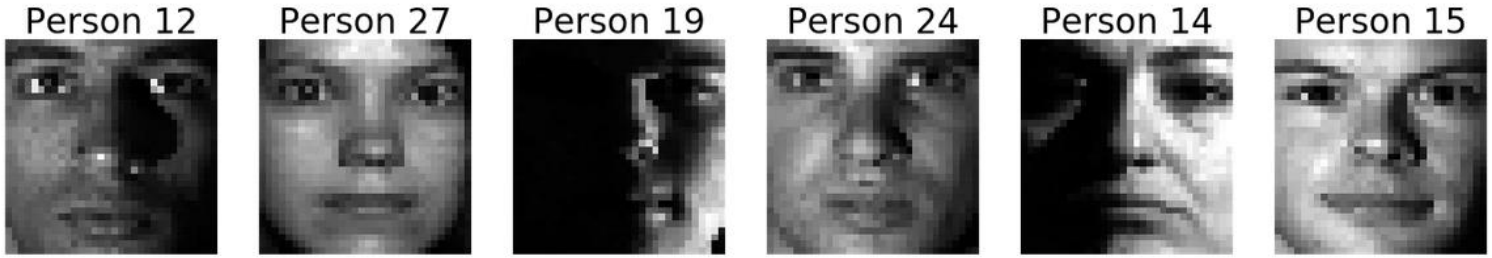
Datasets

Mnist



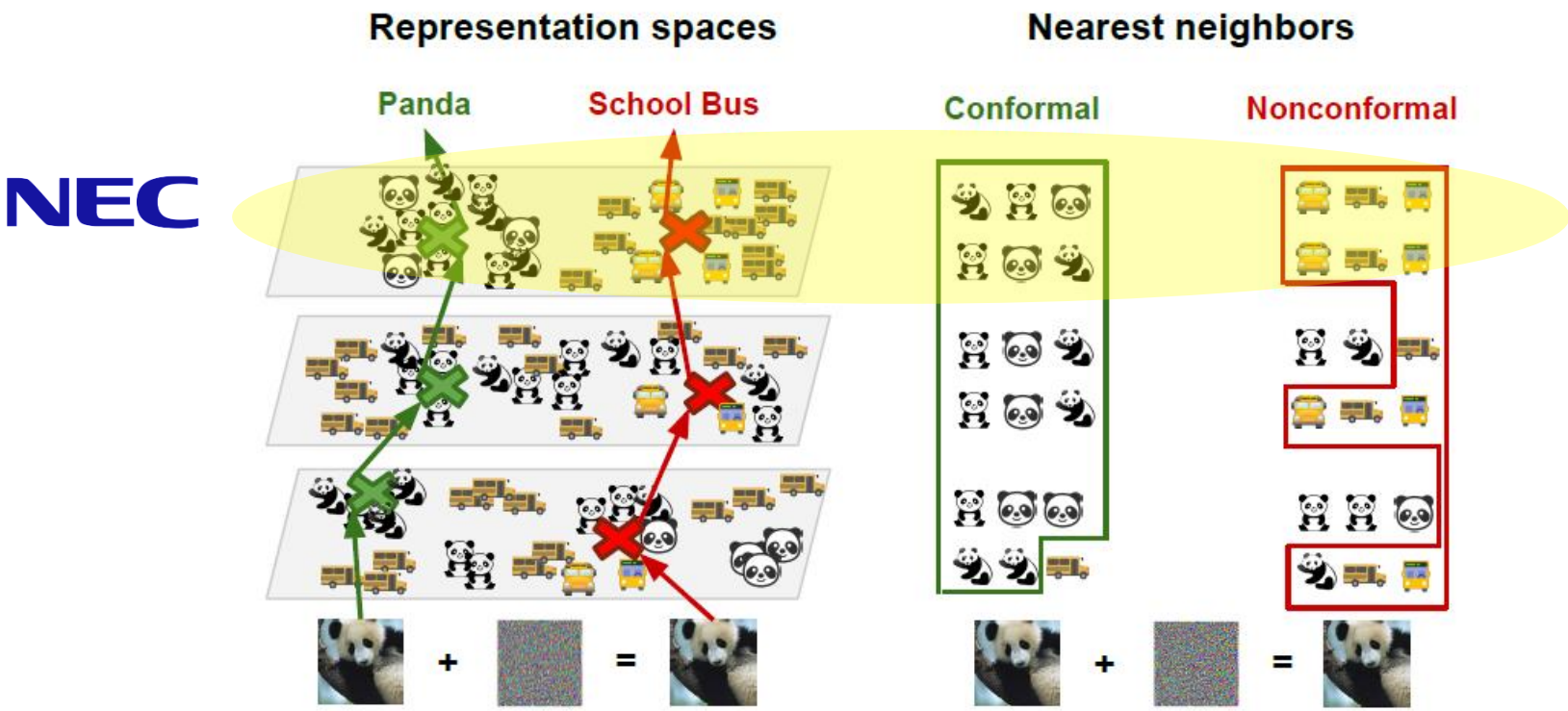
Cifar10

Yale32



Heuristics

- SoftMax - In this case we predict the probability of the test class and control the sensitivity of the results by setting a threshold.
- Metric distance (Euclidean or Chebyshev) –
  - Let V be the output vector of the network prior to the softmax layer
  - For each test image we compute the distance between V(test) and V(train) for the entire training set
  - Check how consistent the network is across the closest training examples



In Code

- Note: the actual distance measure (L1,L2,etc.) is apparently not the key factor.
- In terms of implementation, we used brute force calculation. The Penn paper uses locality sensitive hashing which is much faster.

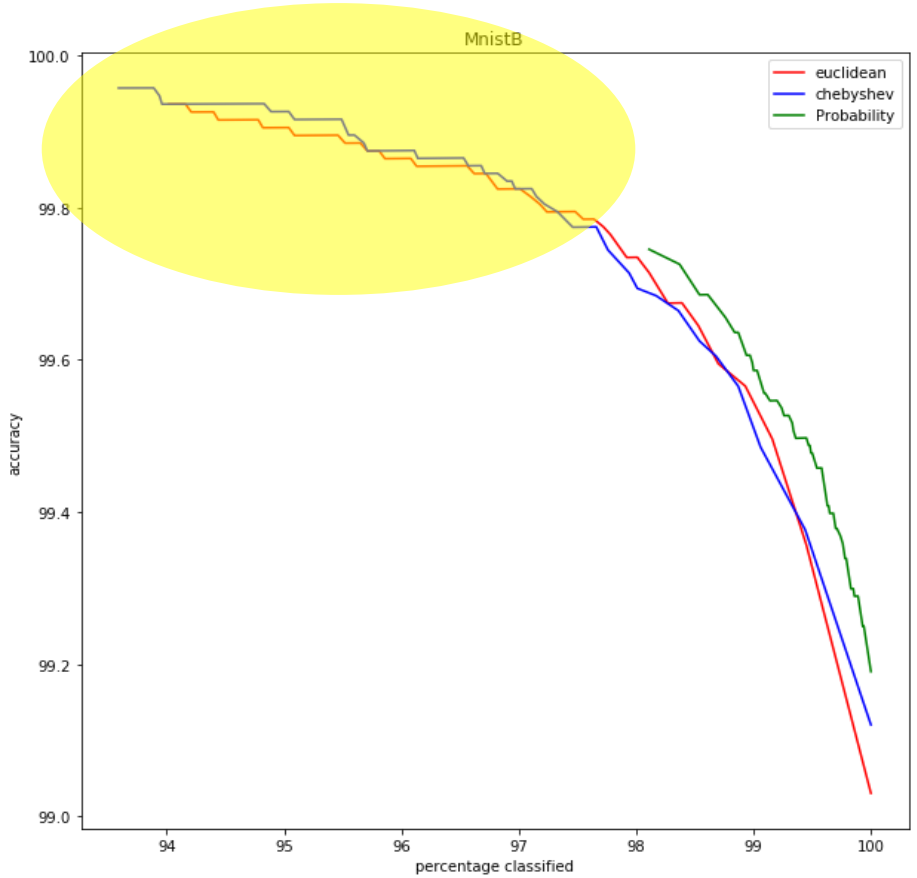
```
def dist_metric(y_train,y_test,Y_Vector_train,Y_Vector_test,w,name='euclidean',batch=100):
    l=len(Y_Vector_test)
    est_labels=np.empty([l,w])
    for i in tqdm(range(0,batch)):
        dist_matrix=scipy.spatial.distance.cdist(Y_Vector_train,Y_Vector_test[i*batch:(batch+i*batch)],metric=name)
        ind_matrix=np.argsort(dist_matrix,axis=0)
        labels_ind=ind_matrix[0:w,:]
        for j in range(0,w):
            est_labels[i*batch:(batch+i*batch),j] =y_train[labels_ind[j,:].T]
    return est_labels

knn_labels = []
for name in ['euclidean','chebyshev']:
    knn_labels.append(dist_metric(y_train,y_test,Y_pred_train,Y_pred_test,w=100,name=name,batch=batch))
```

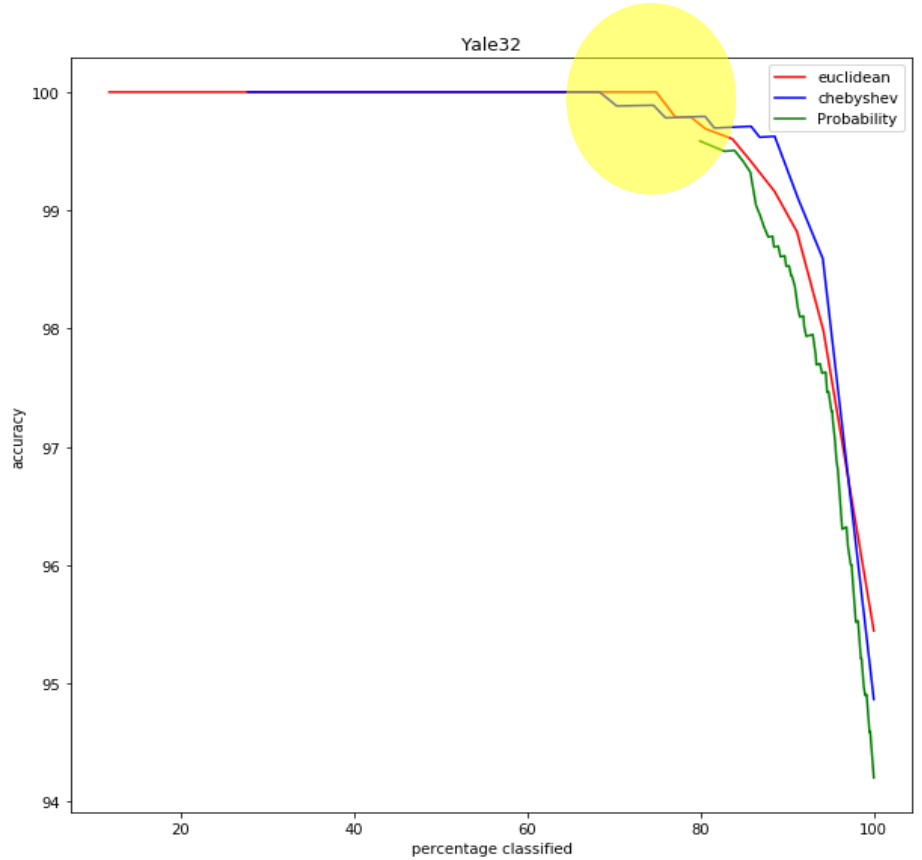
Results – Euclidean distance

Dataset	Score	W=2		W=5		W=10	
	Accuracy%	Accuracy%	Classified%	Accuracy%	Classified%	Accuracy%	Classified%
Mnist-A	98.04	98.93	97.69	99.48	95.27	99.68	92.23
Mnist-B	99.19	99.49	99.16	99.64	98.53	99.73	97.92
Yale32	94.20	97.97	94.21	99.42	85.75	100	74.90
Cifar10	84.81	92.44	74.49	95.80	63.00	97.48	54.39

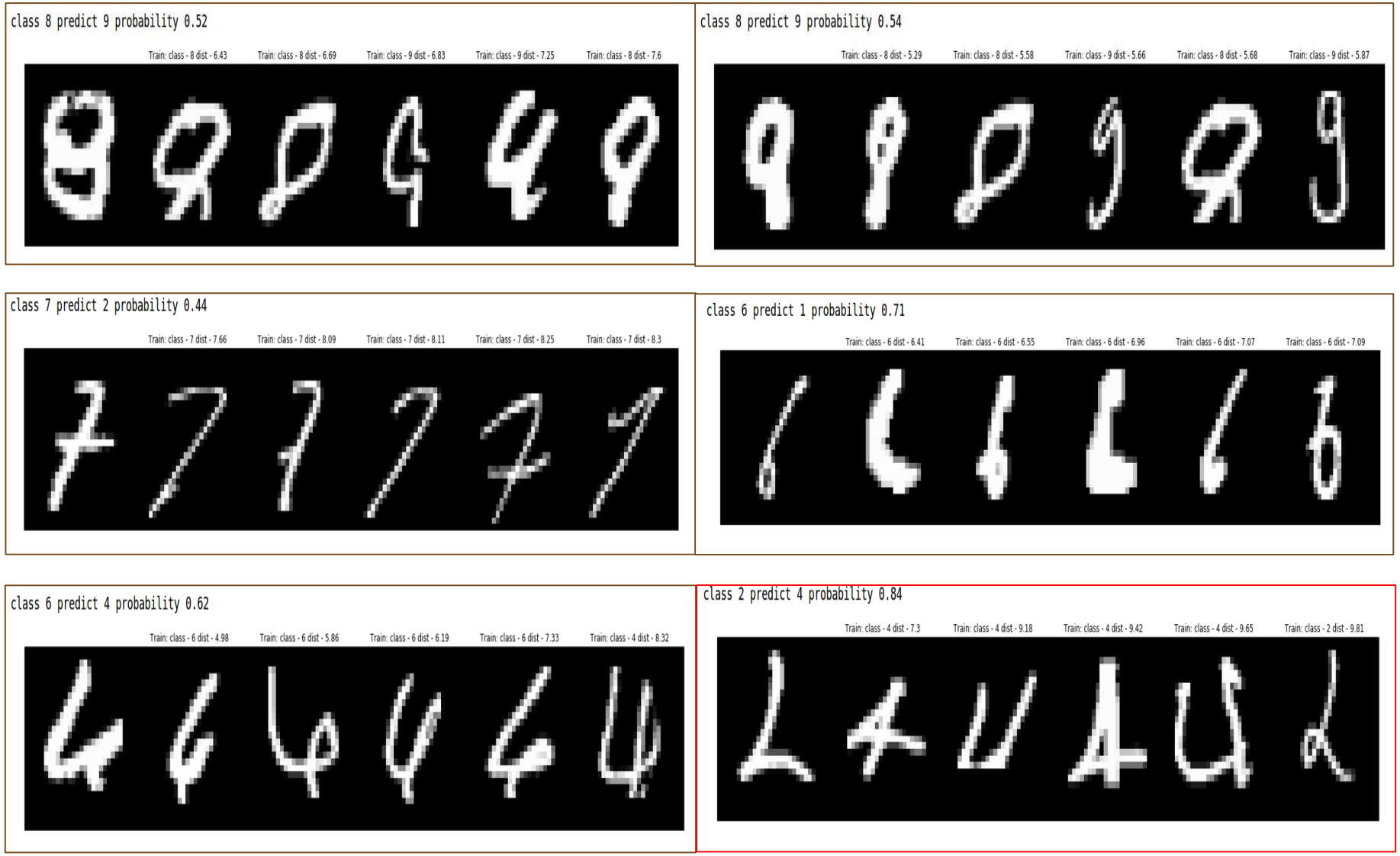
Table 1 shows the performance of the homogeneous nearest neighbors heuristic with Euclidean distance for the simple convolutional network with the number of neighbors set to W = {2, 5, 10}

Results - “Simple” Network Model





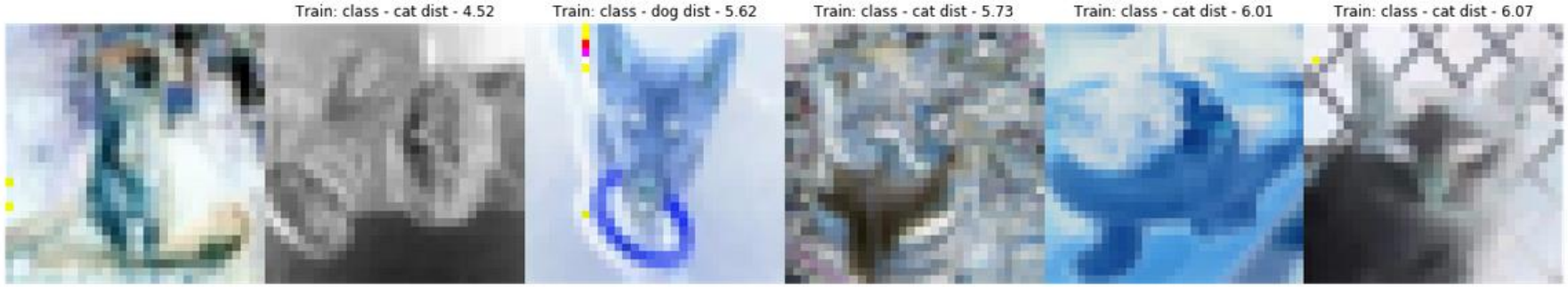
Examples - MNIST



Examples - CIFAR

class airplane predict bird probability 0.81

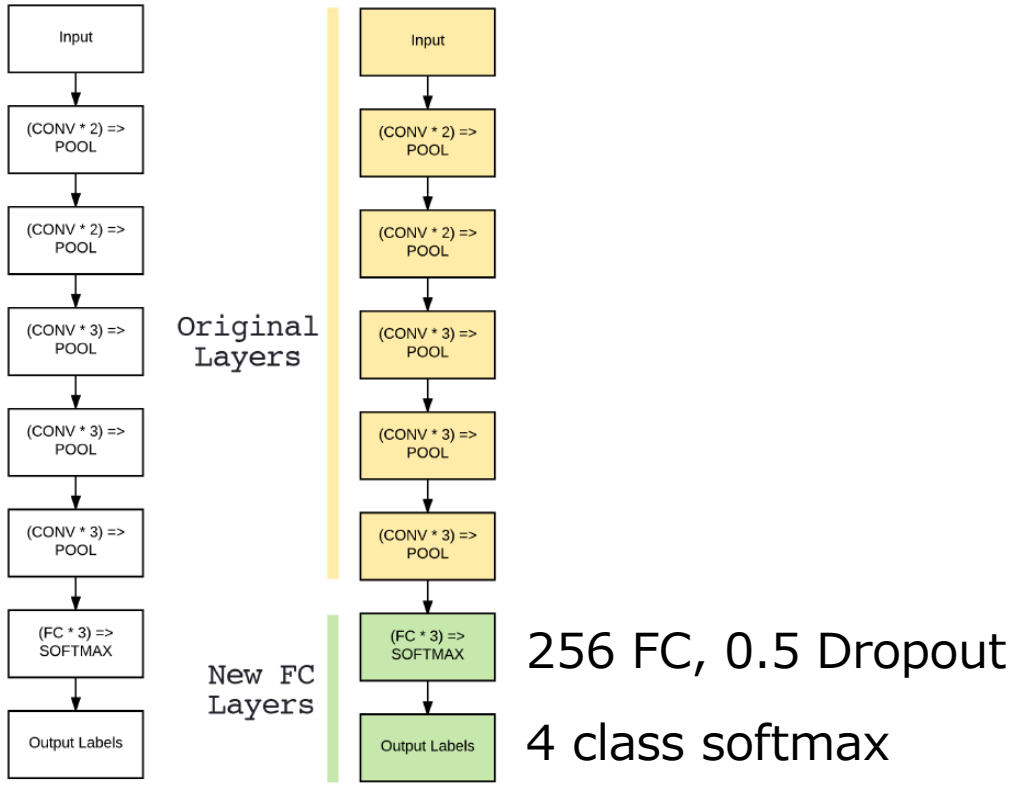
class cat predict deer probability 0.51



class ship predict airplane probability 0.29

Going Deeper

- We performed transfer learning on a ‘standard’ VGG-16 to differentiate between 3 classes of animals: added 1 FC (256), 50% dropout, Softmax final layer.
- 400 training samples each, 10 Epochs with all VGG layers frozen, then 10 more with all layers open (lower rate), reaching >90% TP for all classes
- Tested for kNN on the FC layer and ‘predicted’ only on samples where 3 NN in the FC layer were of the same (true) category.
- Average Classification Success jumped
  94.73% => 97.2%
  (**5.3%=>2.8% error**)
  for population of 88% of samples!



Some ‘intuition’ (fancy hand-waving here..)

- See below FC last layer activations in an MNIST model – a sparse set of strong activations are the trigger for the SoftMax classifier.
- CERTAIN combinations of those neurons seem to be the signature of ‘normal, we’ve seen this before’ while NEW combinations/other neurons might be suspicious.
- CNN (lower) layers are spatially constrained the lower you get – so indicate ‘image similarity’ rather than class similarity.

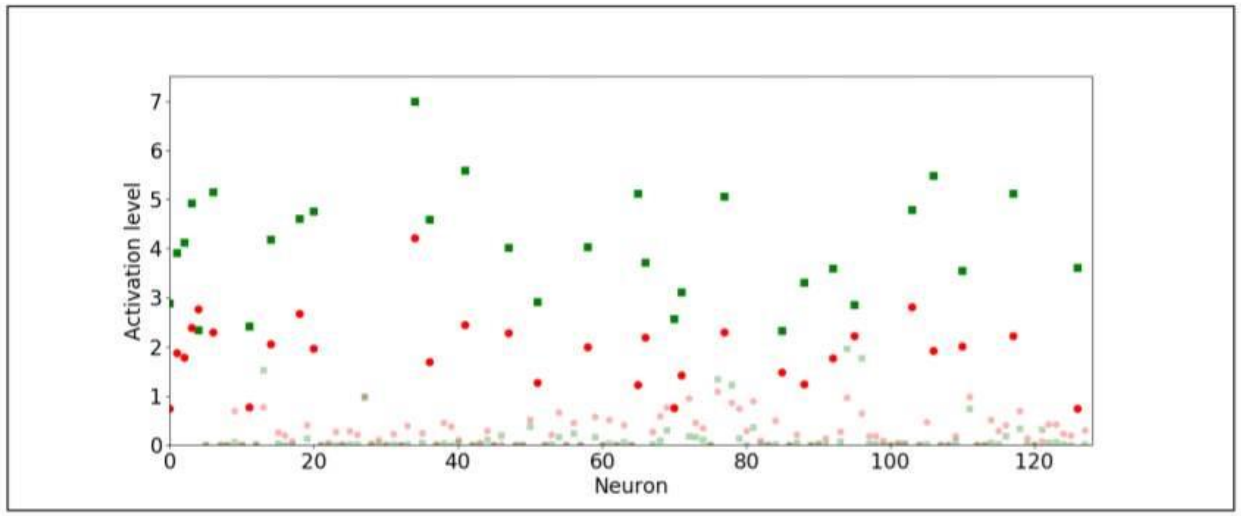
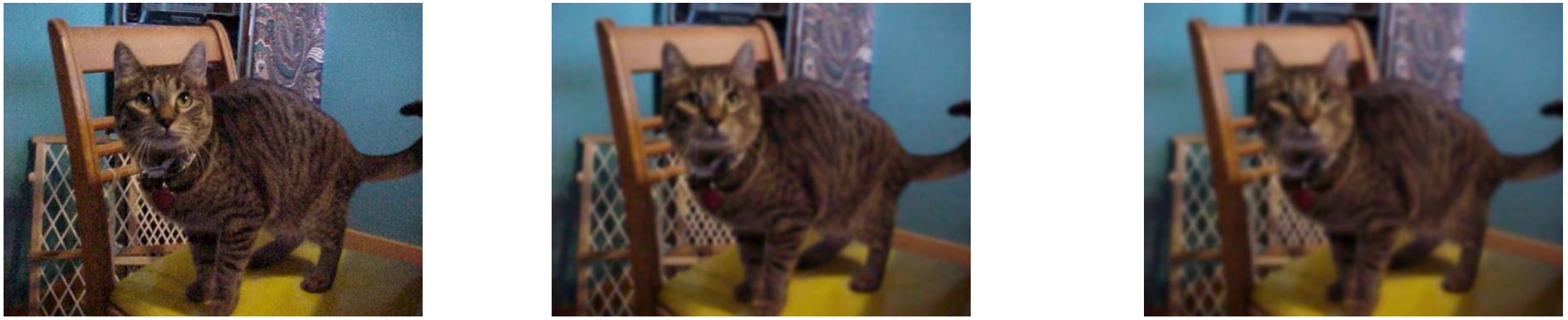


Figure 1: Average response values of neuron activations for class "8" on the MNIST dataset; green squares, true positives, red circles, false negatives

Using kNN to detect input issues

- For systems in operation over time, input population may change – weather, time of year.
- These changes can be very hard to detect, even if they affect the performance – the system DOES NOT KNOW it is wrong…
- Logit values are pretty much meaningless – by the time you get consistently low values, things are really bad and you are probably an ‘expensive random’.
- Yet kNN distances and class uniformity are a good indication **before** logit collapse.
- **kNN yourself** – input images under a transform compared to themselves



Future planned work

- **Detection of environment shift** – camera change, population shift
- **Explanation generation** via kNN
- Data augmentation
- Path Integral approach

Come Join Us!

Or.Katz@necam.com