Introduction to Neuromorphic Computing

With an emphasis on General Vision's consumer-affordable CM1K chip

Keith Wiley
kwiley@keithwiley.com
http://keithwiley.com

Initiated: 20160412
Completed: 20160616

Neuromorphic computing

A novel processor design,
The *neuromorphic processing unit* (NPU)*,
Inspired by the brain

Consists of numerous simple processors, which:

- Process local storage
- Process in massive parallel
- Communicate only with local units
- Aggregate results automatically
- Consume extremely low power
- And in some cases, scale modularly to increase performance.

*Compare with CPU and GPU.

Von Neumann vs. neuromorphic architectures

Von Neumann	Neuromorphic
Small number of general and powerful processing units	Larger number of specialized, weak processing units, often constrained to limited functions
Essentially serial except for mild multicore options	Massively parallel
Processing and memory separated from each other	Processing units only act on their own localized memory
Process by moving data from memory to CPU registers, computing, then moving result back to memory	Process by a parallel data broadcast, then local computing, then global aggregation
High power requirements (50–200+ W)	Low power requirements (.5–1 W)
Data-parallel algorithms scale linearly in time with problem size by iterating over the data	Data-parallel algorithms scale constantly in time with problem size by parallel broadcast and simultaneous processing

GPGPU vs. neuromorphic architectures

GPGPU (General Purpose Graphics Processing Unit)	Neuromorphic
GPUs only exist as part of larger overall computers; they are not amenable to the *internet of things*	Consist of single chips or minimal electronics cards; ready for integration into small devices
Cores often share memory, thereby inducing access and transfer challenges	Processors and memory are coincident and isolated from other procs/mems
High power requirements (200–500 W)	Low power requirements (.5–1 W)
Despite massive parallelism, they are still slower than NPUs above a certain problem size	Faster than GPUs on larger datasets

Neuromorphic terminology

- NM chips are inspired by the brain.
- They are essentially hardware implementations of neural networks*.
- Consequently:
  - Processing units are colloquially called *neurons*.
  - Inter-unit communication is called a *synapse*.

*There is wide diversity amongst NM chips in the kinds of neural nets they offer.

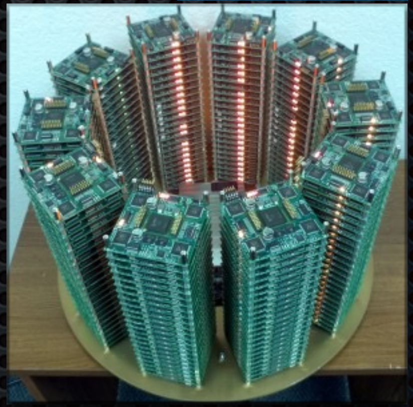
Neuromorphic applications

- Brain-computer interfaces
  - Recognizing user motor intent from EEG
- Streaming data
  - Time-based patterns
  - Medical vital-sign tracking
- Vision
  - Object detection/recognition/tracking
  - Recognizing novel (never-before-seen) objects
  - Visual anomaly detection (security)
  - Autonomous vehicles
  - Home robotics (Roomba)
  - Medical imaging & analysis
- Robotics
- Chemical sensors
- Audio & speech detection/filtering/recognition
- Natural language processing
- Internet of Things
  - Appliances
  - Wearables
  - Phones
- Distributed sensors
- Sensor fusion
- Anomaly/fraud/criminal detection
- Personal assistants

Neuromorphic options

CM1K

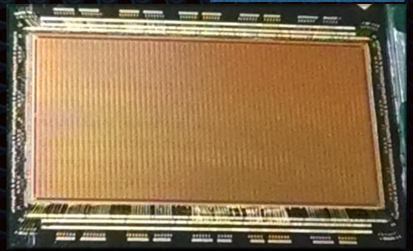
920 CM1Ks



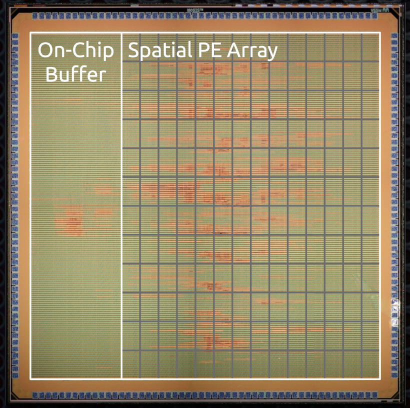
Arduino w/ Curie

- ***CM1K*** (General Vision): Simplified Radial Basis Function (RBF) network of 1K neurons. Modularly connectable to 1M neurons.
- ***Curie*** (Intel): Licenses CM1K design for 128 RBF neurons. Powers Arduino 101.
- ***TrueNorth*** (IBM, via DARPA SyNAPSE funding): 4K processors of 256 spiking neurons (1M total), 256M synapses, 70 mW. Lawrence Livermore bought a 16-chip array for $1M (16M neurons, 4B synapses, 2.5W).
- ***Eyeriss*** (MIT/DARPA): convolutional NN,168 processors, 278 mW.
- ***Zeroth*** (Qualcomm): included in Snapdragon 820 (cars, phones).
- ***BrainScaleS*** (Human Brain Project): 1 wafer of 384 chips of 512 spiking neurons & 128K synapses: 200K neurons & 49M synapses.
- ***SpiNNaker*** (HBP): 1M cores of 1K spiking neurons: 1B neurons.
- ***Darwin*** (Hangzhou Dianzi U. & Zhejiang U.): 2K neurons & 4M synapses, .84 W/MHz.
- ***KnuPath*** (KnuEdge): 256 cores capable of independent programming. Scalable to 512K chips (131M cores).

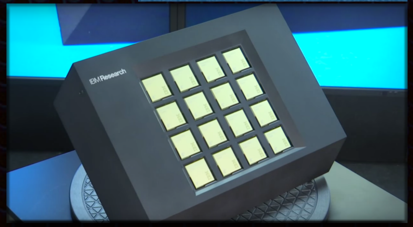
TrueNorth



Eyeriss

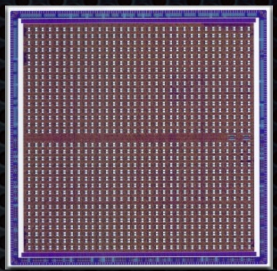


16 TrueNorths

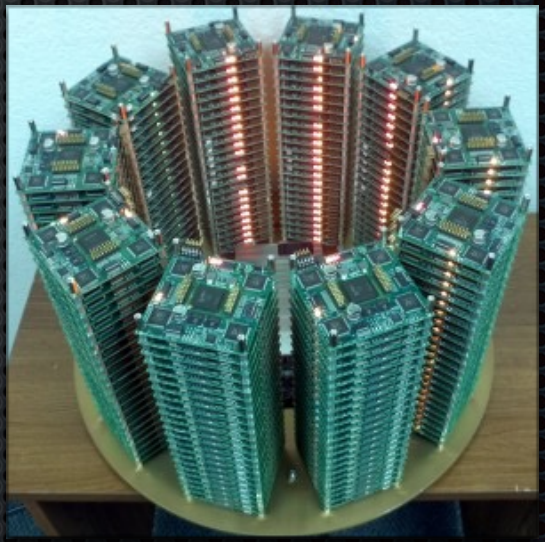


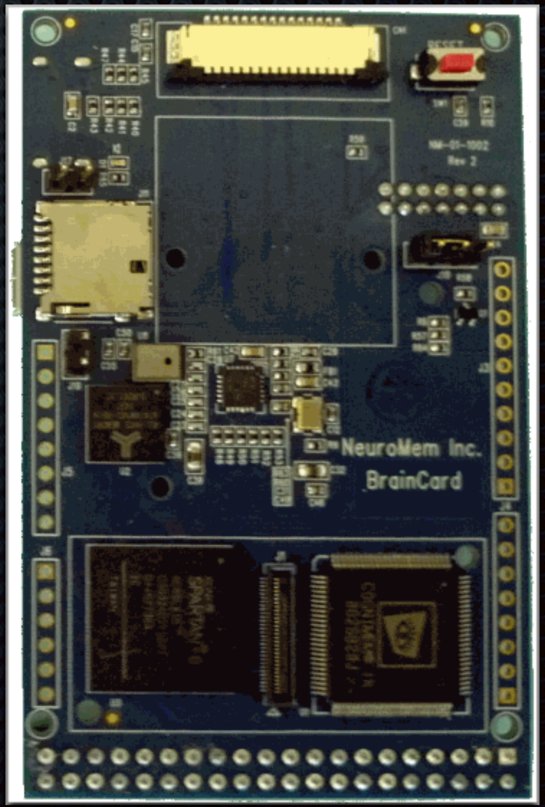
http://general-vision.com/wp-content/uploads/2014/08/NeuroMem-CM1K-296x300.png
http://general-vision.com/wp-content/uploads/2014/09/BEYOND_SC.jpg
https://cdn-shop.adafruit.com/970x728/3033-00.jpg
http://www.johnarthur.org/tnchip.jpg
http://i.kinja-img.com/gawker-media/image/upload/s---C8RY0sU--/c_scale,fl_progressive,q_80,w_800/lmpksiuoutxslnzai30e.png
http://www.mit.edu/~sze/images/eyeriss_die_photo_with_annotation.jpg

General Vision's CM1K



- 1024 RBF neurons, 127 contexts*, 32768 categories†
- Multiple chips combine up to 1M neurons
- Processes 1K–1M neurons in constant time
  - Broadcast and classification/learning in ~10μs
- .5 Watts per chip
- Integrated 1D and 2D input interfaces for seamless audio/video processing
- Accessible directly from Arduino, Raspberry Pi, Edison via the BrainCard (expandable to 9 CM1K chips)
- $80-$100 for the bare chip
- $140+ integrated into various packages:
  usb sticks, electronic boards/cards, circuit pin socket modules, etc.





*Independent sets of neurons dedicated to unrelated classification tasks

†Num categories ≤ num neurons in any given configuration

http://general-vision.com/wp-content/uploads/2014/04/CM1Kdie-300x298.jpg
http://www.cognimem.com/_images/_products/CogniStix-large.jpg
http://www.cognimem.com/_images/_products/CM1K-PGA69-large.jpg
http://general-vision.com/wp-content/uploads/2015/06/cart_BC.png

Restricted Coulomb Energy & Radial Basis Function Networks

An RBF network classifies a pattern by:

- Comparing it to a suite of previously observed and remembered patterns (aka prototypes).
- Selecting the prototype which is most similar to the new pattern (assuming any are similar enough to “fire”).
- Returning the category of the most similar firing prototype.

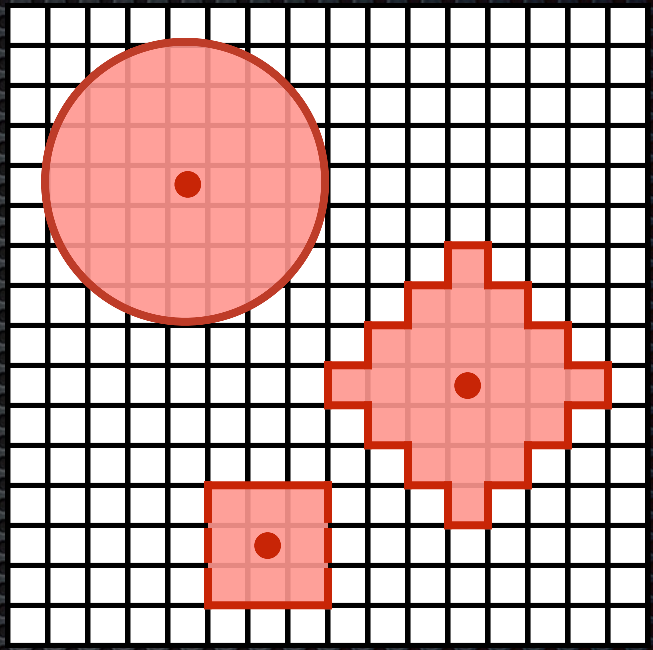
In a similar network, K-Nearest-Neighbors (KNN)*, all neurons always fire and are returned sorted by similarity to the observed pattern (from which the top ‘k’ may be selected).

RCE is a training method for RBF and KNN networks.

*The CM1K supports both RBF and KNN behavior.

Radial Basis Function Networks

- RBF networks partially cover the *feature space* with hypershapes (finite subspaces) defined by a *distance norm* metric:
  - Euclidean: hyperspheres (circles)[*]
  - Manhattan (L1): hyperoctohedrons (diamonds)[†]
  - Lsup (LMAX): hypercubes (squares)[†]
- Learned prototypes indicate the centers of hypershapes.
- A hypershape's size is indicated by an *active influence field* (AIF), a radius within which the prototype can generalize a match.
- If an observed pattern's distance (via dist. norm) falls within a neuron's AIF for its prototype, the neuron fires.
- A neuron's response, as a function of the dist. norm, is generally attenuated via a Gaussian.[‡]



[*] For simplicity, the CM1K does not support the Euclidean dist. norm.

[†] Since the CM1K calculates byte-pair-differences, it uses integer-digitized influence fields (integer-stepped diamonds and squares, as shown above).

[‡] For simplicity, the CM1K does not perform Gaussian scaling of the distance. Thus, its response is linear with the dist. norm, i.e., triangular (in fact, response ***is*** the dist. norm value).

Restricted Coulomb Energy

A method for training RBF and KNN neural networks

- Learning is incremental. Each new sample is evaluated in the following manner:
  - Novel samples (those not matching current prototypes) are stored in new neurons to increase coverage of the feature space (FS), thereby reducing *unknown* categorizations (false negatives).
  - False positives (neurons of the wrong category that fire for a given sample) shrink their AIFs to reduce subsequent false positives.
    - However, shrinking an AIF increases the risk of false negatives by reducing the FS coverage!
    - Thus, there is a tradeoff in how many training samples to learn.

CM1K *Broadcast* stage

How the CM1K classifies input signals

Patterns are ≤ 256 bytes. Each neuron holds a learned pattern.

Given an input vector, all neurons calculate their distance via one of two metrics:

- Manhattan (aka L1, sum of per-byte differences).
- Lsup (aka LMAX, max single-byte difference).

Euclidean dist. norm and subsequent Gaussian attenuation are not offered due to computational overhead.

Two classifiers are offered:

- K-Nearest Neighbors: All neurons fire.
- Radial Basis Function (RBF) based on Restricted Coulomb Energy (RCE):
    - All neurons hold an *active influence field* (AIF), i.e., a firing threshold.
    - All neurons for which the distance < AIF fire.

All firing neurons are automatically sorted by distance so the best match can be immediately retrieved. Alternatively, the entire firing set may be investigated in sorted order.

CM1K *Learn* stage

How the CM1K learns patterns (via RCE)

Following a broadcast, the network can optionally learn the previous input vector (assuming its ground truth is available) in the following manner:

If no neurons fired, a new neuron is recruited and assigned the previous input as its pattern with an AIF of *PresetMaxAIF*.

If some neurons fire:

- Any neurons that fire with the correct category do nothing.
- Any neurons that fire with the wrong category shrink their AIFs enough to exclude the input vector.

If no neurons fire correctly, a new neuron is recruited and assigned an AIF of min(*PresetMaxAIF*, all firing AIFs).

RBF vs. perceptron neural nets

- Perceptrons don’t need many neurons because they divide the feature space (FS) into vast half-spaces along hyperplanes (green).
  - (Multilayer perceptrons *can* carve the FS into finite regions by combining neurons (often via multiple layers (Fig. 2)), but they don’t have to (Fig. 1).
- RBFs divide the FS into finite volumes (red & blue).
- Consequently, RBFs often require many training samples and learned prototypes to cover the FS and avoid *unknown* classifications (Fig. 1), but it depends on the category distribution in FS; *in theory* they can be more efficient than perceptrons (Fig. 2)!
- For this reason, RBFs have traditionally been infeasible in classic von Neumann computers via simulation; they are too large to process serially.
- NM chips to the rescue: hardware parallelization!

Fig. 1

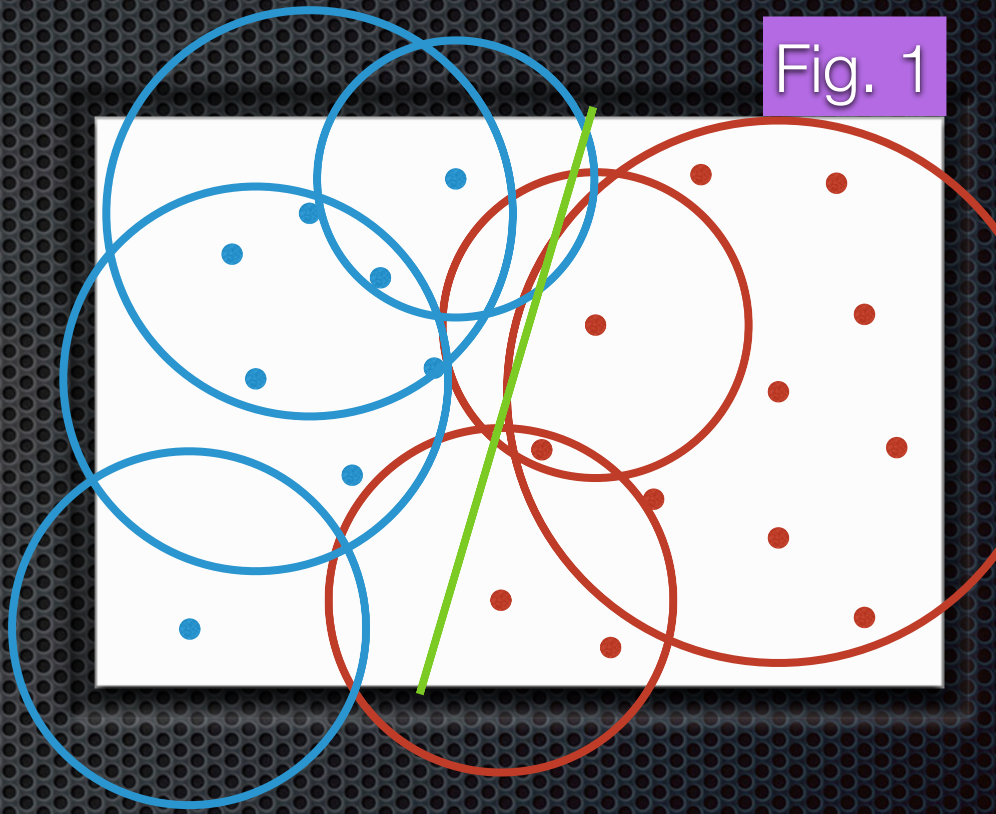
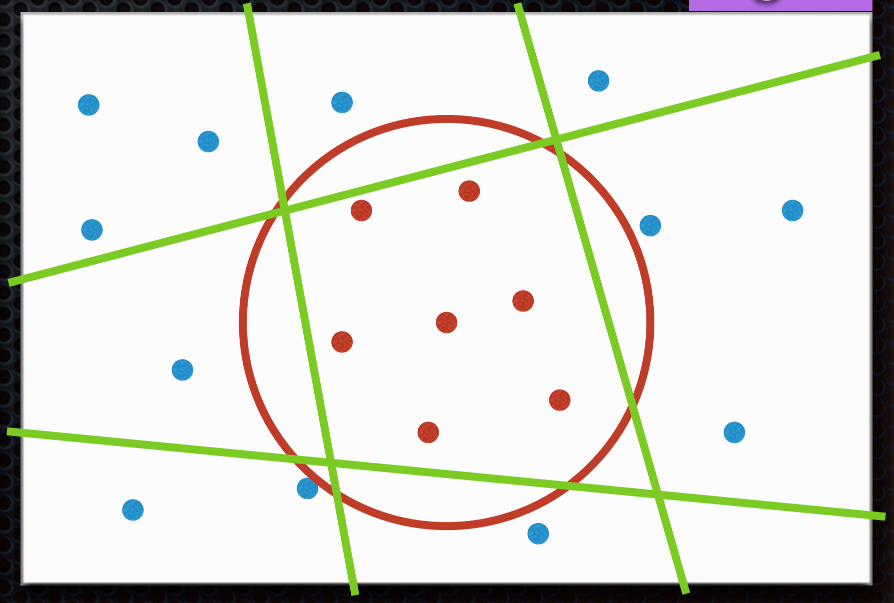


Fig. 2



CM1K contexts

Neurons have a *context* in the range 1–127.

Contexts isolate the neurons into groups so that different kinds of patterns can be learned and classified.

For example, a conglomerated data source may consist of:

- Video stream
- Audio stream
- Temperature/pressure/tilt/accelerometer/etc. sensors
- Various metadata

Using CM1K contexts for multilayer RBF networks

Contexts can be used to create multilayer RBF networks in the following way:

The input vector is learned and classified with some context (e.g., 1).

The sorted list of firing tuples [(category, distance),…] represents a new input layer to be learned and classified by a different set of neurons using a different context (e.g., 2).

The output of the final layer indicates the network's overall classification.

Multilayer networks obviously increase classification time linearly with the number of layers since they are iterative.

Python CM1K Emulator

I have written a CM1K emulator in Python to investigate the chip's classification performance.

- http://keithwiley.com/software/CM1K_emulator.shtml
- https://github.com/kebwi/CM1K_emulator

Obviously, the emulator suffers from serial processing, but the purpose is to investigate the learning and classification of the CM1K's simplified RBF network.

The emulator can simulate low-level functionality (the CM1K's data bus and registers), but it can also simulate at a high-level, disregarding procedural minutiae but yielding the same end result.

Evaluation datasets

- *MNIST handwritten digits*
  (essentially black & white image data*)
- *AT&T Faces*
  (grayscale photographic image data)
- *Iris*
  (numerical data)
- *Mushroom*
  (nominal data)

*The NIST data is truly B&W, but MNIST is resampled from NIST, resulting in interpolated gray levels.

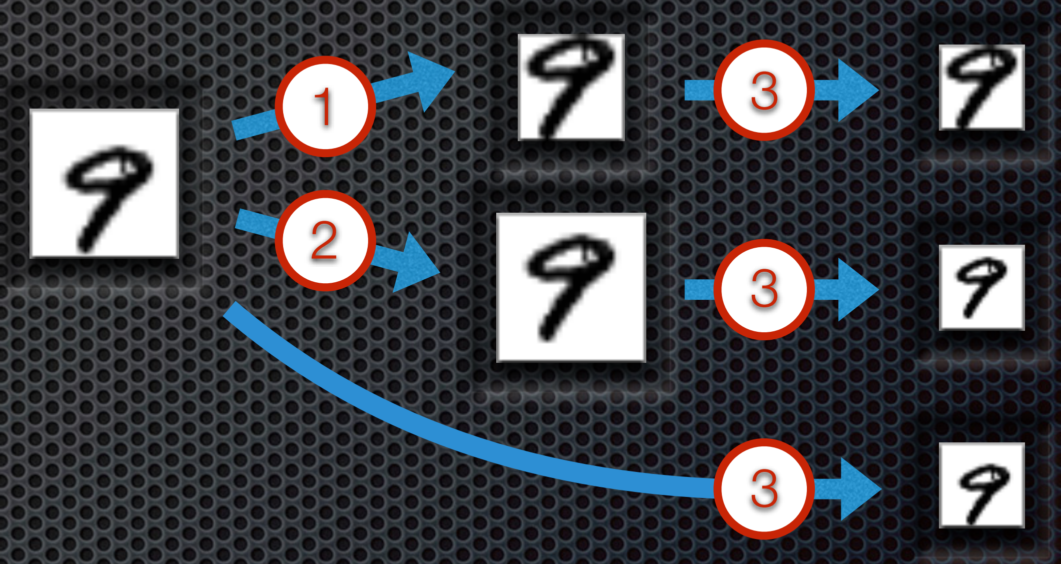
MNIST handwritten digits on CM1K emulator

- 70,000 handwritten digits 0–9
    - Chance odds of 10-class classification: 10%
- Divided into a 60k train set and 10k test set
- 250 “writers” in each set (disjoint between train/test)
- 8-bit grayscale
- 28×28 pixels

MNIST on CM1K emulator

CM1K data preparation

- In the hope of improved image registration, optionally either:
  - Crop* ①
    (original bounding box is 20×20†)
  - Bounding-box-center ②
    (originally center-of-gravity-centered)
- Resample to 16×16 to fit in the CM1K’s 256-byte pattern length. ③



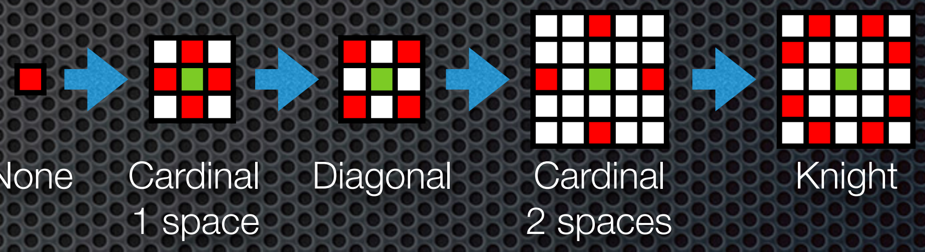
*Results on the following slides represent the “cropping” method since that approach yielded the best performance.

†The previous slide described the images as 28×28. The MNIST images were generated by first scaling to a 20px Bbox, then placing within a 28px frame such that centers-of-gravity reside at the center. In other words, the MNIST data are already COG-centered, but since Bbox-centering might offer better modeling performance, I tried both.

MNIST on CM1K emulator

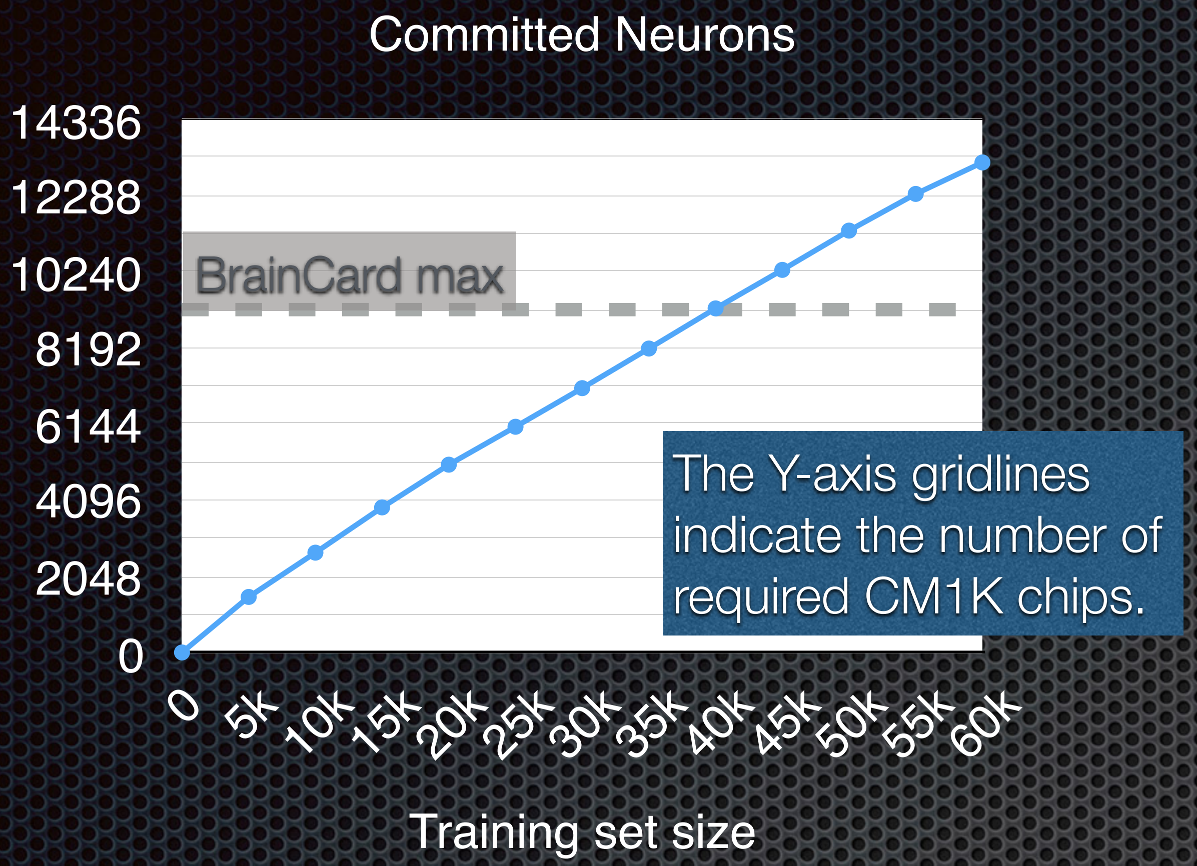
Evaluation parameters

- Various training set sizes: 5k–60k
- When classification is *unknown* (no neurons fire), increment over increasingly distant translations to find a registration with the network's learned prototypes.



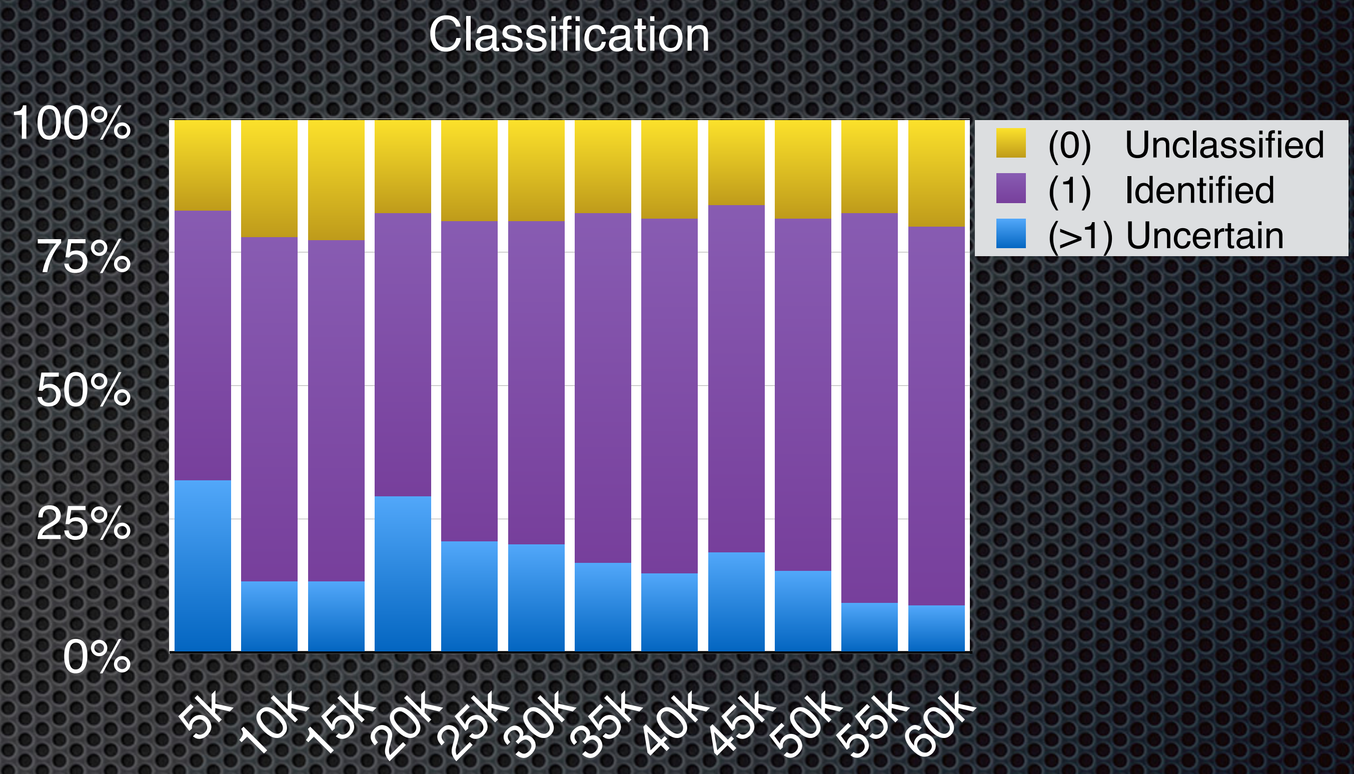
- Euclidean distance function
- Mode-voting classification when *uncertain* (i.e., when neurons of varying category fire)

MNIST on CM1K emulator

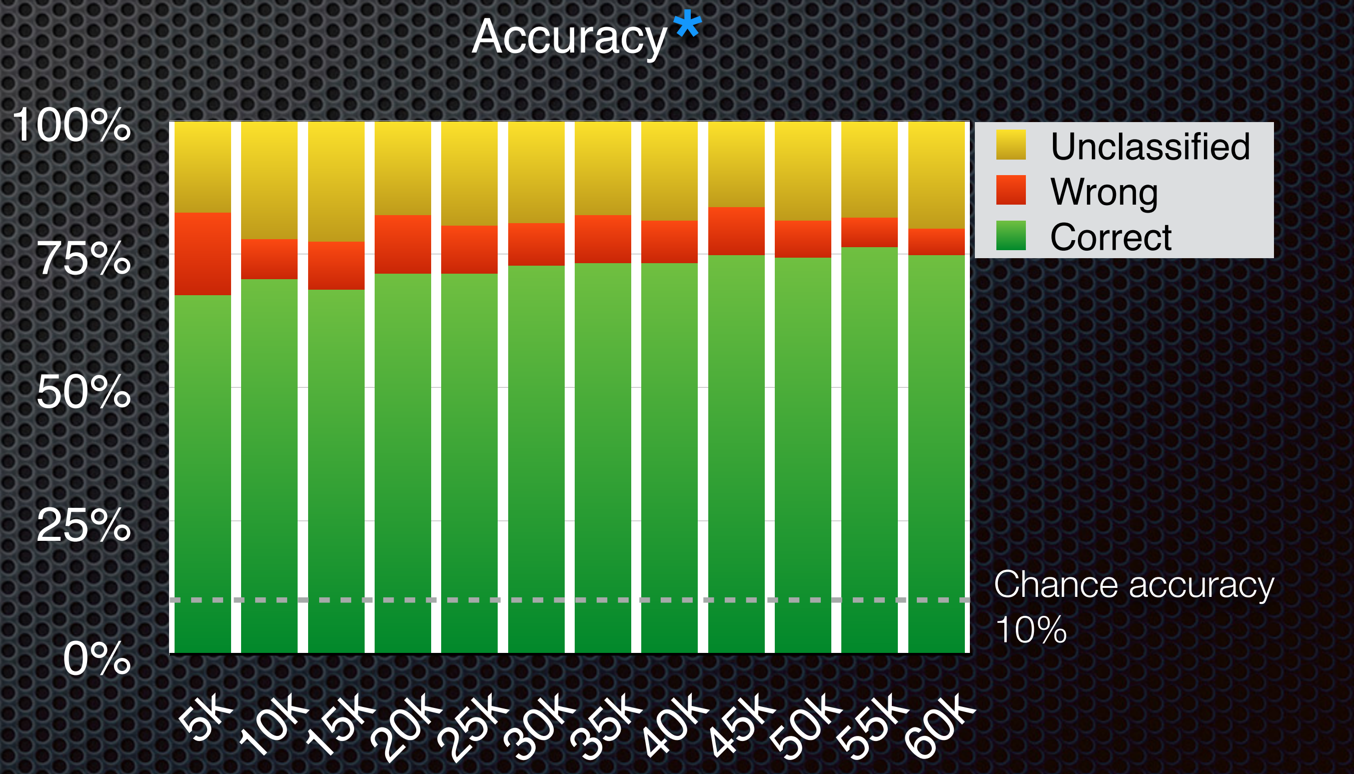


Committed Neurons

The Y-axis gridlines indicate the number of required CM1K chips.

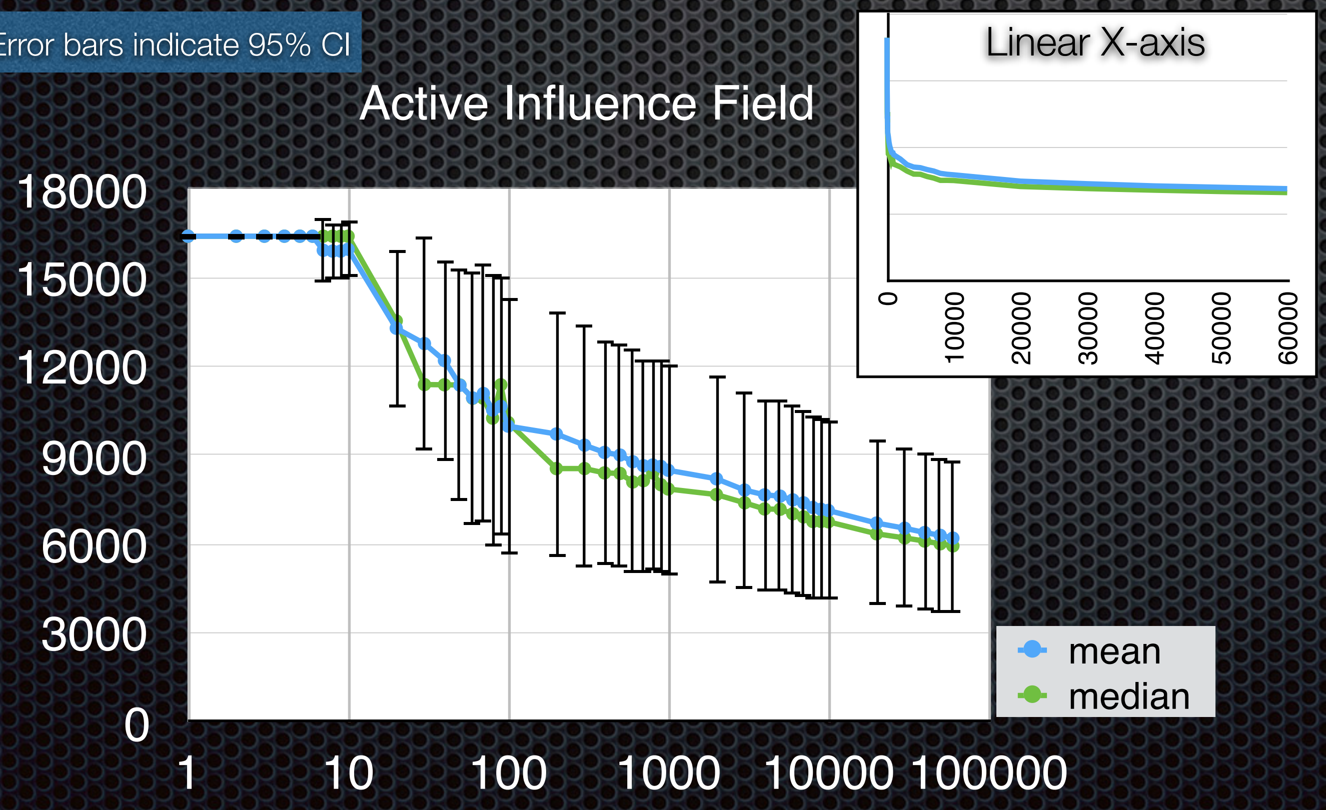


Classification



Accuracy*

Error bars indicate 95% CI



Active Influence Field

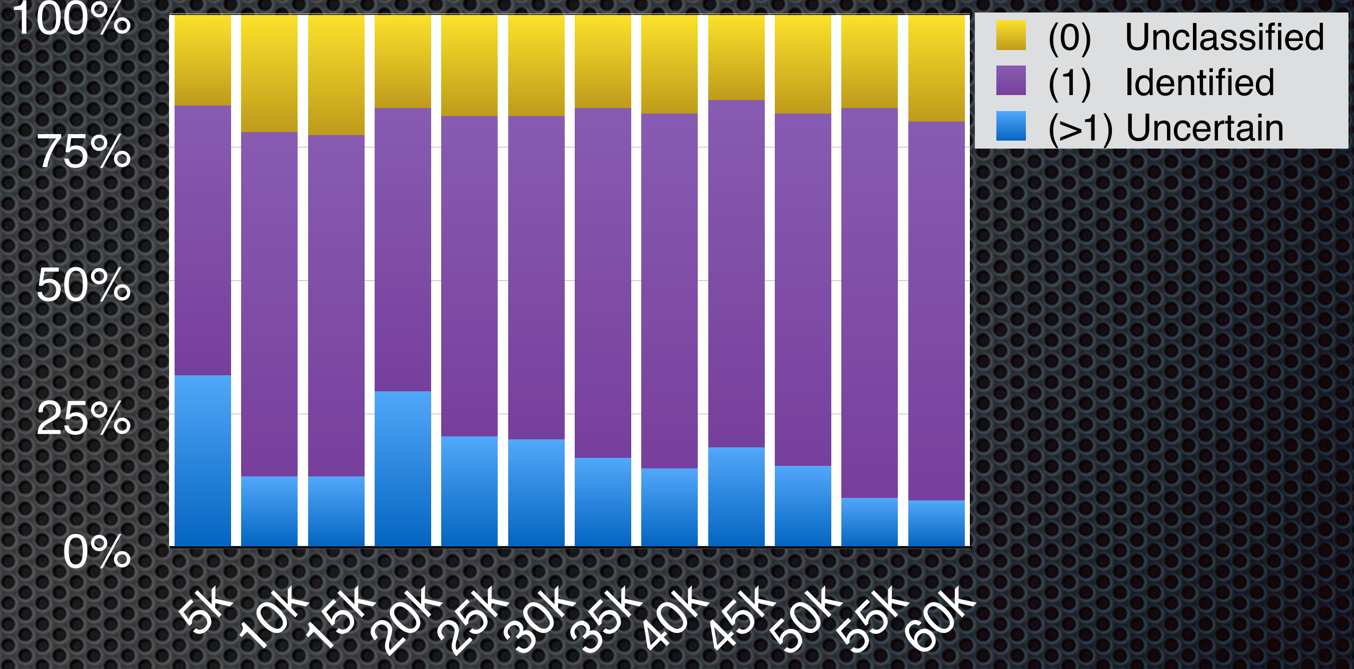
*MNIST classification can only exhibit true positives (**correct**), false positives (**wrong**), and false negatives (**unclassified**). True negatives have no meaning for MNIST.

MNIST on CM1K emulator

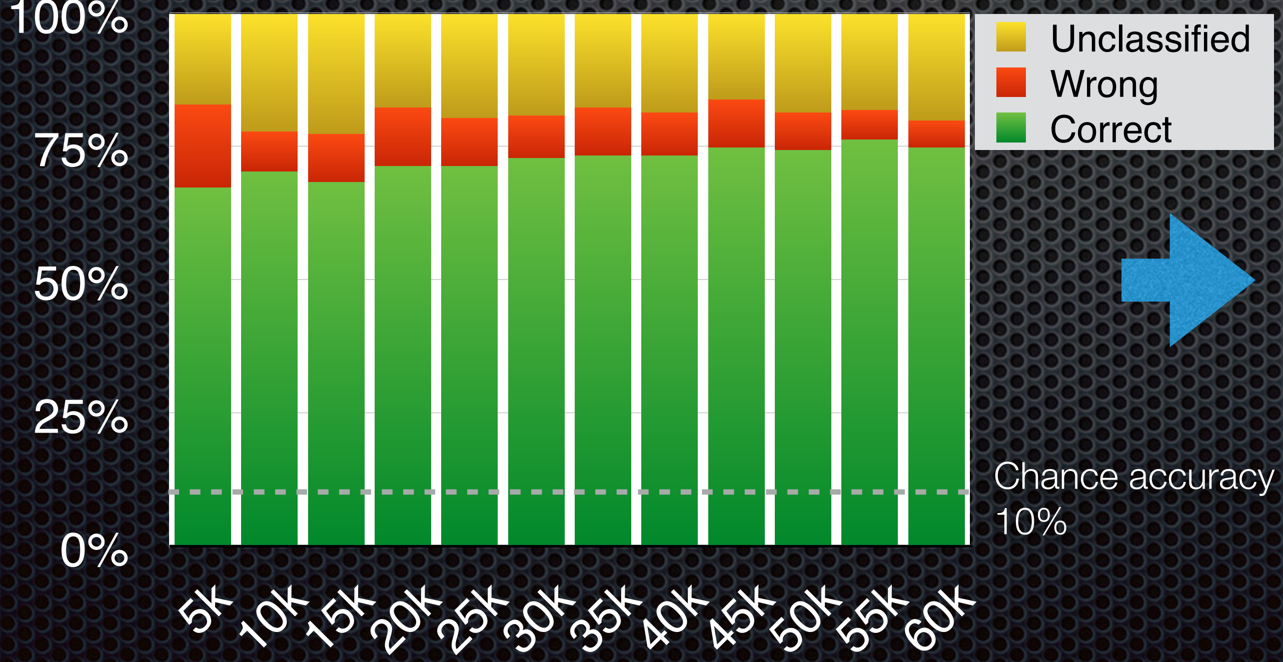
Combined accuracy and classification distribution

What is the distribution of identified and uncertain classifications within each of correct and wrong predictions?

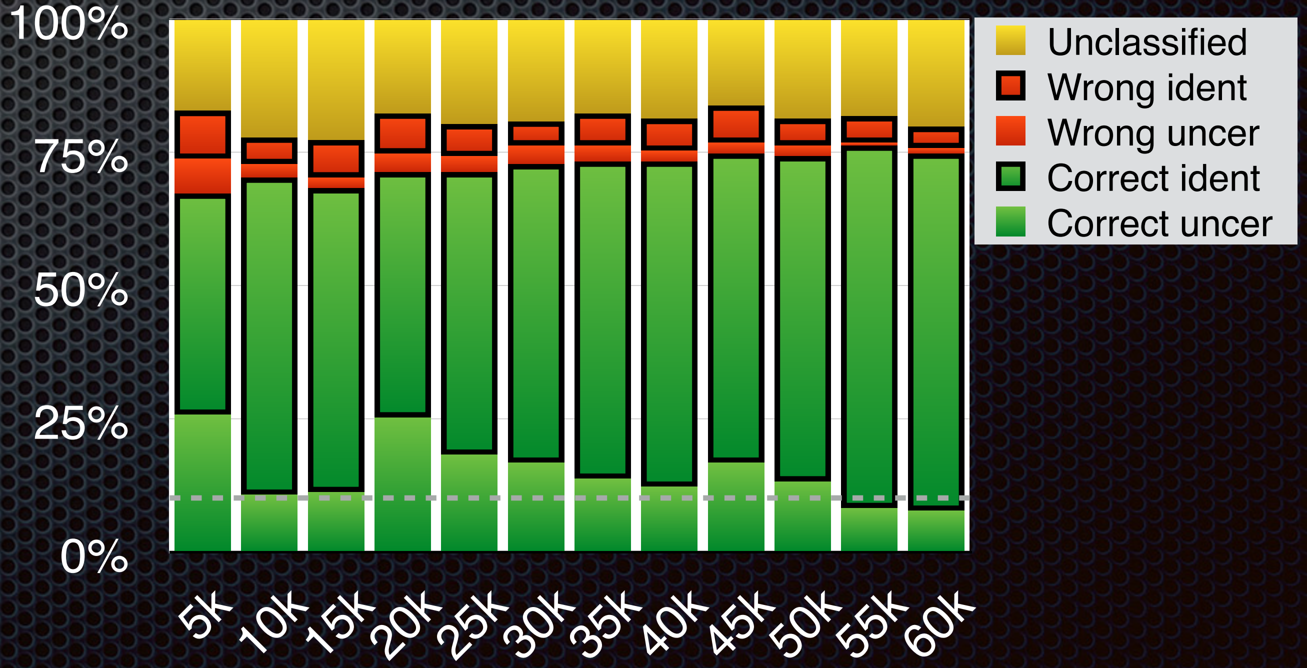
Classification



Accuracy



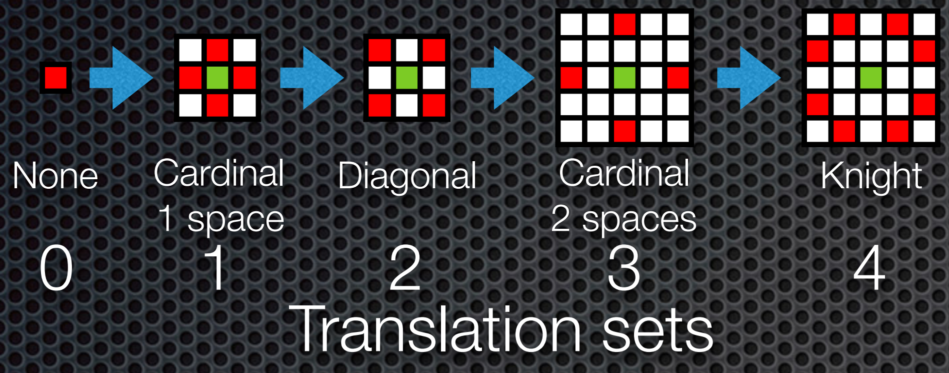
Accuracy & Classification



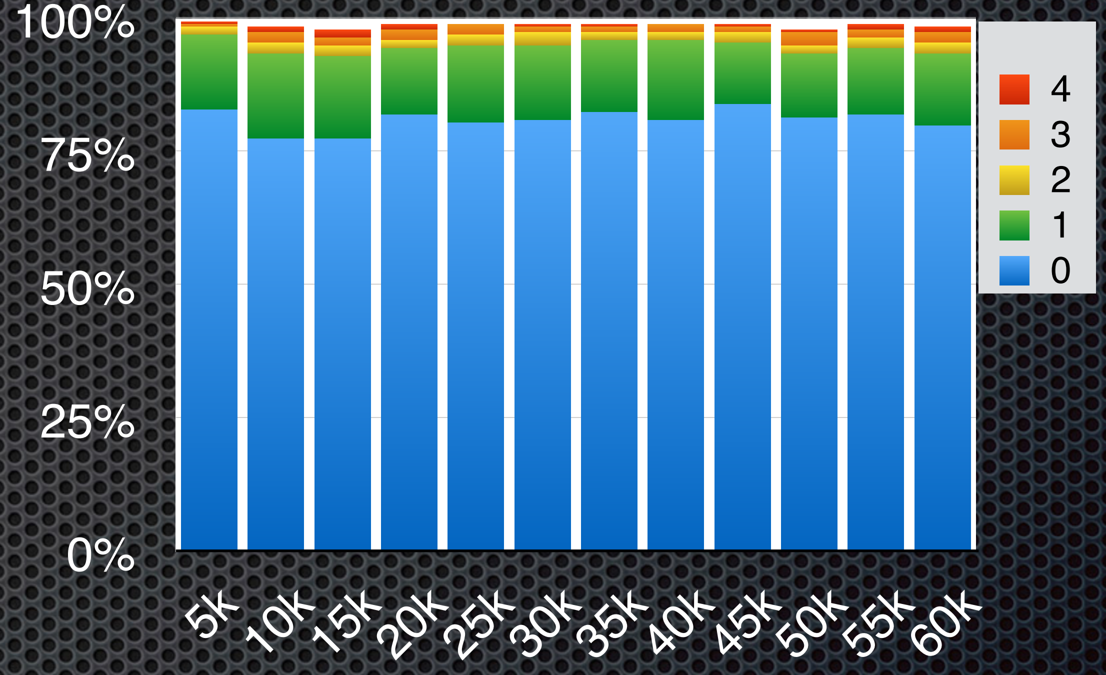
MNIST on CM1K emulator

Translating test images to improve registration

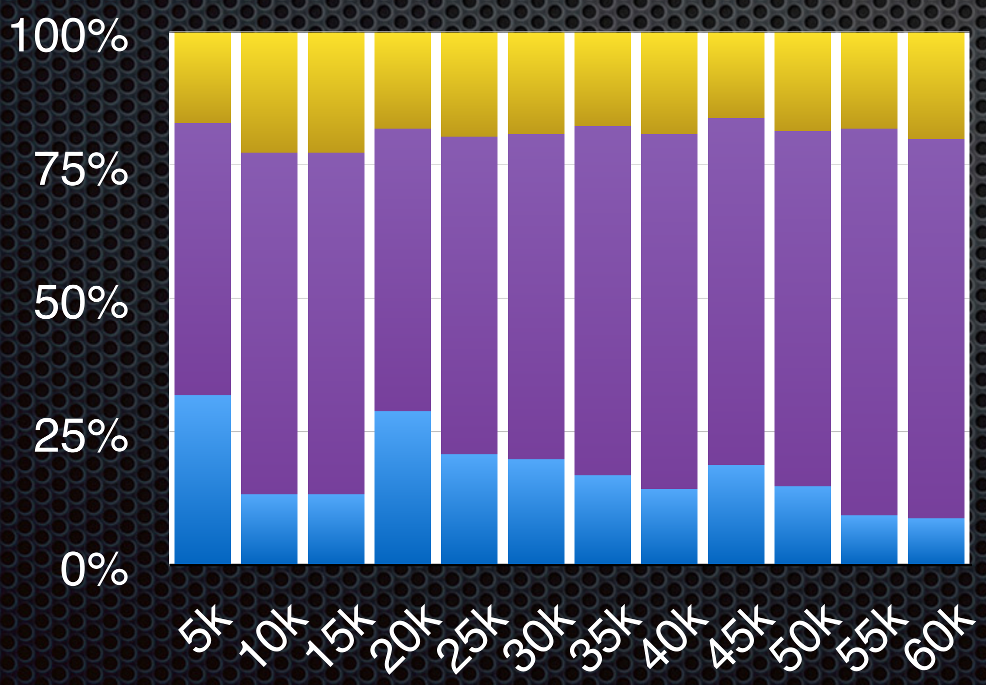
When a classification is *unknown* (no neurons fire), increment over increasingly distant translations to find a registration with the network's learned prototypes.


None
0
Cardinal 1 space
1
Diagonal
2
Cardinal 2 spaces
3
Knight
4
Translation sets

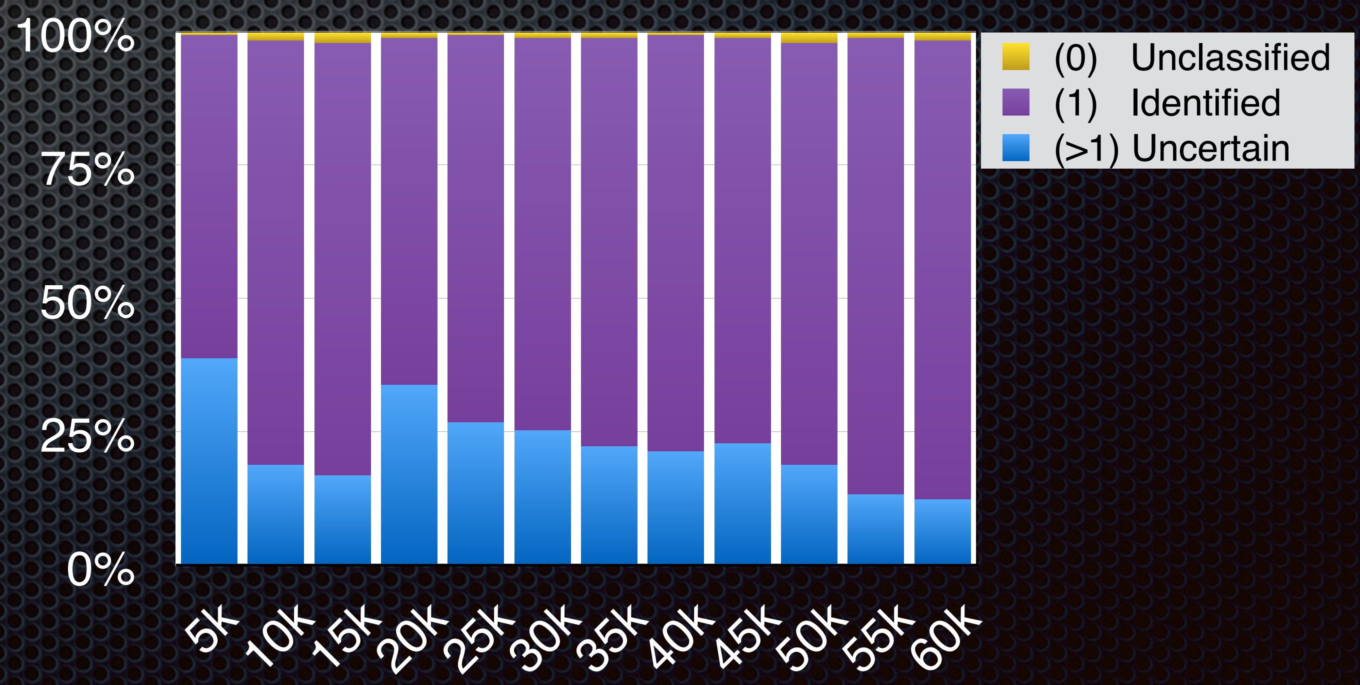
Translation set in which a classification was made



Classification w/o translation sets



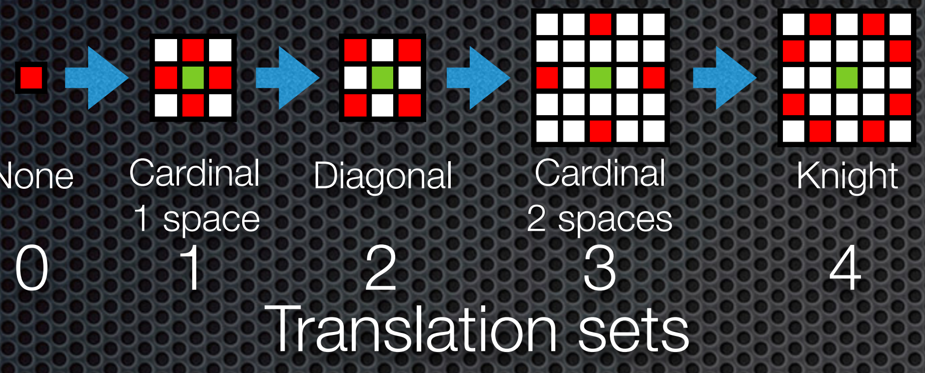
Classification w/ translation sets



MNIST on CM1K emulator

Translating test images to improve registration

When a classification is *unknown* (no neurons fire), increment over increasingly distant translations to find a registration with the network's learned prototypes.



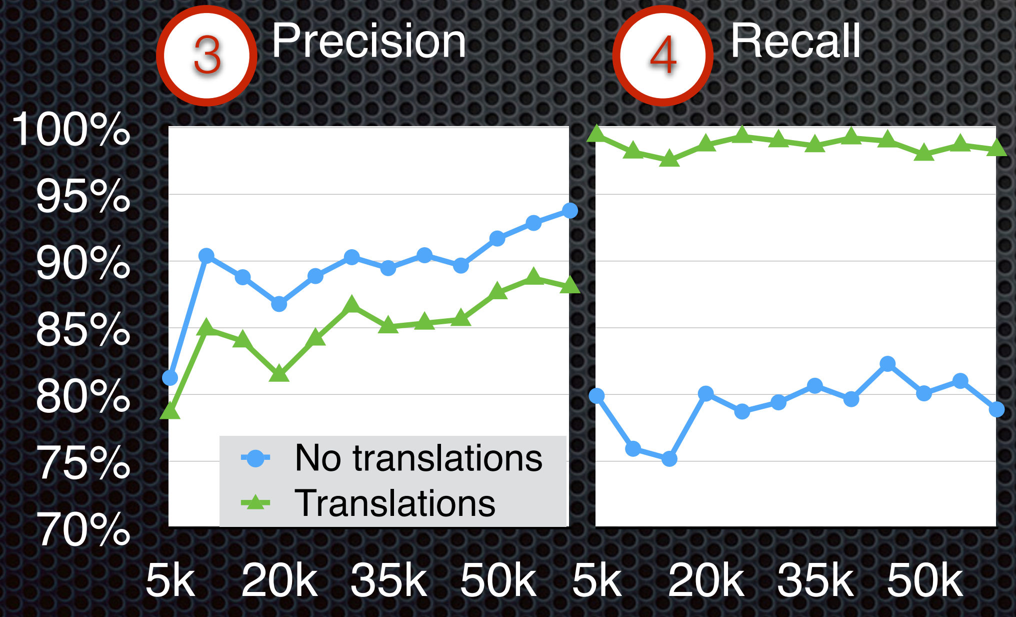
∴Considering translations of an image:

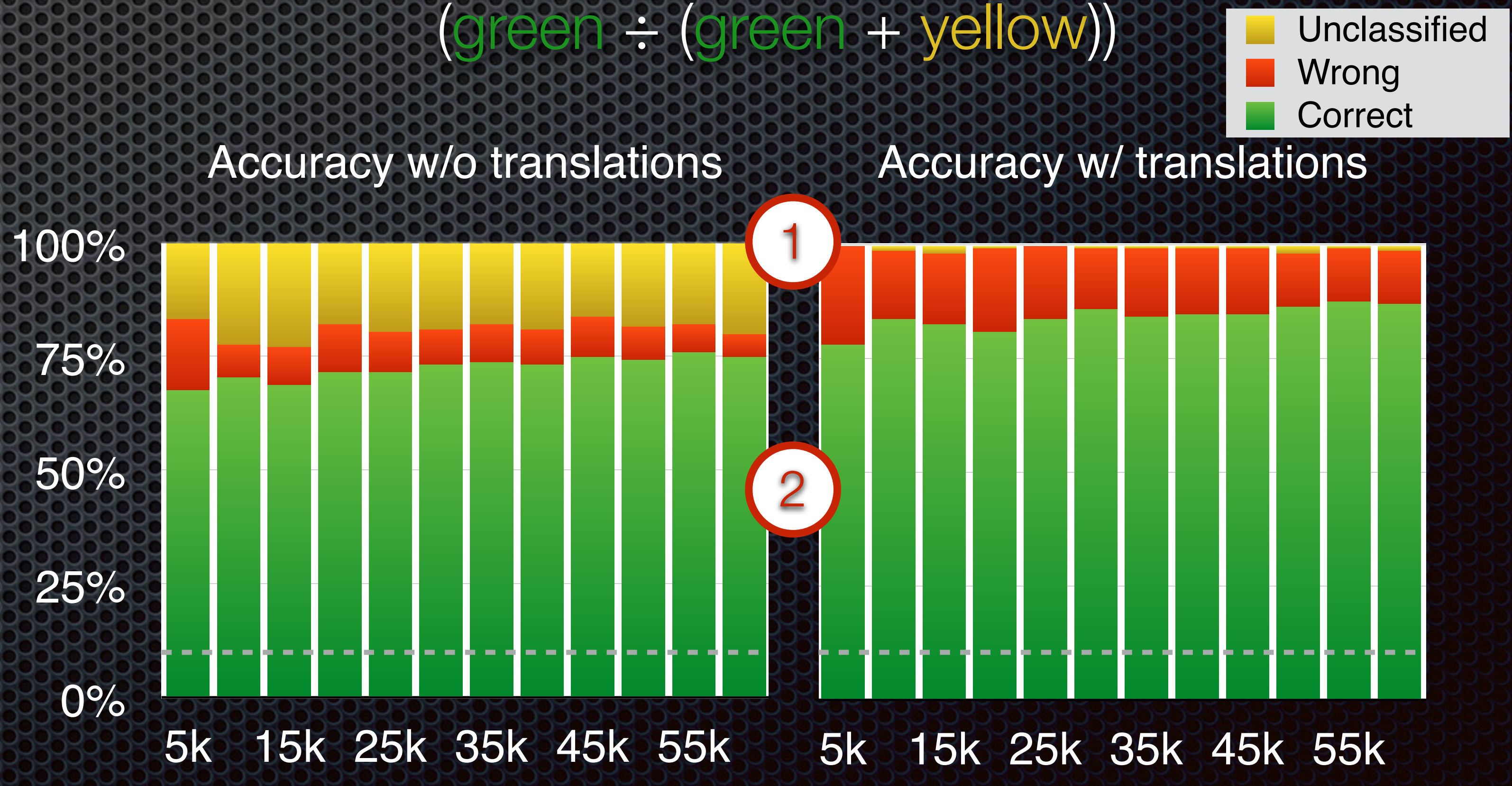
1 Virtually eliminates unclassifications (yellow bars)

2 Improves accuracy (green ÷ total) (or simply green bar height)

3 Curiously, it hurts precision (green ÷ (green + red))

4 Improves recall (green ÷ (green + yellow))





MNIST on CM1K emulator

Comparing CM1K’s L1 to Euclidean distance function

Remember: the RCE/RBF model uses a *PresetMaxAIF* parameter, the max. possible AIF assigned to a newly recruited neuron.

Any given neuron’s AIF is subsequently shrunk during training to reduce false positives (i.e., when a neuron of category 1 fires for a sample of category 2).

The CM1K’s default *PresetMaxAIF* is 16384* (configurable), i.e., ~25% of the max. sample-to-prototype distance (255 max-byte-dif × 256 bytes = 65280).

A Euclidean distance norm has a max. distance of $\sqrt{(255^2 \times 256)} = 4{,}080$. So, if there is some rationale to the 25% default, then perhaps *PresetMaxAIF* for a Euclidean dist. norm should be ~25%: 1024.

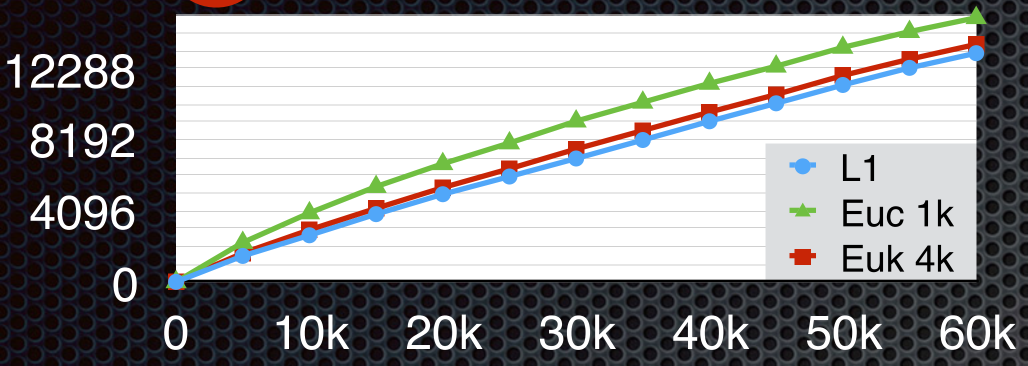
So, I tested with both 1024 and 4096.

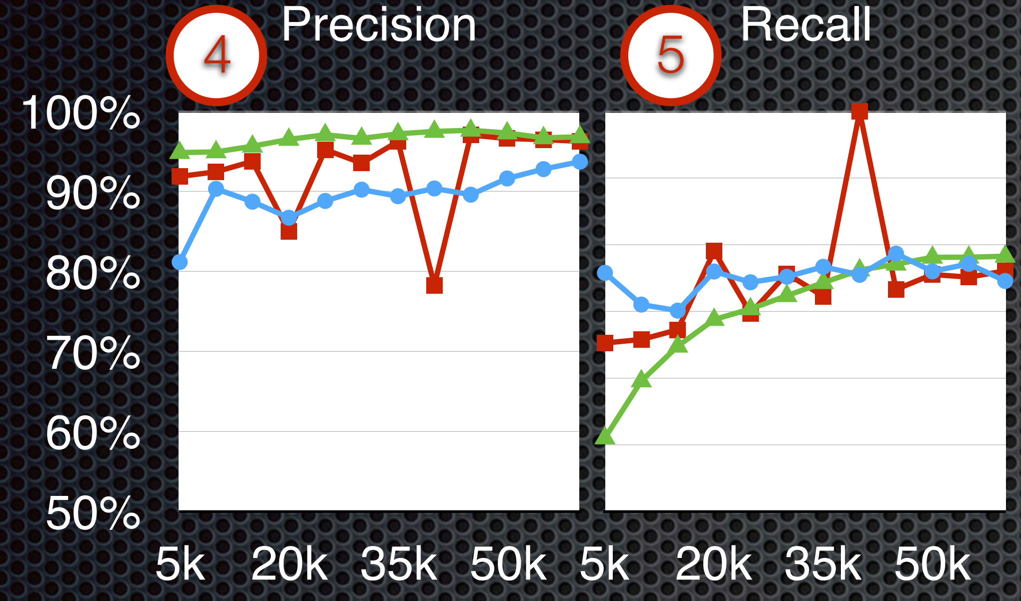
*Curiously, switching the CM1K to the Lsup norm doesn’t alter the *PresetMaxAIF* parameter even though Lsup’s max. sample-to-prototype distance is a measly 255.

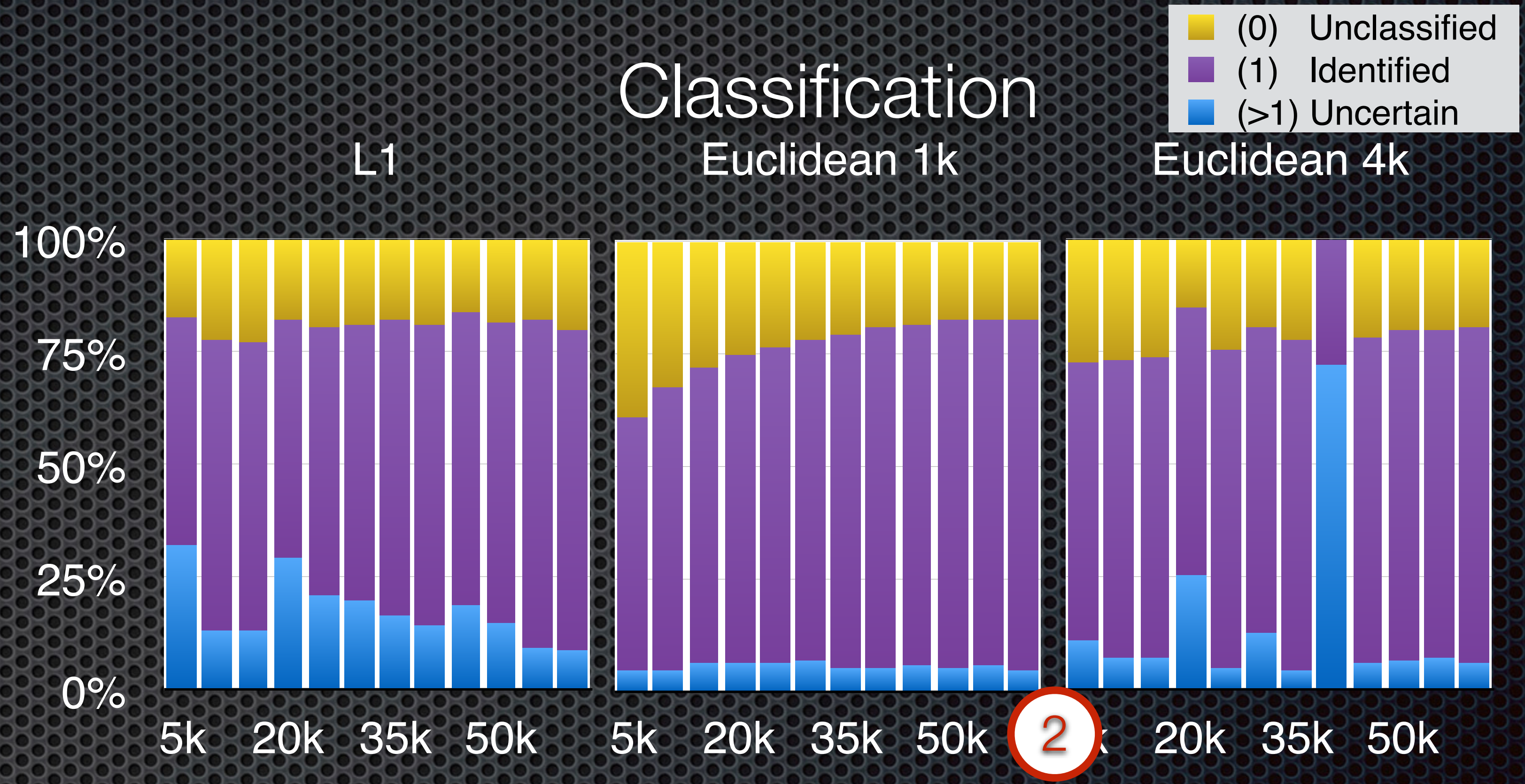
MNIST on CM1K emulator

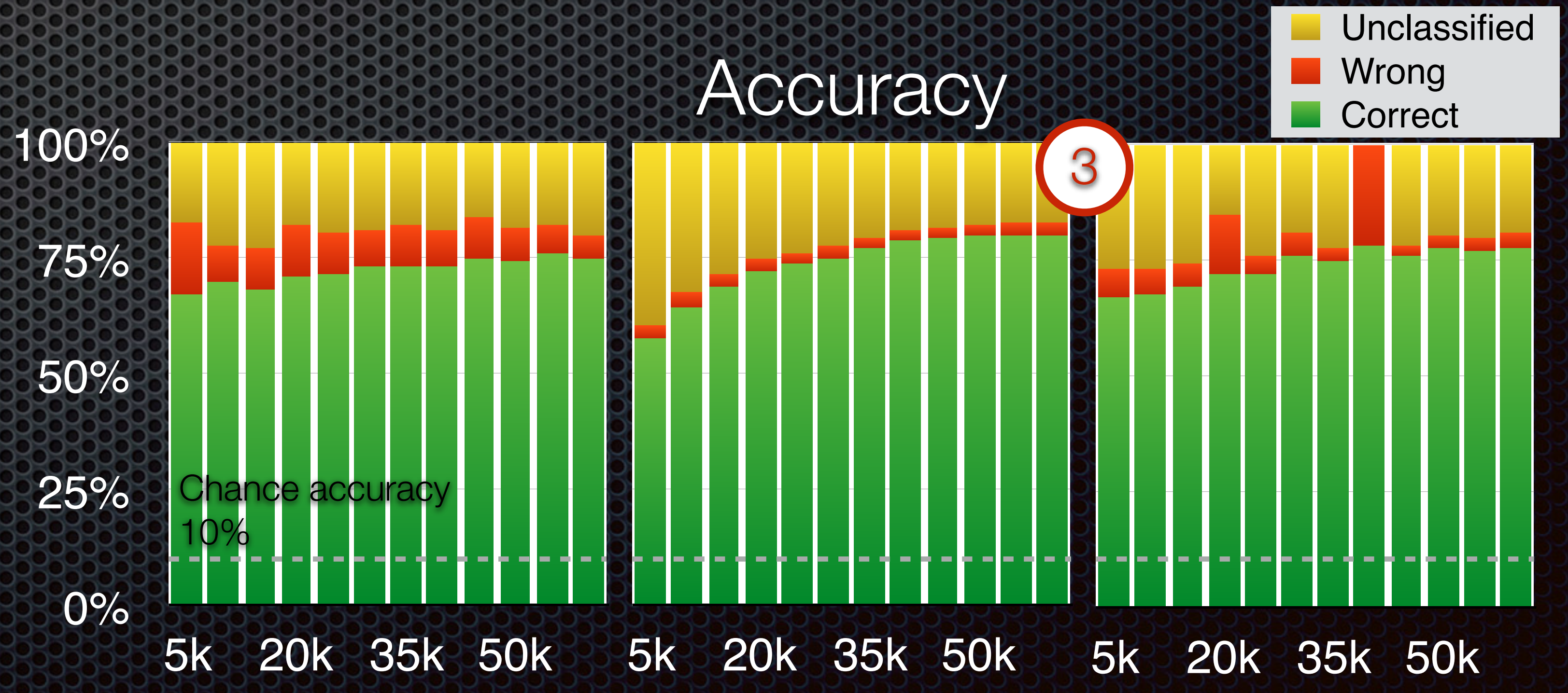
Comparing CM1K’s L1 to Euclidean distance function

1 Committed Neurons









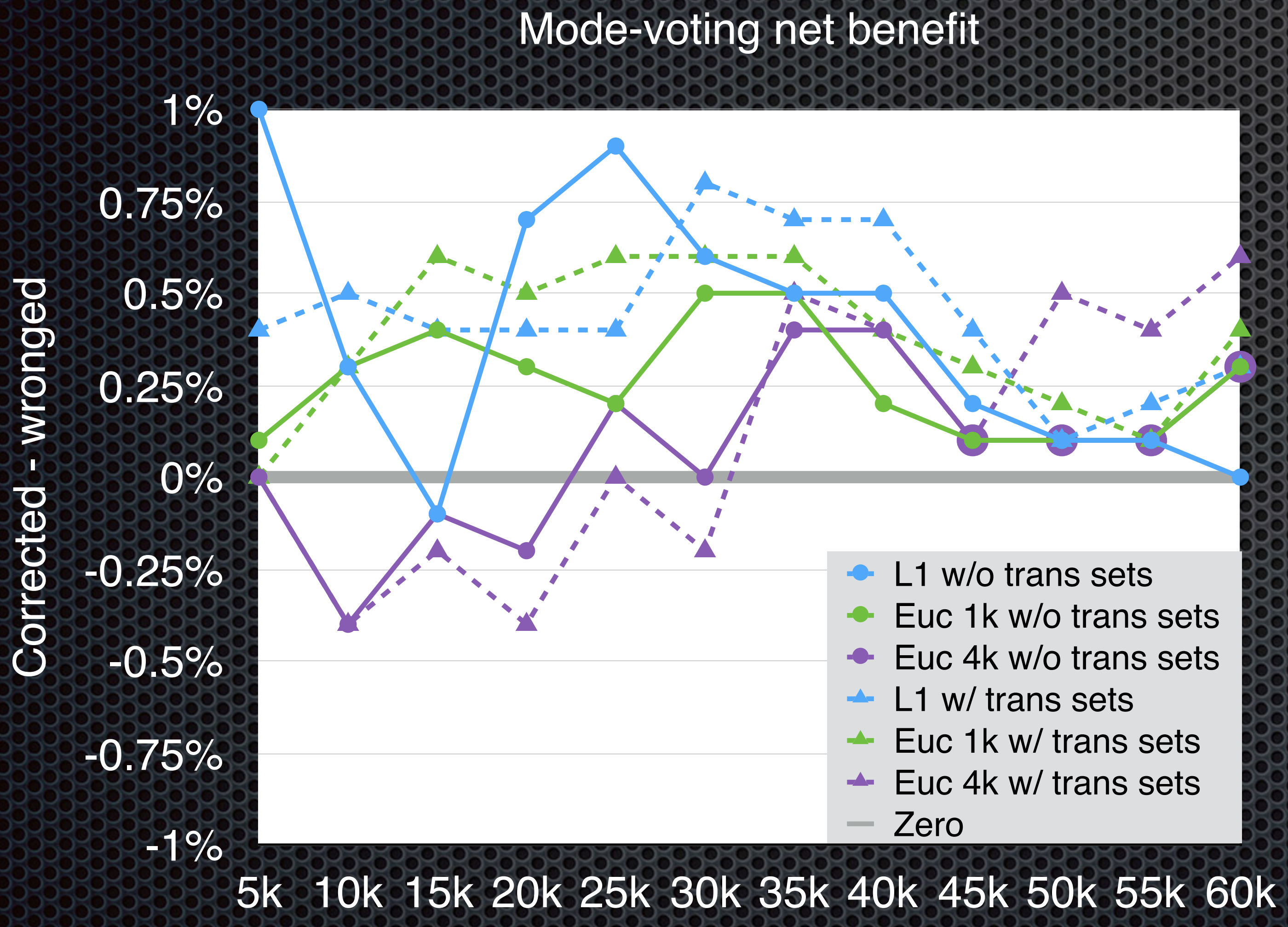
∴A Euclidean distance function:

1. Recruits more neurons
2. Has fewer uncertain classifications
3. Has fewer wrong classifications
4. Improves precision
5. Has little effect on recall

MNIST on CM1K emulator

Mode-voting to resolve *uncertain* classifications

Whenever multiple neurons fire with varying categories, take the most-voted category (mode-voting), not the strongest firing neuron (winner-take-all). If the mode is not unique, resort to winner-take-all.



Note that voting risks "wronging" a correct winner-take-all classification.

A negative plot value implies that more correct winner-take-all values were "wronged" by mode-voting than vice versa.

∴The benefit of resolving uncertain classifications by mode-voting instead of winner-take-all is consistently positive but negligible.

MNIST on CM1K vs. MNIST by other models

- MNIST results are usually reported in terms of *error rate* (all incorrect classifications, in which I include RBF *unclassifications*). The lowest error rate I achieved with the CM1K emulator was 12.2%*.

- Yann LeCun maintains the primary MNIST website, which includes numerous modeling results. http://yann.lecun.com/exdb/mnist/

- Error rates across 69 models vary (obviously), but are generally 1–2% with lows ~.2% and highs ~7–8% (and 12.0% the worst), including:
  - An RBF: 3.6% error
  - A few 16px subsampled examples:
    - KNN: 1.1% error
    - Convolutional net: 1.7% error

- Why is the CM1K underperforming so much?

*Cropping, translation sets, mode-voting

MNIST on CM1K vs. MNIST by other models

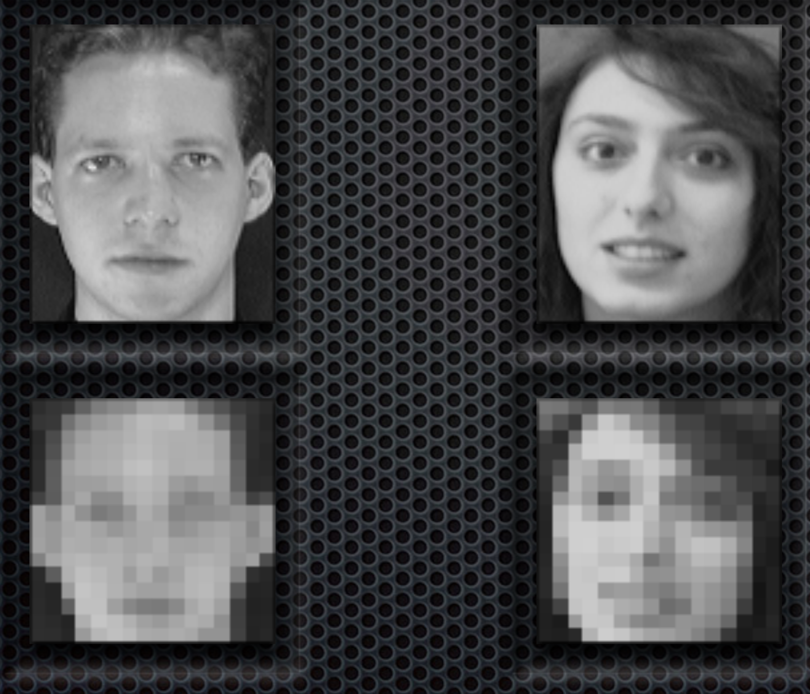
- I only tested with the first 1000 images of the MNIST test set. If that subset is more difficult to classify than the other 9000, then my results suffered unfairly—but this seems doubtful.

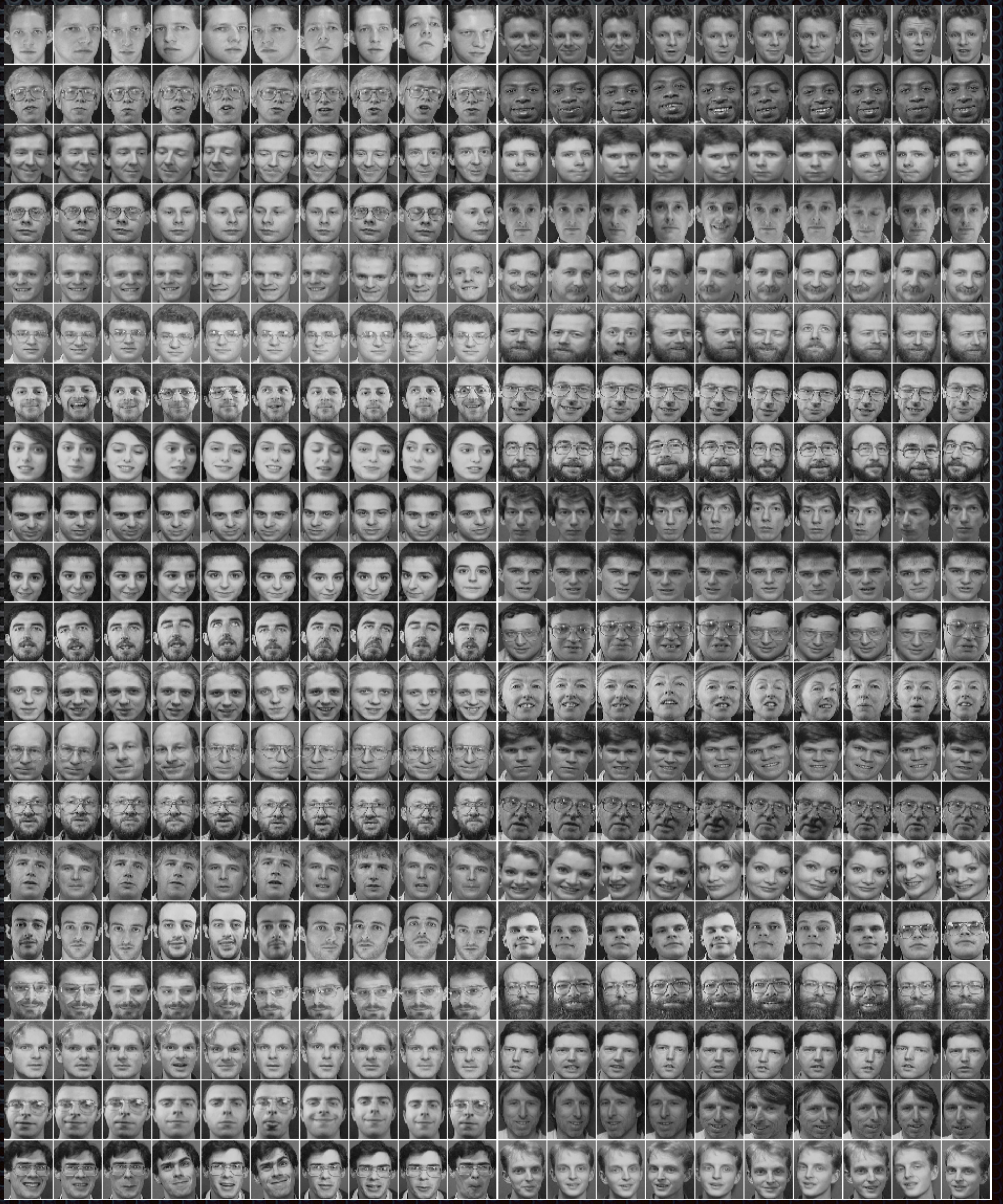
- RBF network described by LeCun:
  - 1000 Gaussian RBF neurons
  - 28px input resolution
  - 2nd layer of 1000 neurons evenly divided into 100 per category
  - Training by K-means, not RCE
  - Weighted connections

Perhaps CM1K performance could be improved via multiple layers, or by tiling to accommodate greater image resolution, or by giving COG-centering another chance…
…but that will have to wait for another day.

AT&T Faces on CM1K emulator

- 40 people, 10 pictures each
  - Chance odds of 40-class classification: 2.5%
- 8-bit grayscale
- 92×112 pixels
  - Cropped to 92×92
  - Downsampled to 16×16

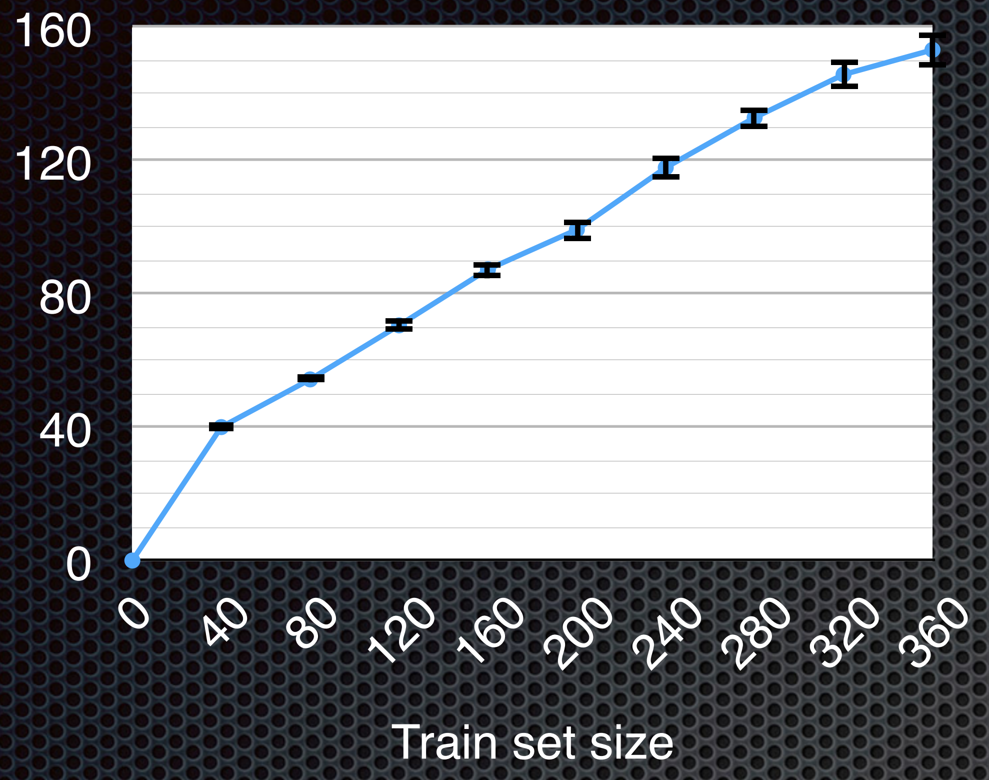




This is the entire dataset, albeit scaled to fit.

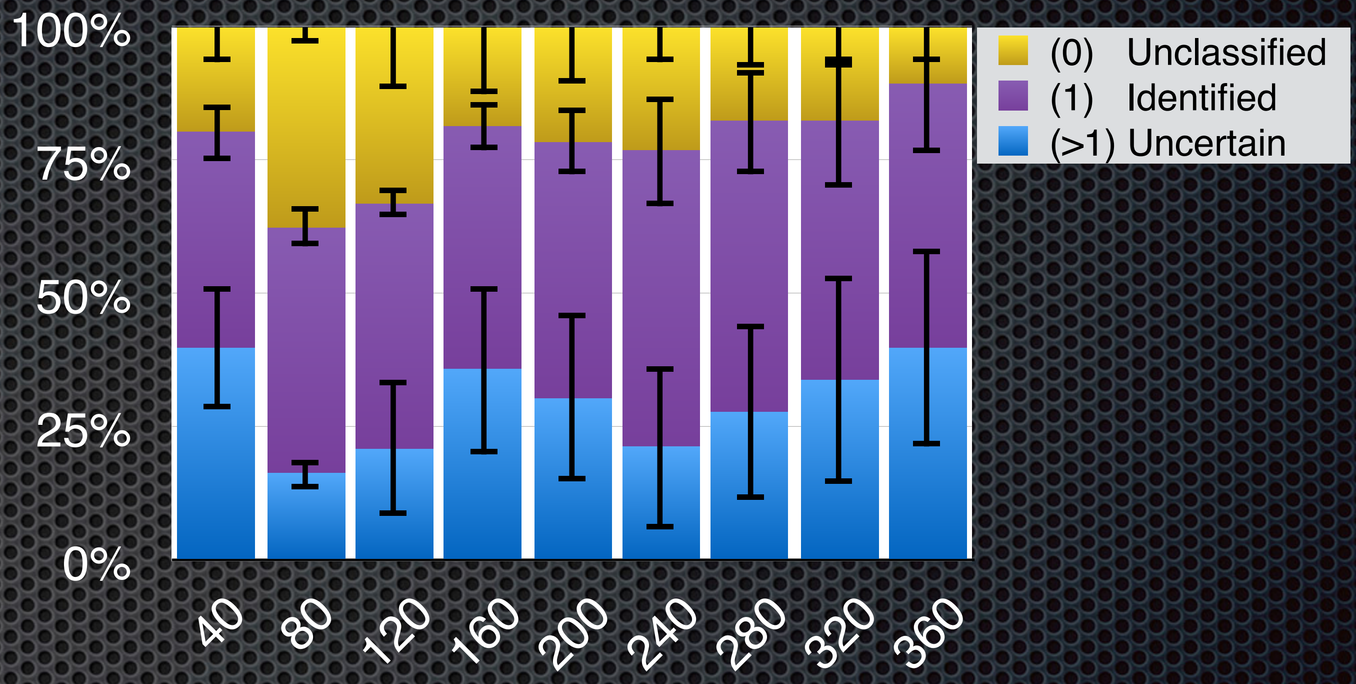
Faces on CM1K emulator

Committed Neurons

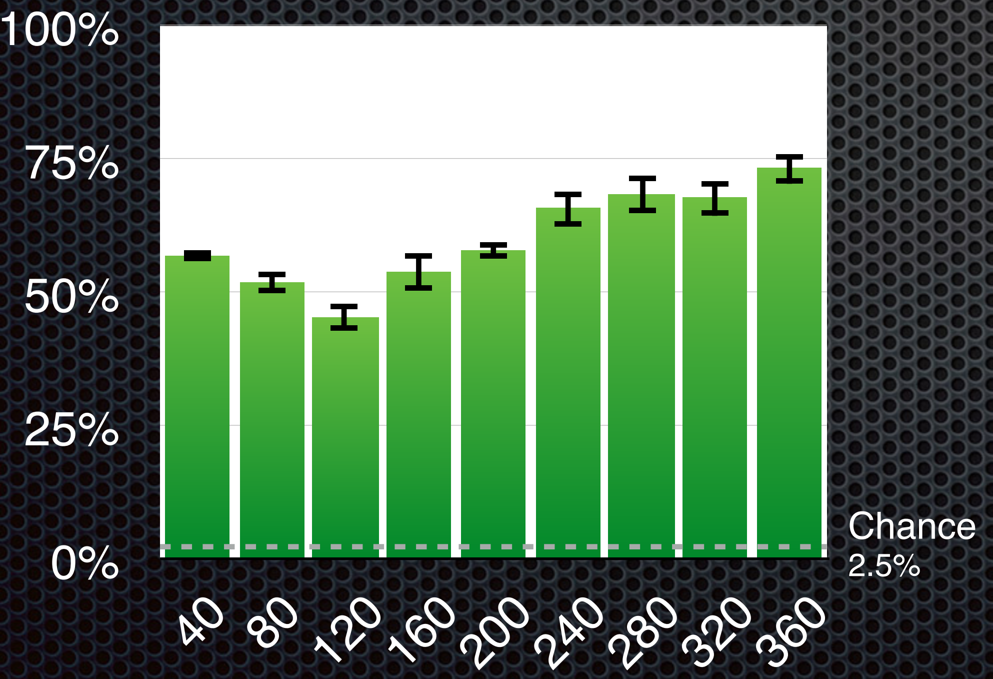


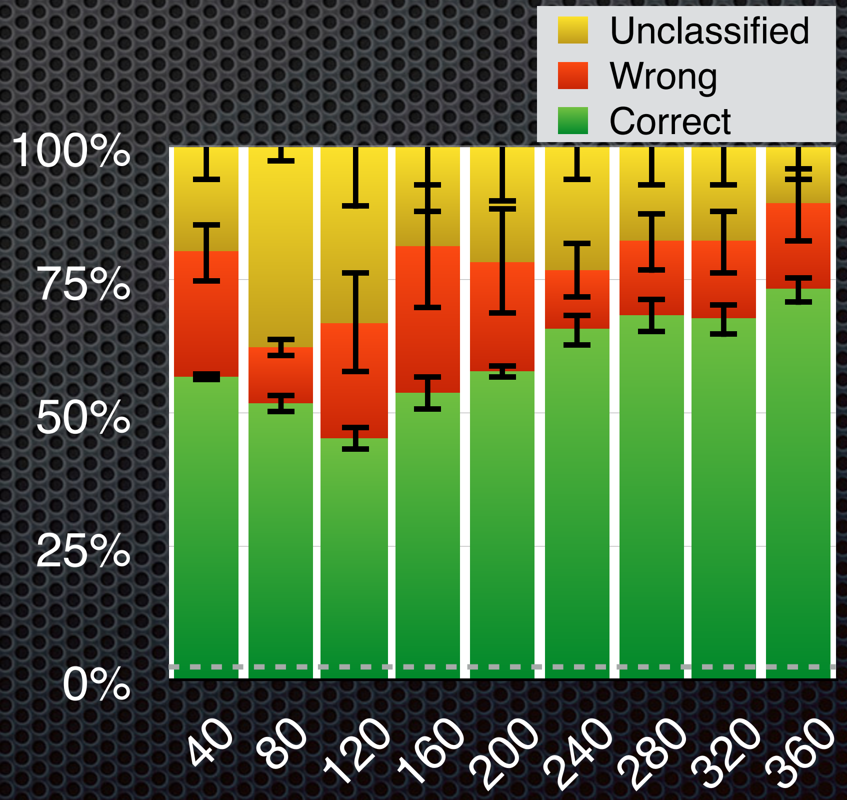
Classification

Error bars indicate 95% CI

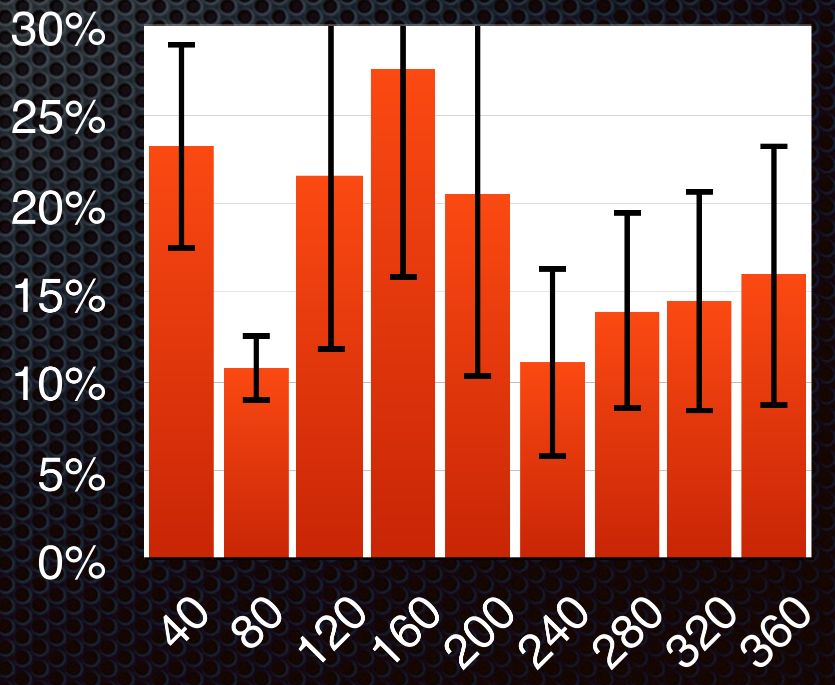


Correct





Wrong



Faces on CM1K emulator

Committed neuron distribution

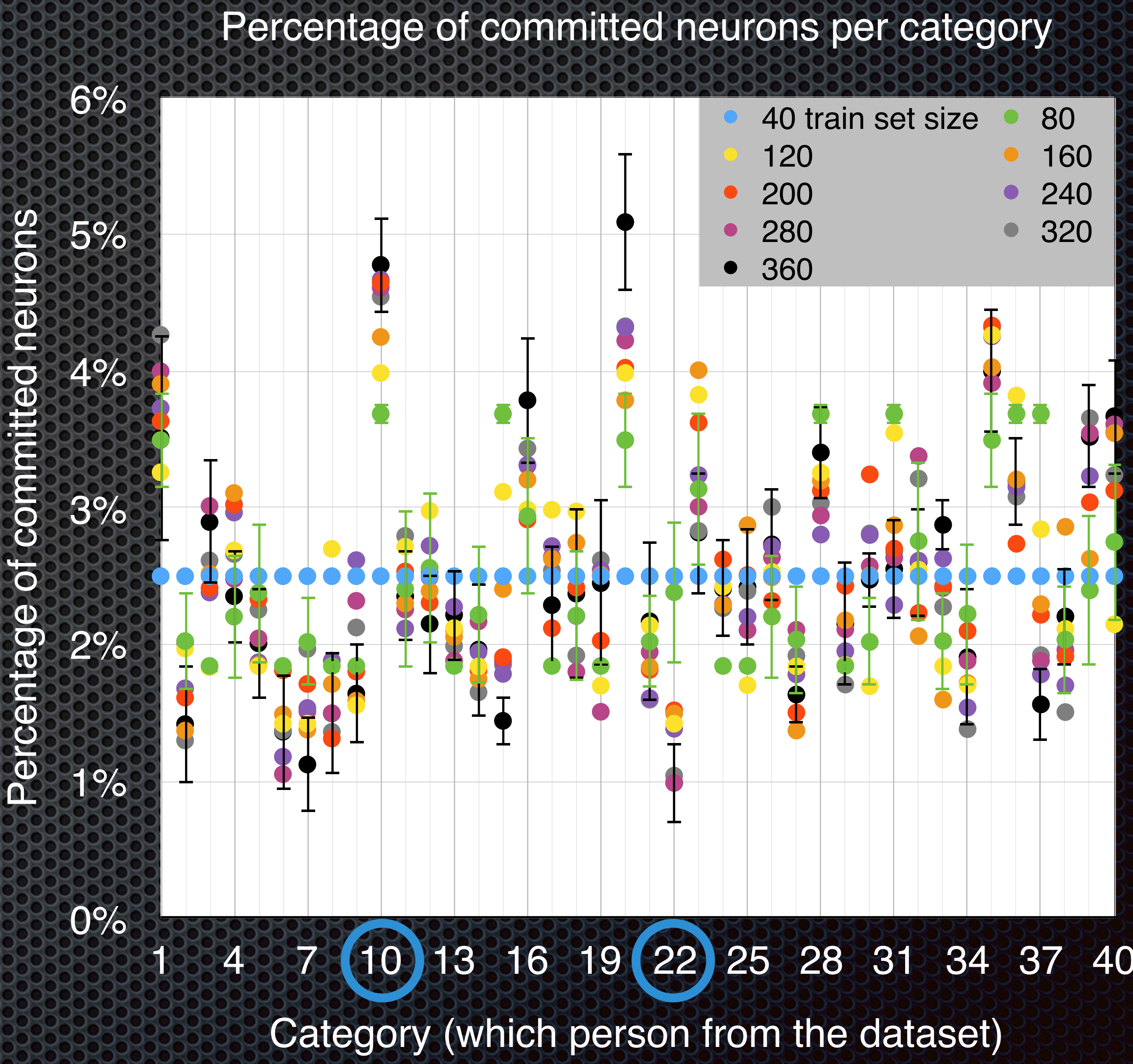
RCE/RBF training can accumulate a wide distribution of prototypes across the categories.

Subject **#10** produces far more prototypes than **#22**. This implies that later training photos of #10 fall outside the AIFs of previous photos of #10, while later photos of #22 fall inside the AIFs of previous photos of #22.

At least two explanations:

- #10's photos have higher variance than #22's.
- #22 resides in an isolated region of the feature space, suffering little overlap with other subjects, thereby maintaining large AIFs, while #10's region overlaps some other subject's region, thus shrinking both of their respective AIFs during training.

Error bars indicate 95% CI
(only included for train set sizes 80 and 360 for clarity)



There is no ordering over the 40 categories (the people). They are ordered in the plot as in the dataset. Train set size 40 looks different because it recruited precisely one instance per class. Thus it exhibits no variance.

Iris on CM1K emulator

- 150 instances:
  - 3 classes of 50 instances each
    - Chance odds of 3-class classification: 33.3%
- 4 measurements provide a 4D feature vector (cm to one decimal point, i.e., mm precision):
  - sepal length
  - sepal width
  - petal length
  - petal width

Sepal length	Sepal width	Petal length	Petal width	Class
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3.0	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
7.0	3.2	4.7	1.4	Iris-versicolor
6.4	3.2	4.5	1.5	Iris-versicolor
6.9	3.1	4.9	1.5	Iris-versicolor
6.3	3.3	6.0	2.5	Iris-virginica
5.8	2.7	5.1	1.9	Iris-virginica
7.1	3.0	5.9	2.1	Iris-virginica
…	…	…	…	…

- Goal: classify or predict a flower’s class from its 4D feature vector

Iris on CM1K emulator

CM1K data preparation

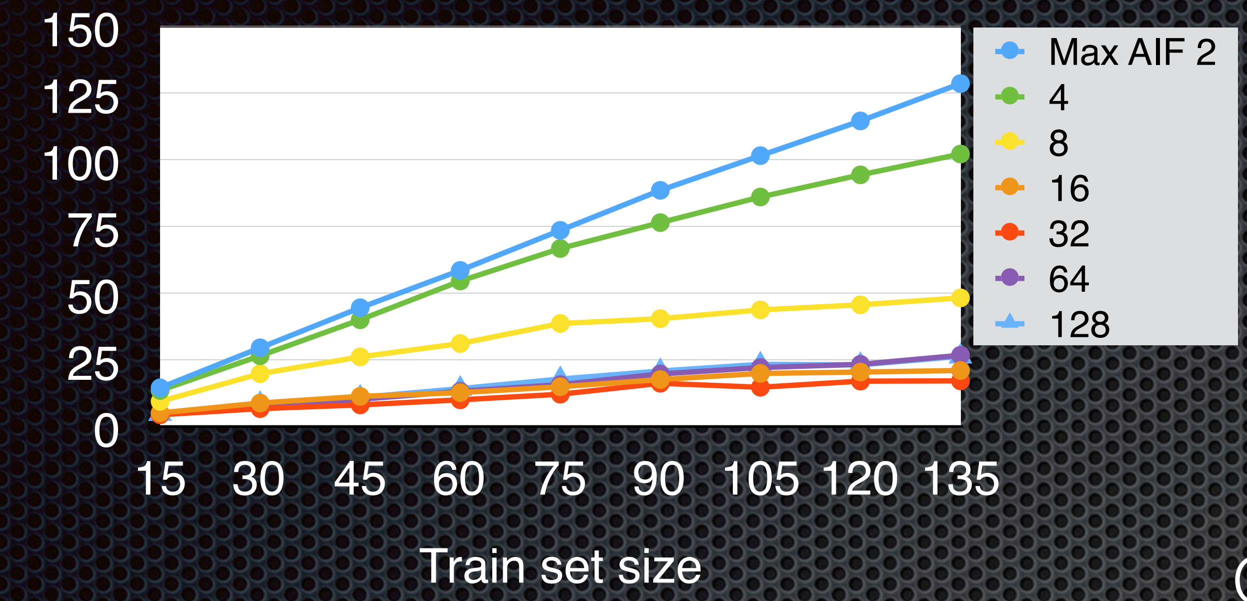
- Drop each feature value's decimal point, thereby converting fractional cm at 1/10th precision into integer mm. All Iris values are below 256mm, and therefore fit in a single byte.
- Arrange the four features in a four-element array, i.e., of the CM1K's 256 available bytes for pattern representation, use only four bytes for the Iris classification problem.
- The maximum theoretical pattern-pair-distance will then be 1020 (255x4), but the range-bounded max. distance is only 143 since the features' various min and max values are > 0 and < 256.

Iris on CM1K emulator

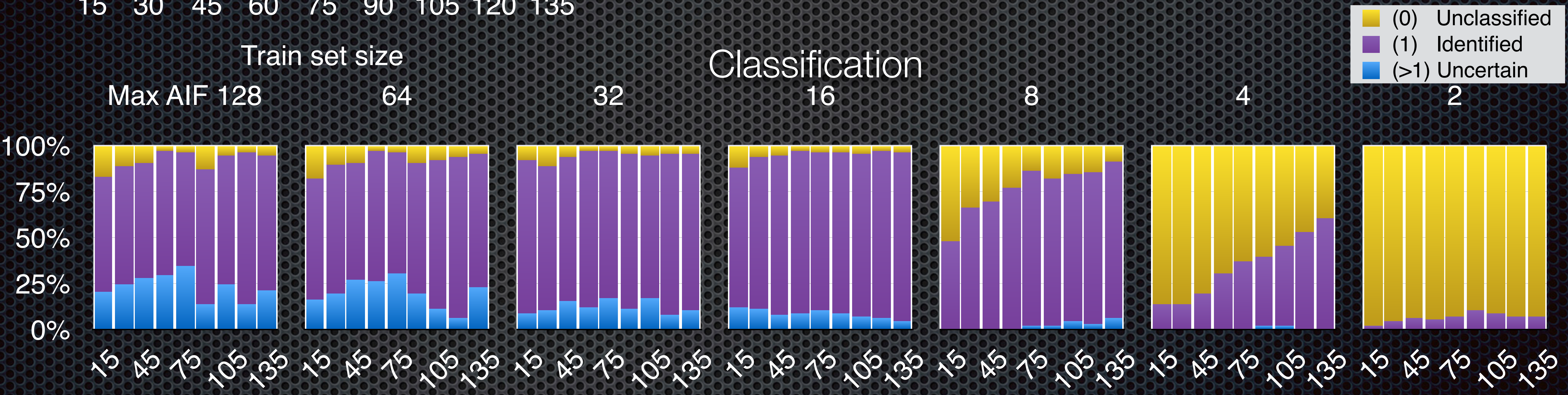
The RCE/RBF parameter *PresetMaxAIF* limits the AIF assigned to a newly recruited neuron, thereby limiting the prototype's generalization to classify new patterns.

Varying *PresetMaxAIF* can significantly alter the network's training and subsequent classification.

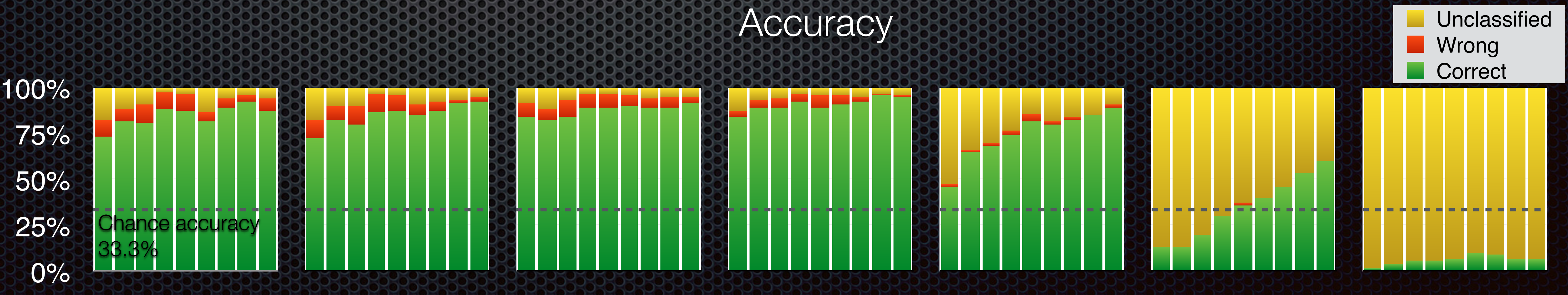
Committed Neurons



Classification



Accuracy



Iris on CM1K emulator

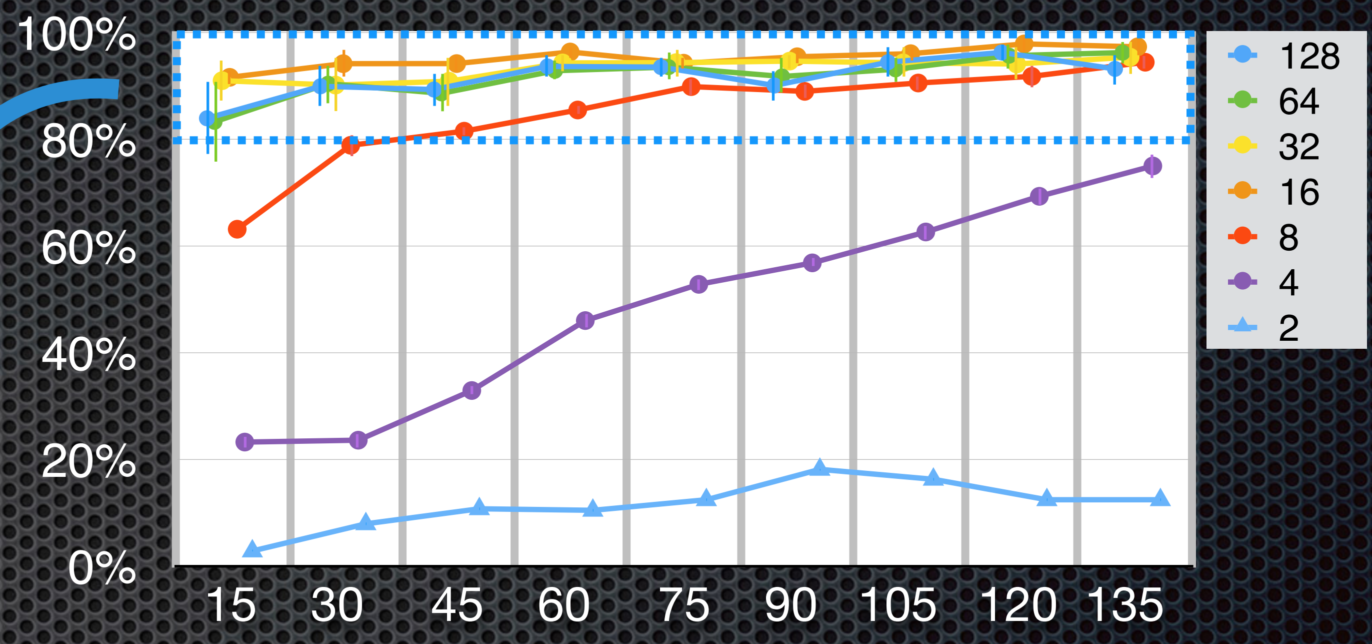
Precision, recall, F1 score

F1 Score

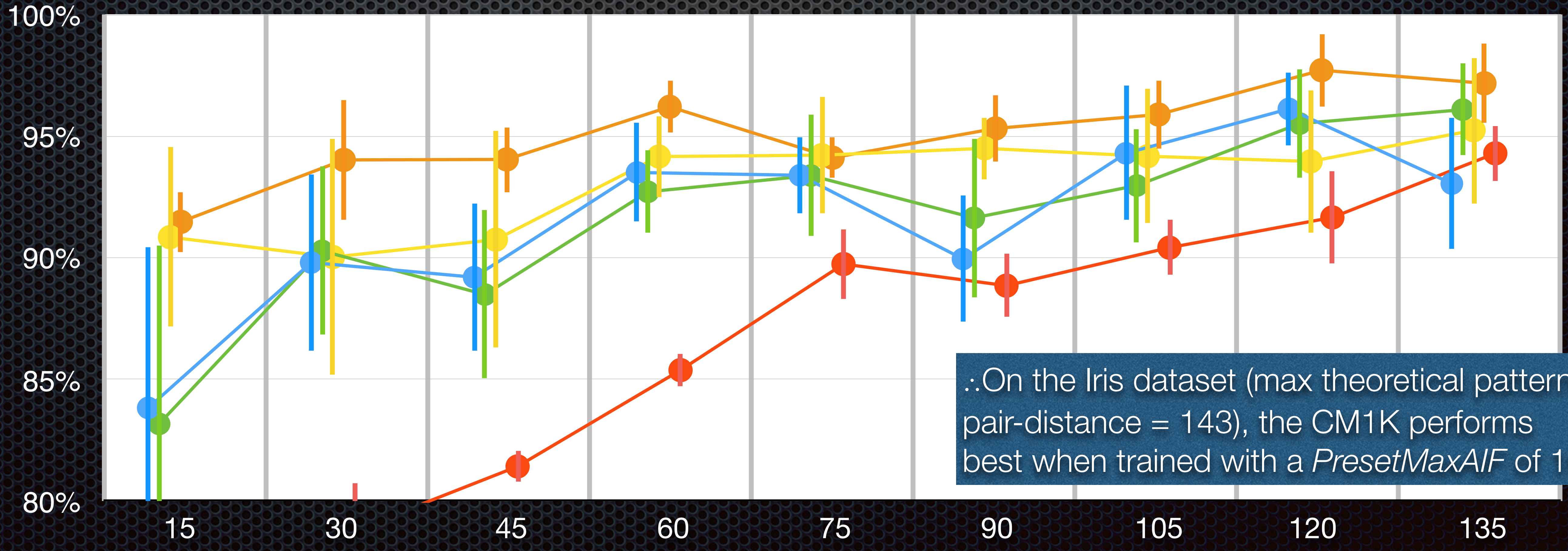
		Train set size								
		15	30	45	60	75	90	105	120	135
Preset Max AIF	**128**	0.84	0.90	0.89	0.94	0.93	0.90	0.94	0.96	0.93
	64	0.83	0.90	0.88	0.93	0.93	0.92	0.93	0.96	0.96
	32	0.91	0.90	0.91	0.94	0.94	0.95	0.94	0.94	0.95
	16	0.91	0.94	0.94	0.96	0.94	0.95	0.96	***0.98***	0.97
	8	0.63	0.79	0.81	0.85	0.90	0.89	0.90	0.92	0.94
	4	0.23	0.24	0.33	0.46	0.53	0.57	0.63	0.69	0.75
	2	0.03	0.08	0.11	0.11	0.13	0.18	0.16	0.13	0.13

Error bars indicate 95% CI

F1 Score



F1 Score



∴On the Iris dataset (max theoretical pattern-pair-distance = 143), the CM1K performs best when trained with a *PresetMaxAIF* of 16.

Mushroom on CM1K emulator

- 8124 instances:
  - 2 classes: edible (4208), poisonous (3916)
    - Chance odds of 2-class classification: 50%
    - Best guess odds: 51.80% (4208 ÷ 8124)
- 22 nominal (i.e., categorical) features provide a 22D feature vector:
  - shape
  - texture
  - color
  - etc.
- Goal: classify or predict a mushroom's edibility or poisonousness from its 22D feature vector

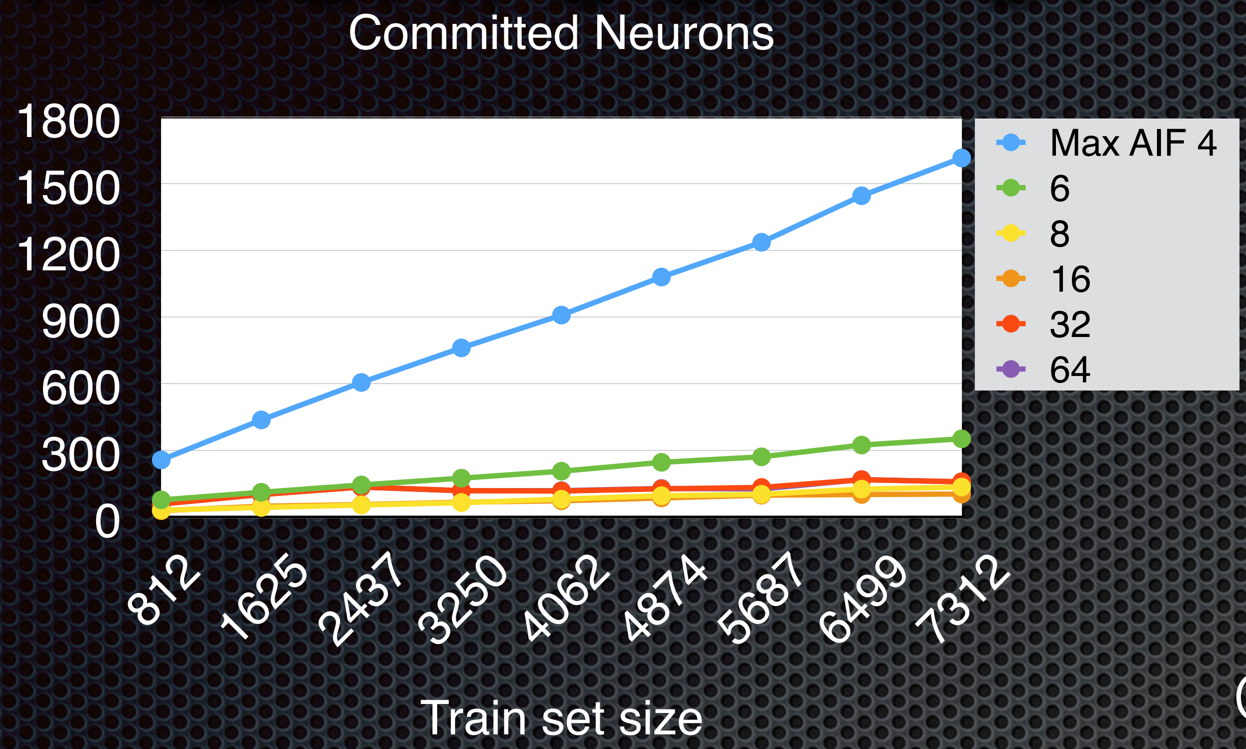
class	p	e	e	p	e	e	e	e	p	e	...
cap-shape	x	x	b	x	x	x	b	b	x	b	...
cap-surface	s	s	s	y	s	y	s	y	y	s	...
cap-color	n	y	w	w	g	y	w	w	w	y	...
bruises	t	t	t	t	f	t	t	t	t	t	...
odor	p	a	l	p	n	a	a	l	p	a	...
gill-attachment	f	f	f	f	f	f	f	f	f	f	...
gill-spacing	c	c	c	c	w	c	c	c	c	c	...
gill-size	n	b	b	n	b	b	b	b	n	b	...
gill-color	k	k	n	n	k	n	g	n	p	g	...
stalk-shape	e	e	e	e	t	e	e	e	e	e	...
stalk-root	e	c	c	e	e	c	c	c	e	c	...
stalk-surface-above-ring	s	s	s	s	s	s	s	s	s	s	...
stalk-surface-below-ring	s	s	s	s	s	s	s	s	s	s	...
stalk-color-above-ring	w	w	w	w	w	w	w	w	w	w	...
stalk-color-below-ring	w	w	w	w	w	w	w	w	w	w	...
veil-type	p	p	p	p	p	p	p	p	p	p	...
veil-color	w	w	w	w	w	w	w	w	w	w	...
ring-number	o	o	o	o	o	o	o	o	o	o	...
ring-type	p	p	p	p	e	p	p	p	p	p	...
spore-print-color	k	n	n	k	n	k	k	n	k	k	...
population	s	n	n	s	a	n	n	s	v	s	...
habitat	u	g	m	u	g	g	m	m	g	m	...

Mushroom on CM1K emulator

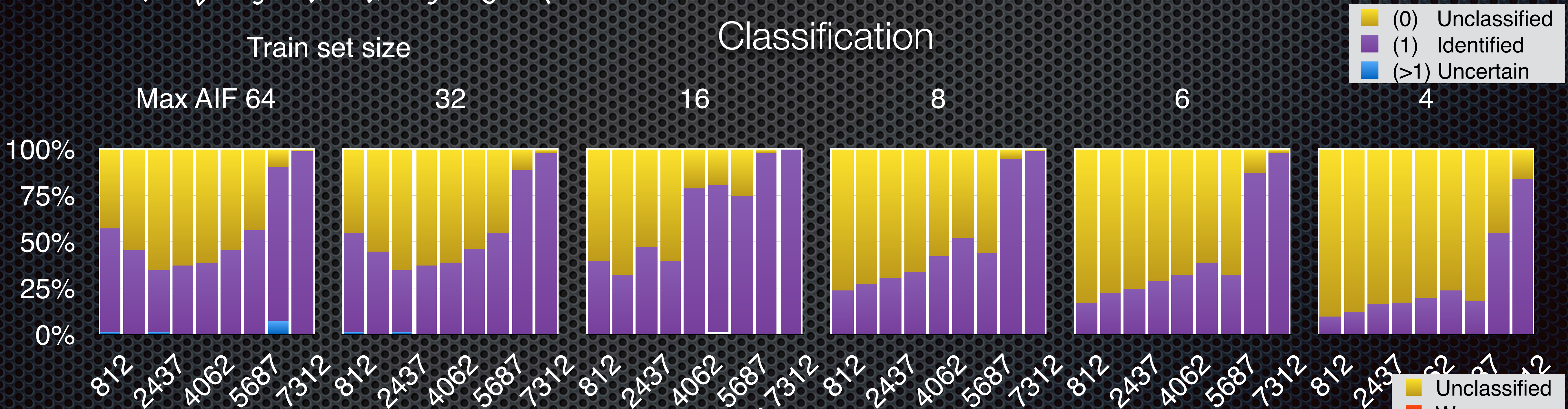
CM1K data preparation

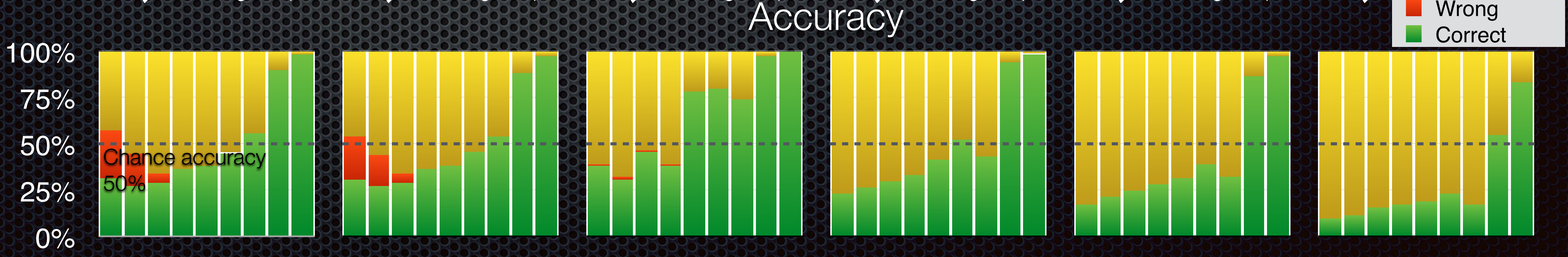
- The 22 features can assume a total of 126 nominal values, which thankfully fits in the CM1K's 256-byte pattern length.
- Convert a given instance to a 126-element boolean array containing precisely 22 *True* values variously interspersed.
- Convert the boolean array to a byte array where each byte is either 0 or 1 (i.e., discard the high 7 bits).
- Of the CM1K's 256 available bytes for pattern representation, use only 126, and of those use only the lowest bit each.
- The maximum theoretical pattern-pair-distance will then be 44, e.g., if all 22 features differ in value.

Mushroom on CM1K emulator



It's interesting that the number of committed neurons is consistent across Max AIFs from 64–8, and then rapidly grows as Max AIF goes to 6 and finally 4.





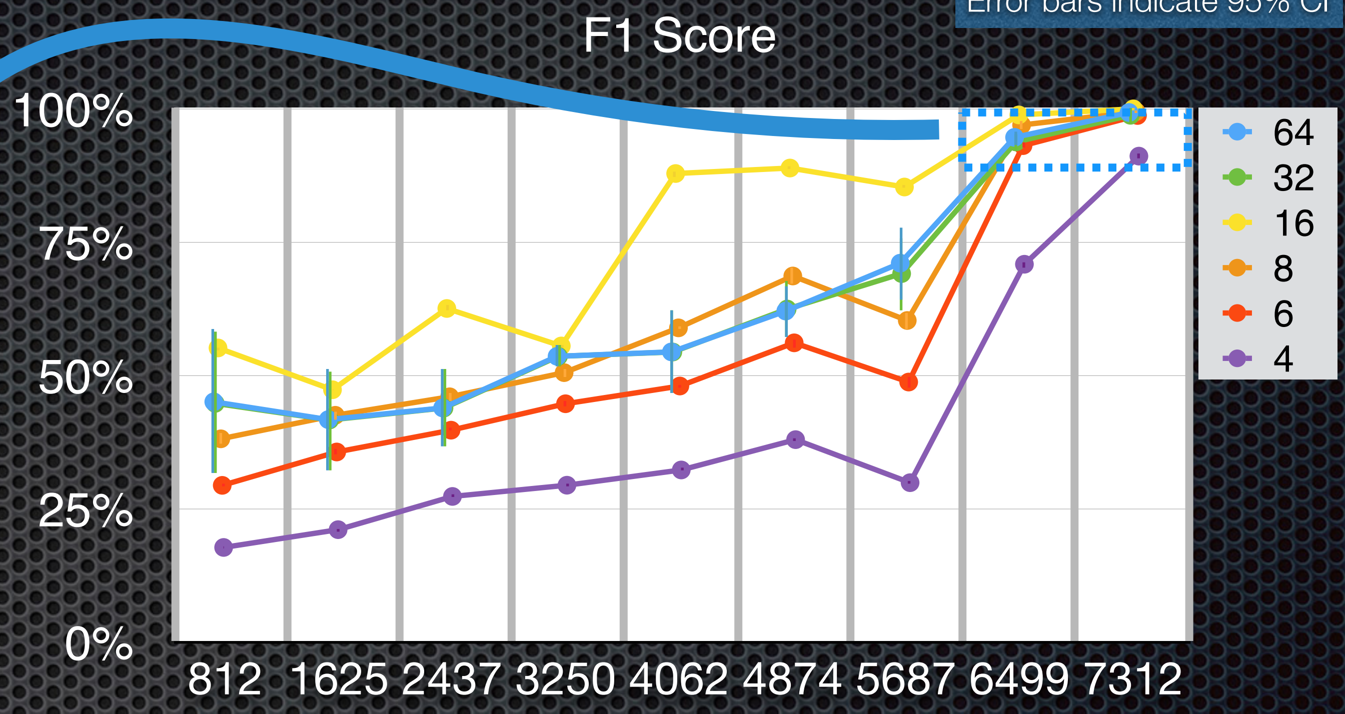
Mushroom on CM1K emulator

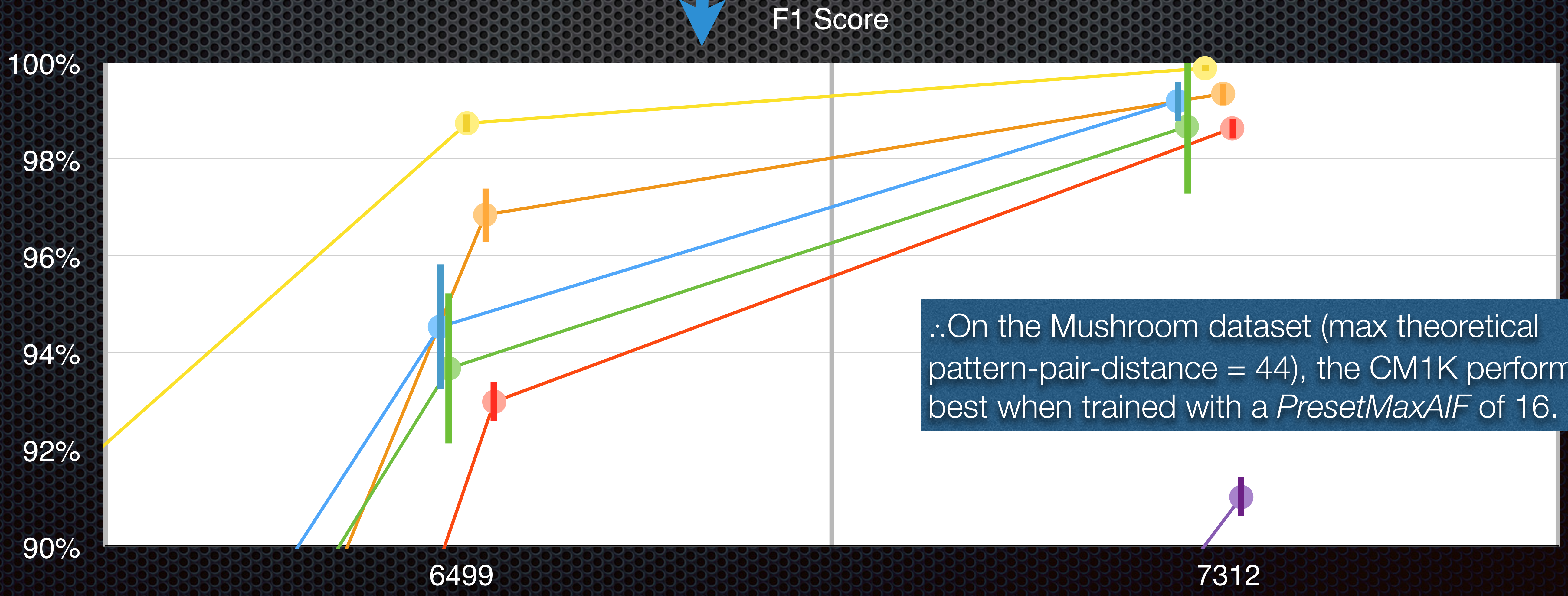
Precision, recall, F1 score

F1 Score

		Train set size								
		812	1625	2437	3250	4062	4874	5687	6499	7312
Preset Max AIF	64	0.45	0.42	0.44	0.53	0.54	0.62	0.71	0.95	0.99
	32	0.45	0.42	0.44	0.53	0.54	0.62	0.69	0.94	0.99
	16	0.55	0.47	0.62	0.55	0.88	0.89	0.85	0.99	***1.00***
	8	0.38	0.42	0.46	0.50	0.59	0.69	0.60	0.97	0.99
	6	0.29	0.36	0.40	0.45	0.48	0.56	0.49	0.93	0.99
	4	0.18	0.21	0.27	0.29	0.32	0.38	0.30	0.71	0.91

Error bars indicate 95% CI





∴On the Mushroom dataset (max theoretical pattern-pair-distance = 44), the CM1K performs best when trained with a *PresetMaxAIF* of 16.

Conclusion

- Neuromorphic computing consists of chip architectures that implement hardware-parallelized neural networks, drawing basic design inspiration from the organization of the brain.
- Fundamental (i.e., Big-O) improvements in performance (constant time scaling with problem size where classical neural network simulations exhibit linear slowdown).
- 2–3 orders-of-magnitude improvement in energy efficiency, enabling entirely new applications, such as bringing pattern recognition to mobile devices.
- But, it is a nascent technology. Commercial designs are either:
    - Affordable but of modest neuron capacity and modeling complexity (CM1K, Intel Curie).
    - Less affordable and still of limited complexity, but offering greater network capacity (modularly networked CM1Ks).
    - Far less affordable, or less available to the public, or still in research stages—but of greater computational complexity and generality (IBM TrueNorth, MIT Eyeriss, Qualcomm Zeroth, KnuPath).