Machine learning is becoming less dependable

Nicholas Carlini
Google

The Year is **2014**

Someone tells you they have a new algorithm to generate synthetic images

The Year is **2014**

Goodfellow et al. Generative Adversarial Nets. NeurIPS 2013.

The Year is **2022**

Someone tells you they have a new algorithm to generate synthetic images

The Year is **2022**

A photo of a Corgi dog riding a bike in Times Square.
It is wearing sunglasses and a beach hat.

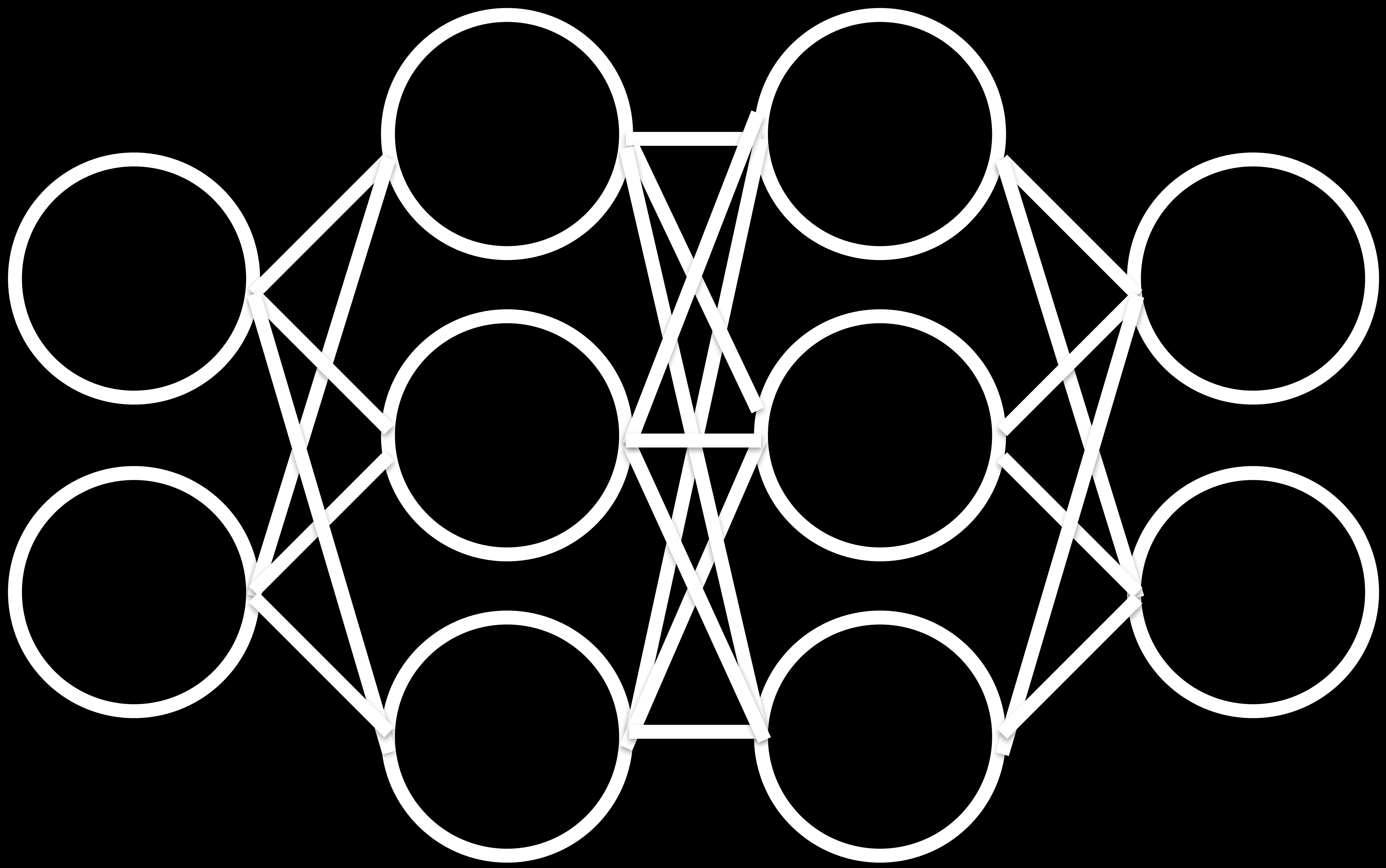
Saharia et al. Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding. 2022.

Bird

or

Bicycle

Competing against:



85% it is a

American Robin

82% it is a

Bicycle

95% it is a

Magpie

99.99% it is

99.99% it is

Bald Eagle

92% it is

Bicycle

99.99% it is

Bald Eagle

This phenomenon is known as an **adversarial example**

B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Srndic, P. Laskov, G. Giacinto, and F. Roli. Evasion attacks against machine learning at test time. 2013.
C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. ICLR 2014.
I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. 2014.

86% it is

45 MPH

Eykholt et al. Robust Physical-World Attacks on Deep Learning Visual Classification. CVPR 2018.

What will a state-of-the-art neural network transcribe?

Carlini & Wagner. Audio Adversarial Examples: Targeted Attacks on Text-to-Speech. 2018

"It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity"

Carlini & Wagner. Audio Adversarial Examples: Targeted Attacks on Text-to-Speech. 2018

Generating Natural Language Adversarial Examples

Moustafa Alzantot[1*], Yash Sharma[2*], Ahmed Elgohary[3], Bo-Jhang Ho[1], Mani B. Srivastava[1], Kai-Wei Chang[1]

[1]Department of Computer Science, University of California, Los Angeles (UCLA)
{malzantot, bojhang, mbs, kwchang}@ucla.edu
[2]Cooper Union sharma2@cooper.edu
[3]Computer Science Department, University of Maryland elgohary@cs.umd.edu

Adversarial Attacks on Neural Network Policies

Sandy Huang[†], Nicolas Papernot[‡], Ian Goodfellow[§], Yan Duan[†§], Pieter Abbeel[†§]
[†] University of California, Berkeley, Department of Electrical Engineering and Computer Sciences
[‡] Pennsylvania State University, School of Electrical Engineering and Computer Science
[§] OpenAI

Abstract

Machine learning classifiers are known to be vulnerable to inputs maliciously constructed by adversaries to force misclassification. Such adversarial examples have been extensively studied in the context of computer vision applications. In this work, we show adversarial attacks are also effective when targeting neural network

Seq2Sick: Evaluating the Robustness of Sequence-to-Sequence Models with Adversarial Examples

Minhao Cheng[1], Jinfeng Yi[2], Huan Zhang[1], Pin-Yu Chen[3], Cho-Jui Hsieh[1]

[1]Department of Computer Science, University of California, Davis, CA 95616
[2]Tencent AI Lab, Bellevue, WA 98004
[3]IBM Research AI, Yorktown Heights, NY 10598
mhcheng@ucdavis.edu, jinfengyi.ustc@gmail.com, ecezhang@ucdavis.edu, pin-yu.chen@ibm.com, chohsieh@ucdavis.edu

HALLUCINATIONS IN NEURAL MACHINE TRANSLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural machine translation (NMT) systems have reached state of the art performance in translating text and are in wide deployment. Yet little is understood about how these systems function or break. Here we show that NMT systems are susceptible to producing highly pathological translations that are completely untethered from the source material, which we term *hallucinations*. Such pathological translations are problematic because they are are deeply disturbing of user trust and easy to find with a simple search. We describe a method to generate hallucinations and show that many common variations of the NMT architecture are susceptible to them. We study a variety of approaches to reduce the frequency of ha[...] nique[...] nally, in the[...]

SYNTHETIC AND NATURAL NOISE BOTH BREAK NEURAL MACHINE TRANSLATION

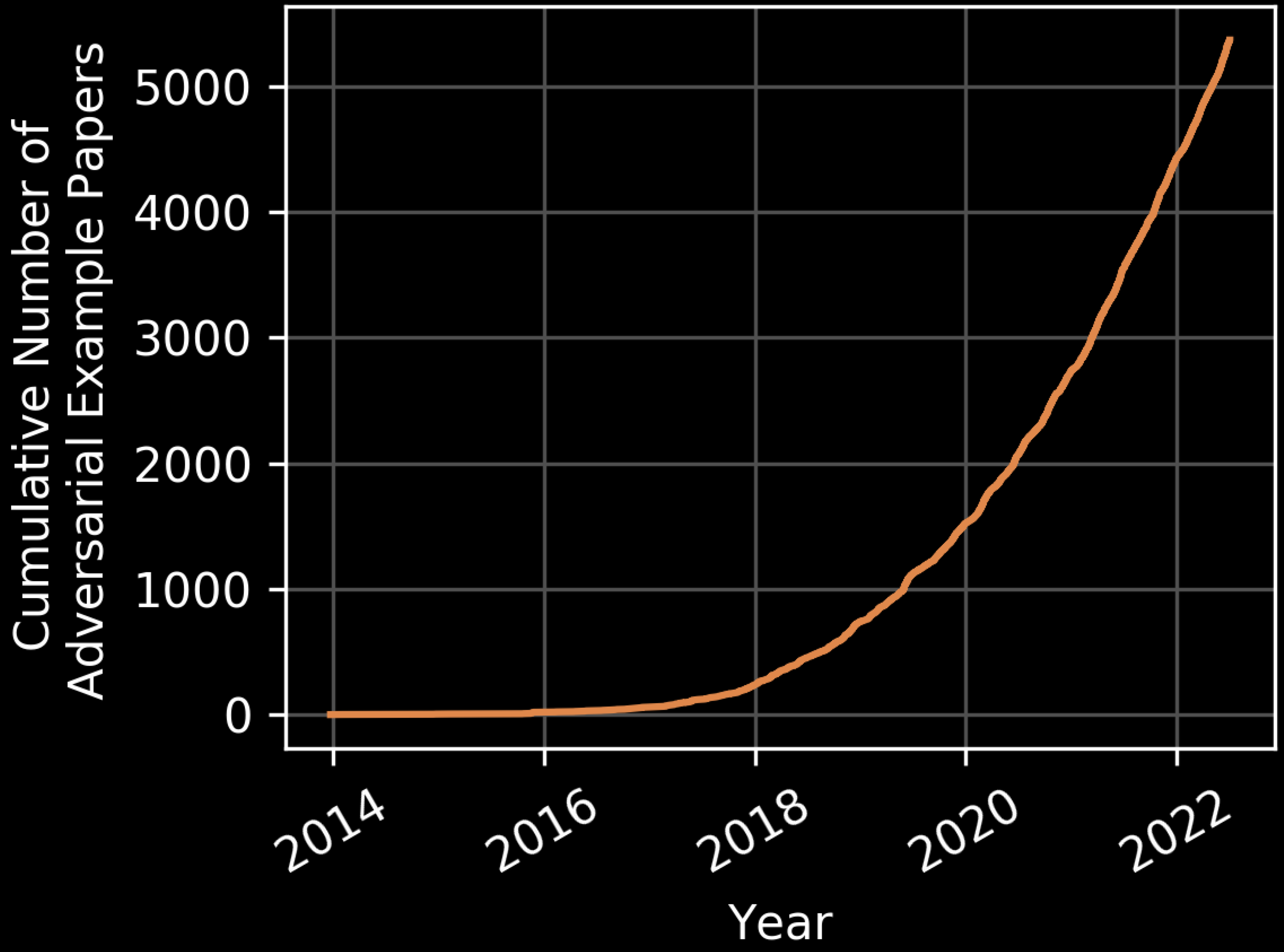
Yonatan Belinkov*
Computer Science and
Artificial Intelligence Laboratory,
Massachusetts Institute of Technology
belinkov@mit.edu

Yonatan Bisk*
Paul G. Allen School
of Computer Science & Engineering,
University of Washington
ybisk@cs.washington.edu

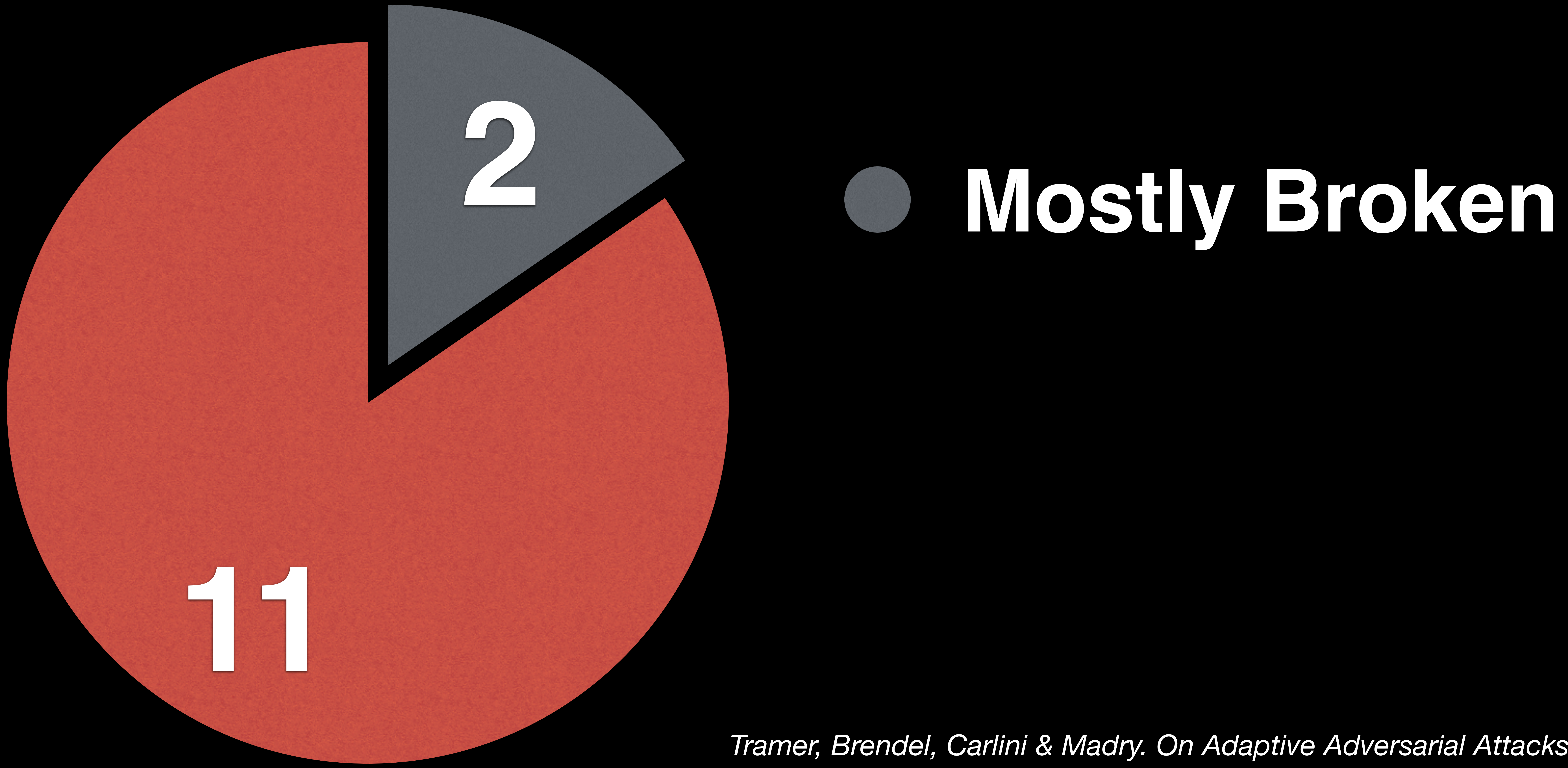
On the Robustness of Semantic Segmentation Models to Adversarial Attacks

Anurag Arnab Ondrej Miksik Philip H.S. Torr
University of Oxford
{anurag.arnab, ondrej.miksik, philip.torr}@eng.ox.ac.uk

People have tried very hard to stop these attack

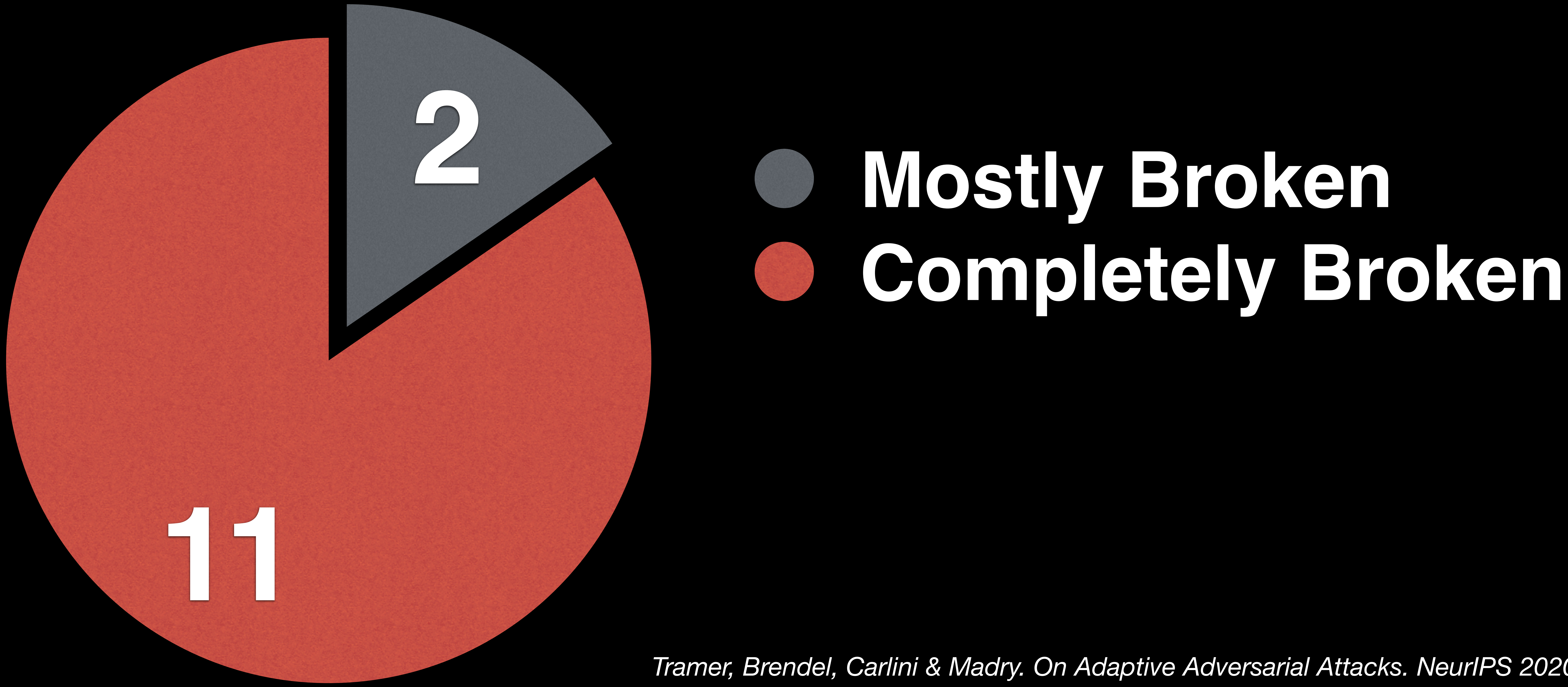


You can't just train this away



Tramer, Brendel, Carlini & Madry. On Adaptive Adversarial Attacks. NeurIPS 2020.

You can't just train this away



Tramer, Brendel, Carlini & Madry. On Adaptive Adversarial Attacks. NeurIPS 2020.

Machine learning is becoming less dependable

Nicholas Carlini
Google

Deep Learning is inscrutable

It's okay for some things to be inscrutable

```
		}
	} while(--i);
	/*
	 * If no process is runnable, idle.
	 */
	if(p == NULL) {
		p = rp;
		idle();
		goto loop;
	}
	rp = p;
	curpri = n;
	/* Switch to stack of the new process and set up
	 * his segmentation registers.
	 */
	retu(rp->p_addr);
	sureg();
	/*
	 * If the new process paused because it was
	 * swapped out, set the stack level to the last call
	 * to savu(u_ssav).  This means that the return
	 * which is executed immediately after the call to aretu
	 * actually returns from the last routine which did
	 * the savu.
	 *
	 * You are not expected to understand this.
	 */
	if(rp->p_flag&SSWAP) {
		rp->p_flag =& ~SSWAP;
		aretu(u.u_ssav);
	}
	/* The value returned here has many subtle implications.
	 * See the newproc comments.
	 */
	return(1);
}
/* -------------------------       */
```



```
 * resume after the swap in swtch, which
 * from this stack level.
 *
 * After the expansion, the caller will take
 * the user's stack towards or away from th
 */
expand(newsize)
{
	int i, n;
	register *p, a1, a2;

	p = u.u_procp;
	n = p->p_size;
	p->p_size = newsize;
	a1 = p->p_addr;
	if(n >= newsize) {
		mfree(coremap, n-newsize, a1+newsize);
		return;
	}
	savu(u.u_rsav);
	a2 = malloc(coremap, newsize);
	if(a2 == NULL) {
		savu(u.u_ssav);
		xswap(p, 1, n);
		p->p_flag =| SSWAP;
		swtch();
		/* no return */
	}
	p->p_addr = a2;
	for(i=0; i<n; i++)
		copyseg(a1+i, a2++);
	mfree(coremap, n, a1);
	retu(p->p_addr);
	sureg();
}
/* -------------------------       */
```

```
		}
	} while(--i);
	/*
	 * If no process is runnable, idle.
	 */
	if(p == NULL) {
		p = rp;
		idle();
		goto loop;
```

```
 * from this stack level.
 *
 * After the expansion, the caller will ...
 * the user's stack towards or away from the ...
 */
expand(newsize)
{
	int i, n;
```

```
	/*
	 * If the new process paused because it was
	 * swapped out, set the stack level to the last call
	 * to savu(u_ssav).  This means that the return
	 * which is executed immediately after the call to aretu
	 * actually returns from the last routine which did
	 * the savu.
	 *
	 * You are not expected to understand this.
	 */
	if(rp->p_flag&SSWAP) {
		rp->p_flag =& ~SSWAP;
		aretu(u.u_ssav);
	}
```


Sheet 22

In contrast:

Deep learning is inscrutable even to the experts.

adversarial perturbation

88% **tabby cat**

99% **guacamole**

Various proposed explanations for adversarial examples

Published as a conference paper at ICLR 2015

EXPLAINING AND HARNESSING
ADVERSARIAL EXAMPLES

Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy
Google Inc., Mountain View, CA
{goodfellow,shlens,szegedy}@google.com

ABSTRACT

Several machine learning models, including neural networks, consistently mis-
classify *adversarial examples*—inputs fo
worst-case perturbations to examples fro
put results in the model outputting an inco
attempts at explaining this phenomenon
We argue instead that the primary cause
versarial perturbation is their linear natu
quantitative results while giving the firs
about them: their generalization across a
this view yields a simple and fast method
ing this approach to provide examples fo
set error of a maxout network on the MN

Adversarial Examples Are Not Bugs, They Are Features

Andrew Ilyas*
MIT
ailyas@mit.edu

Shibani Santurkar*
MIT
shibani@mit.edu

Dimitris Tsipras*
MIT
tsipras@mit.ed

Logan Engstrom*
MIT
engstrom@mit.edu

Brandon Tran
MIT
btran115@mit.edu

Aleksander Mą
MIT
madry@mit.ed

Abstract

Adversarial examples have attracted significant attention in machine learning, but the reas
existence and pervasiveness remain unclear. We demonstrate that adversarial examples can be
tributed to the presence of *non-robust features*: features (derived from patterns in the data distri
are highly predictive, yet brittle and (thus) incomprehensible to humans. After capturing th
within a theoretical framework, we establish their widespread existence in standard datasets.
present a simple setting where we can rigorously tie the phenomena we observe in practice t
ment between the (human-specified) notion of robustness and the inherent geometry of the da

Adversarial Examples Are a Natural Con

Nicolas Ford [*12] **Justin Gilmer** [*1] **Nichol**

Abstract

Over the last few years, the phenomenon of *adversarial examples* — maliciously constructed inputs that fool trained machine learning models — has captured the attention of the research community, especially when the adversary is restricted to small modifications of a correctly handled input. Less surprisingly, image classifiers also lack human-level performance on randomly corrupted images, such as images with additive Gaussian noise. In this paper we provide both empirical and theoretical evidence that these are two manifestations of the same underlying phenomenon, establishing close connections between the adversarial robustness and corruption robustness research programs. This suggests that improving adversarial robustness should go hand in hand with improving performance in the presence of more general and realistic image corruptions. Based on our results we recommend that future adversarial defenses consider evaluating the robustness of their methods to distributional shift with benchmarks such as Imagenet-C.

latter phenomenon has struck many in the machine learning community as surprising and has attracted a great deal of research interest, while the former has received considerably less attention.

The machine learning community has researchers working on each of these two types of errors: adversarial example researchers seek to measure and improve robustness to small-worst case perturbations of the input while corruption robustness researchers seek to measure and improve model robustness to distributional shift. In this work we analyze the connection between these two research directions, and we see that adversarial robustness is closely related to robustness to certain kinds of distributional shift. In other words, the existence of adversarial examples follows naturally from the fact that our models have nonzero test error in certain corrupted image distributions.

We make this connection in several ways. First, in Section 4, we provide a novel analysis of the error set of an image classifier. We see that, given the error rates we observe in Gaussian noise, the small adversarial perturbations we observe in practice appear at roughly the distances we would expect from a *linear* model, and that therefore there is no need to invoke any strange properties of the decision bound-

The Dimpled Manifold Model of Adversarial Examples in Machine Learning

Adi Shamir
Faculty of Math&CS
Weizmann Institute of Science
Israel
adi.shamir@weizmann.ac.il

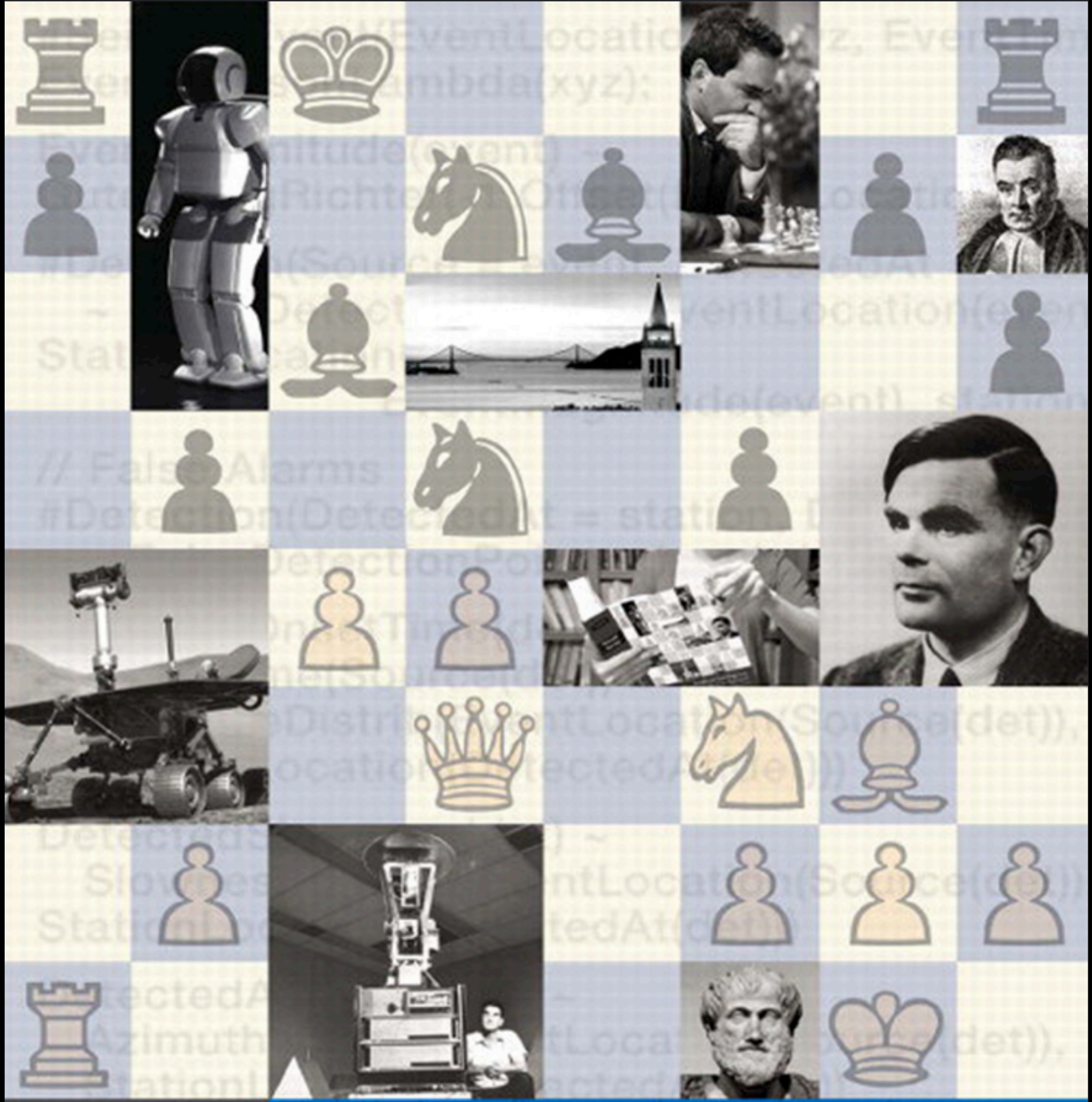
Odelia Melamed
Faculty of Math&CS
Weizmann Institute of Science
Israel
odelia.melamed@weizmann.ac.il

Oriel BenShmuel
Faculty of Math&CS
Weizmann Institute of Science
Israel
oriel.benshmuel@weizmann.ac.il

Abstract

The extreme fragility of deep neural networks, when presented with tiny perturbations in their inputs, was independently discovered by several research groups in 2013. However, despite enormous effort, these adversarial examples remained a counterintuitive phenomenon with no simple testable explanation. In this paper, we introduce a new conceptual framework for how the decision boundary between classes evolves during training, which we call the *Dimpled Manifold Model*. In particular, we demonstrate that training is divided into two distinct phases. The first phase is a (typically fast) clinging process in which the initially randomly oriented decision boundary gets very close to the low dimensional image manifold, which contains all the training examples. Next, there is a (typically slow) dimpling phase which creates shallow bulges in the decision boundary that move it to the correct side of the training examples. This framework provides a simple explanation for why adversarial examples exist, why their perturbations have such tiny norms, and why they look like random noise rather than like the target class. This explanation is also used to show that a network that was adversarially trained with incorrectly labeled images might still correctly classify most test images, and to show that the main effect of adversarial training is just to deepen the generated dimples in the decision boundary. Finally, we discuss and demonstrate the very different properties of on-manifold and off-manifold adversarial perturbations. We describe the results of numerous experiments which strongly support this new model, using both low dimensional synthetic datasets and high dimensional natural datasets.

Let me just give you another example...



Stuart
Russell
Peter
Norvig

Artificial Intelligence
A Modern Approach
Third Edition

```
function CROSS-VALIDATION-WRAPPER(Learner, k, examples) returns a hypothesis
  local variables: errT, an array, indexed by size, storing training-set error rates
                   errV, an array, indexed by size, storing validation-set error rates
  for size = 1 to ∞ do
     errT[size], errV[size] ← CROSS-VALIDATION(Learner, size, k, examples)
     if errT has converged then do
        best_size ← the value of size with minimum errV[size]
        return Learner(best_size, examples)

function CROSS-VALIDATION(Learner, size, k, examples) returns two values:
        average training set error rate, average validation set error rate

  fold_errT ← 0; fold_errV ← 0
  for fold = 1 to k do
     training_set, validation_set ← PARTITION(examples, fold, k)
     h ← Learner(size, training_set)
     fold_errT ← fold_errT + ERROR-RATE(h, training_set)
     fold_errV ← fold_errV +ERROR-RATE(h, validation_set)
  return fold_errT/k, fold_errV/k
```

Figure 18.8 An algorithm to select the model that has the lowest error rate on validation data by building models of increasing complexity, and choosing the one with best empirical error rate on validation data. Here *errT* means error rate on the training data, and *errV* means error rate on the validation data. *Learner*(*size*, *examples*) returns a hypothesis whose complexity is set by the parameter *size*, and which is trained on the *examples*. PARTITION(*examples*, *fold*, *k*) splits *examples* into two subsets: a validation set of size N/k and a training set with all the other examples. The split is different for each value of *fold*.

18.4.2 From error rates to loss

So far, we have been trying to minimize error rate. This is clearly better than maximizing error rate, but it is not the full story. Consider the problem of classifying email messages as spam or non-spam. It is worse to classify non-spam as spam (and thus potentially miss an important message) then to classify spam as non-spam (and thus suffer a few seconds of annoyance). So a classifier with a 1% error rate, where almost all the errors were classifying spam as non-spam, would be better than a classifier with only a 0.5% error rate, if most of those errors were classifying non-spam as spam. We saw in Chapter 16 that decision-makers should maximize expected utility, and utility is what learners should maximize as well. In machine learning it is traditional to express utilities by means of a **loss function**. The loss function $L(x, y, \hat{y})$ is defined as the amount of utility lost by predicting $h(x) = \hat{y}$ when the correct answer is $f(x) = y$:

LOSS FUNCTION

$$\begin{aligned} L(x, y, \hat{y}) &= \mathit{Utility}(\text{result of using } y \text{ given an input } x) \\ &- \mathit{Utility}(\text{result of using } \hat{y} \text{ given an input } x) \end{aligned}$$

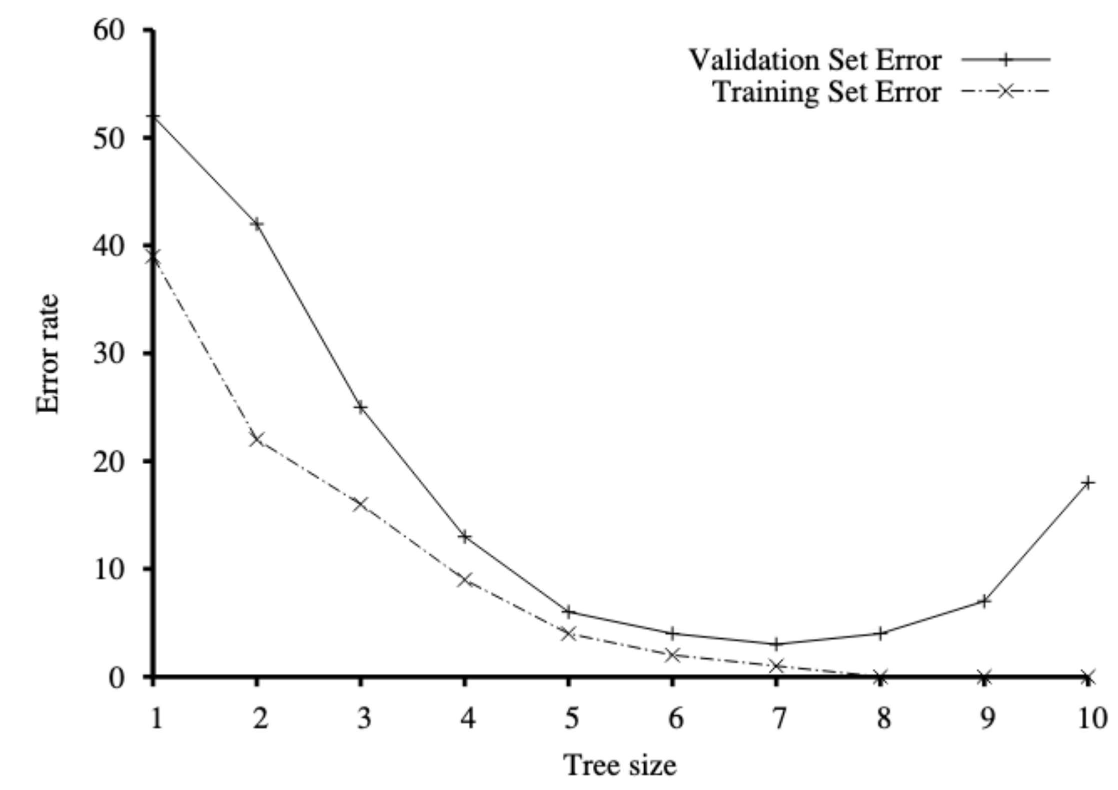


Figure 18.9 Error rates on training data (lower, dashed line) and validation data (upper, solid line) for different size decision trees. We stop when the training set error rate asymptotes, and then choose the tree with minimal error on the validation set; in this case the tree of size 7 nodes.

This is the most general formulation of the loss function. Often a simplified version is used, $L(y, \hat{y})$, that is independent of x. We will use the simplified version for the rest of this chapter, which means we can't say that it is worse to misclassify a letter from Mom than it is to misclassify a letter from our annoying cousin, but we can say it is 10 times worse to classify non-spam as spam than vice-versa:

$$L(\mathit{spam}, \mathit{nospam}) = 1, \quad L(\mathit{nospam}, \mathit{spam}) = 10\,.$$

Note that $L(y, y)$ is always zero; by definition there is no loss when you guess exactly right. For functions with discrete outputs, we can enumerate a loss value for each possible misclassification, but we can't enumerate all the possibilities for real-valued data. If $f(x)$ is 137.035999, we would be fairly happy with $h(x) = 137.036$, but just how happy should we be? In general small errors are better than large ones; two functions that implement that idea are the absolute value of the difference (called the L_1 loss), and the square of the difference (called the L_2 loss). If we are content with the idea of minimizing error rate, we can use the $L_{0/1}$ loss function, which has a loss of 1 for an incorrect answer and is appropriate for discrete-valued outputs:

$$\begin{aligned} &\text{Absolute value loss:} && L_1(y, \hat{y}) = |y - \hat{y}| \\ &\text{Squared error loss:} && L_2(y, \hat{y}) = (y - \hat{y})^2 \\ &\text{0/1 loss:} && L_{0/1}(y, \hat{y}) = 0 \text{ if } y = \hat{y}, \text{ else } 1 \end{aligned}$$

The learning agent can theoretically maximize its expected utility by choosing the hypothesis that minimizes expected loss over all input–output pairs it will see. It is meaningless to talk about this expectation without defining a prior probability distribution, $\mathbf{P}(X, Y)$ over examples. Let $\mathcal{E}$ be the set of all possible input–output examples. Then the expected **generalization loss** for a hypothesis h (with respect to loss function L) is

GENERALIZATION LOSS

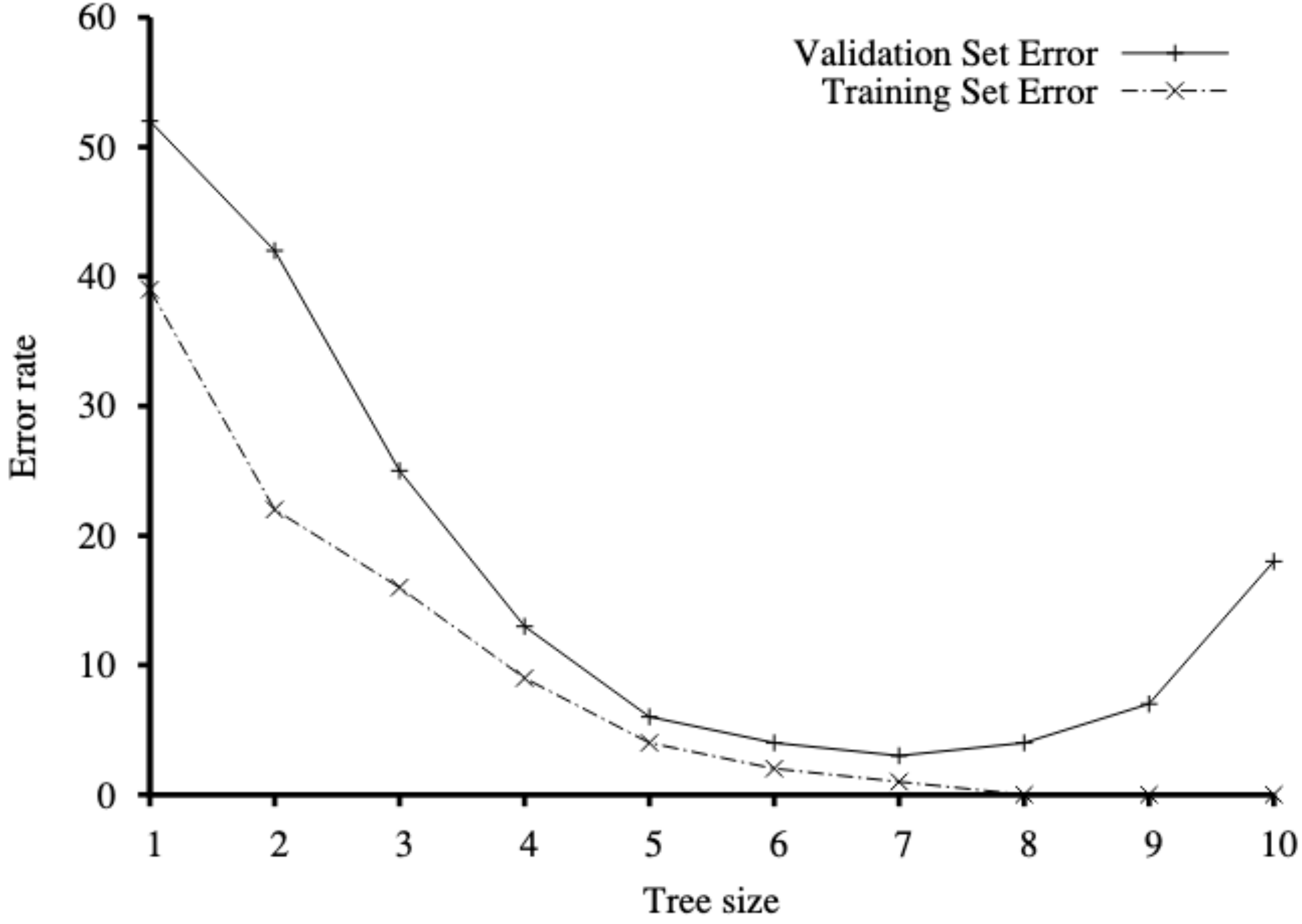


Figure 18.9 Error rates on training data (lower, dashed line) and validation data (upper, solid line) for different size decision trees. We stop when the training set error rate asymptotes, and then choose the tree with minimal error on the validation set; in this case the tree

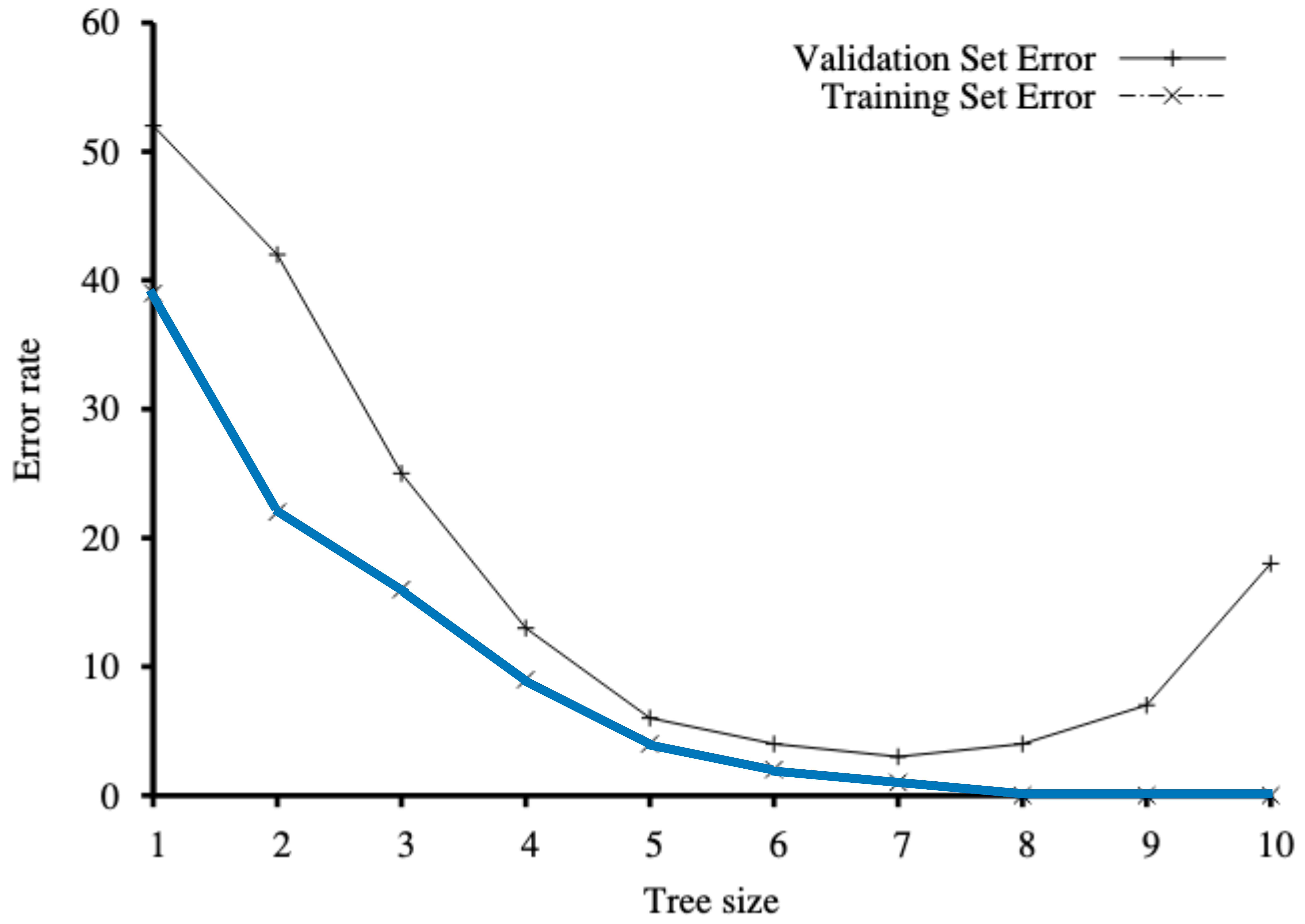


Figure 18.9 Error rates on training data (lower, dashed line) and validation data (upper, solid line) for different size decision trees. We stop when the training set error rate asymp-

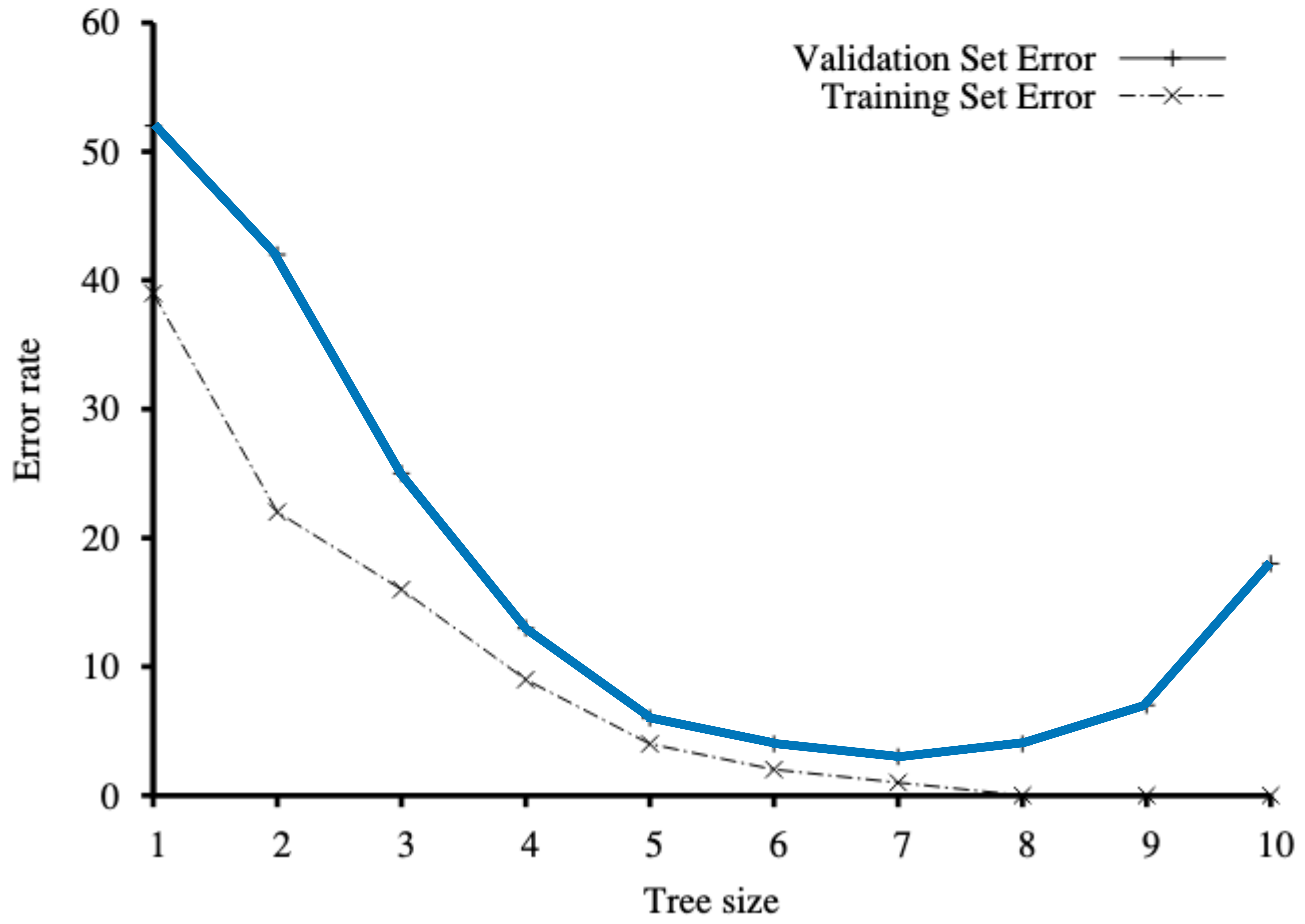


Figure 18.9 Error rates on training data (lower, dashed line) and validation data (upper, solid line) for different size decision trees. We stop when the training set error rate asymptotes, and then choose the tree with minimal error on the validation set; in this case the tree

In mathematical modeling, **overfitting** is "the production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit to additional data or predict future observations reliably".[1] An **overfitted model** is a mathematical model that contains more parameters than can be justified by the data.[2] The essence of overfitting is to have unknowingly extracted some of the residual variation (i.e., the noise)

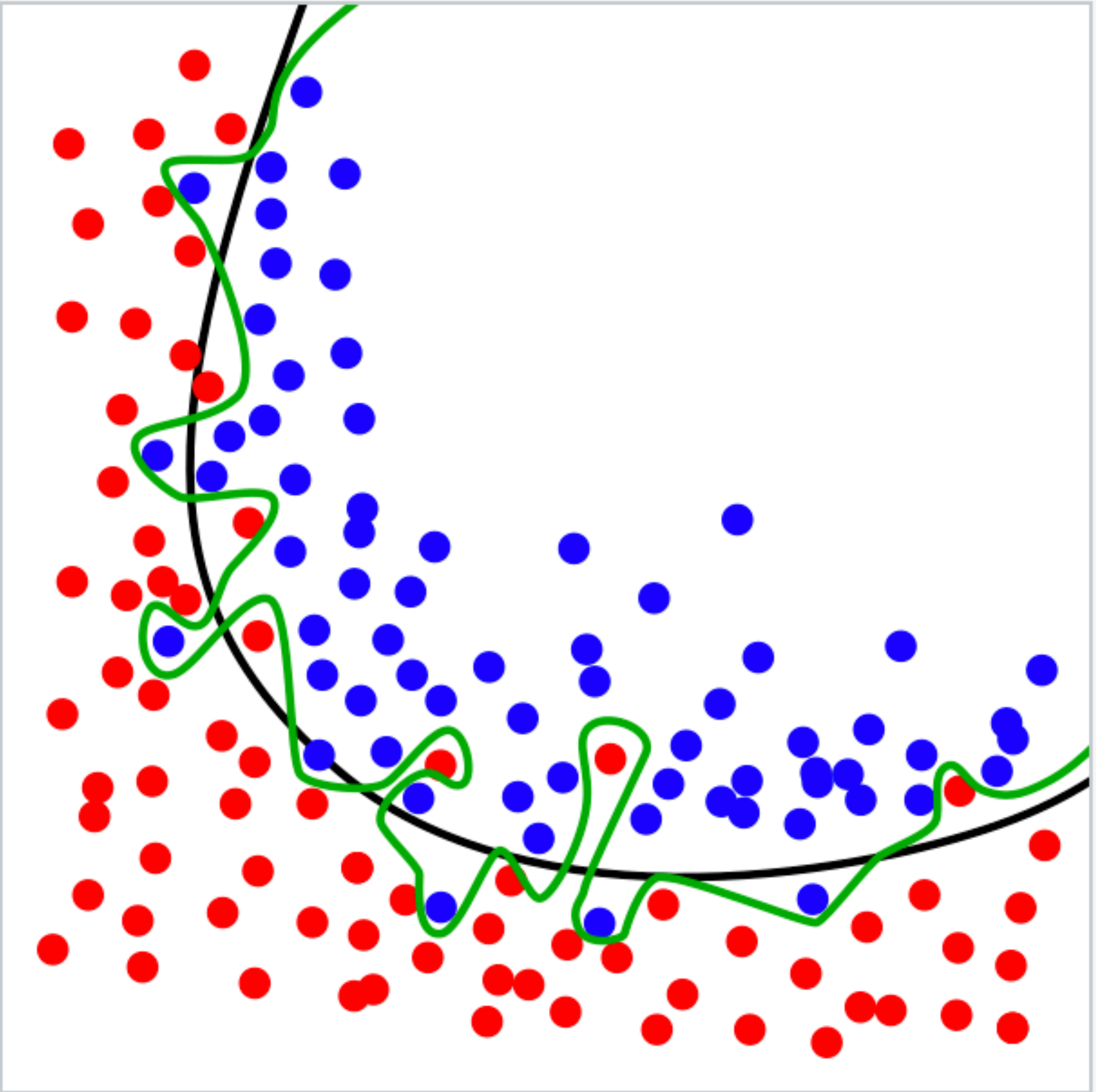
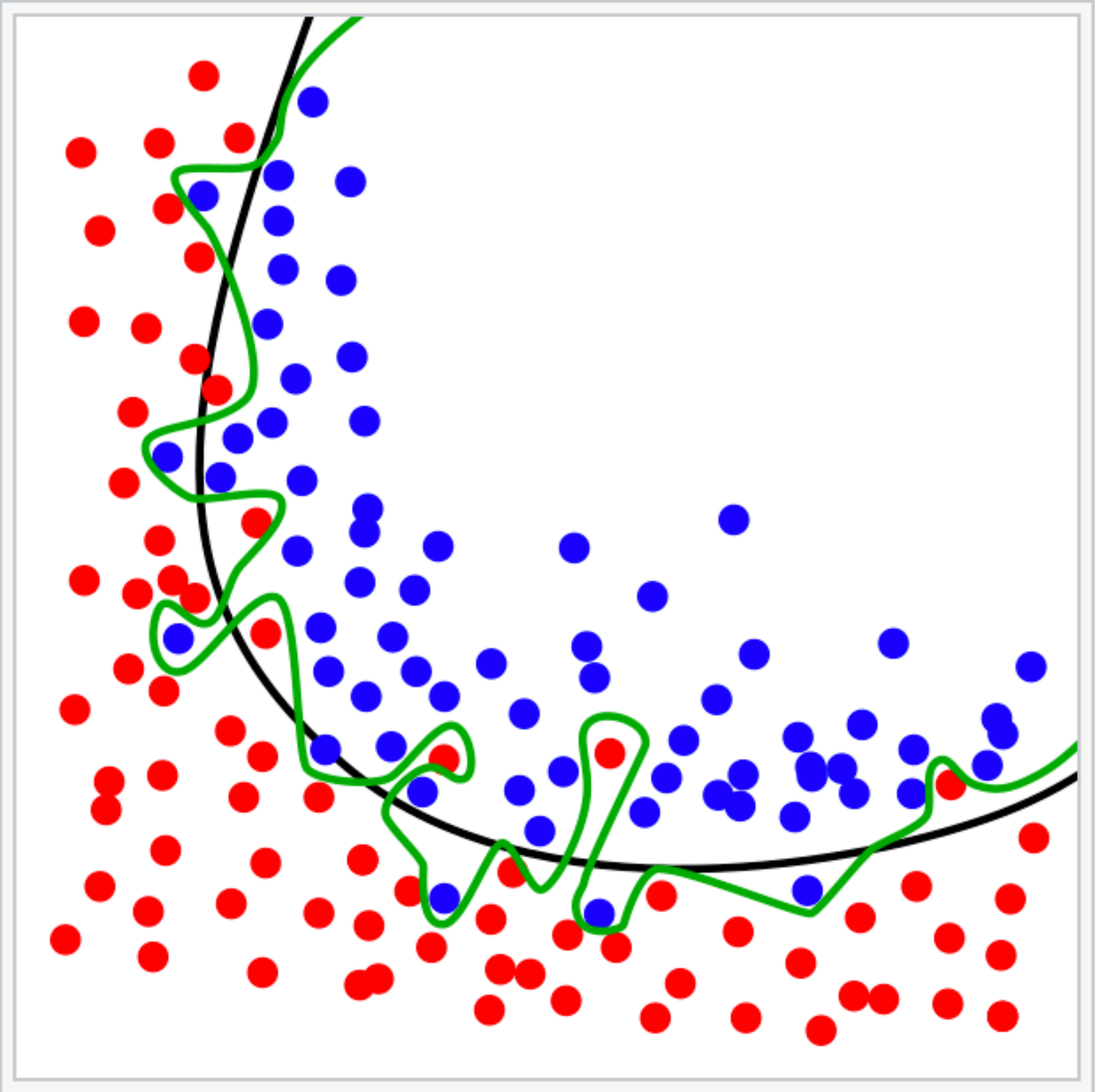


Figure 1. The green line represents an overfitted model and the black line represents a regularized model. While the green line best follows the training data, it is too dependent on that data and it is likely to have a higher error rate on new unseen data, compared to the black line.



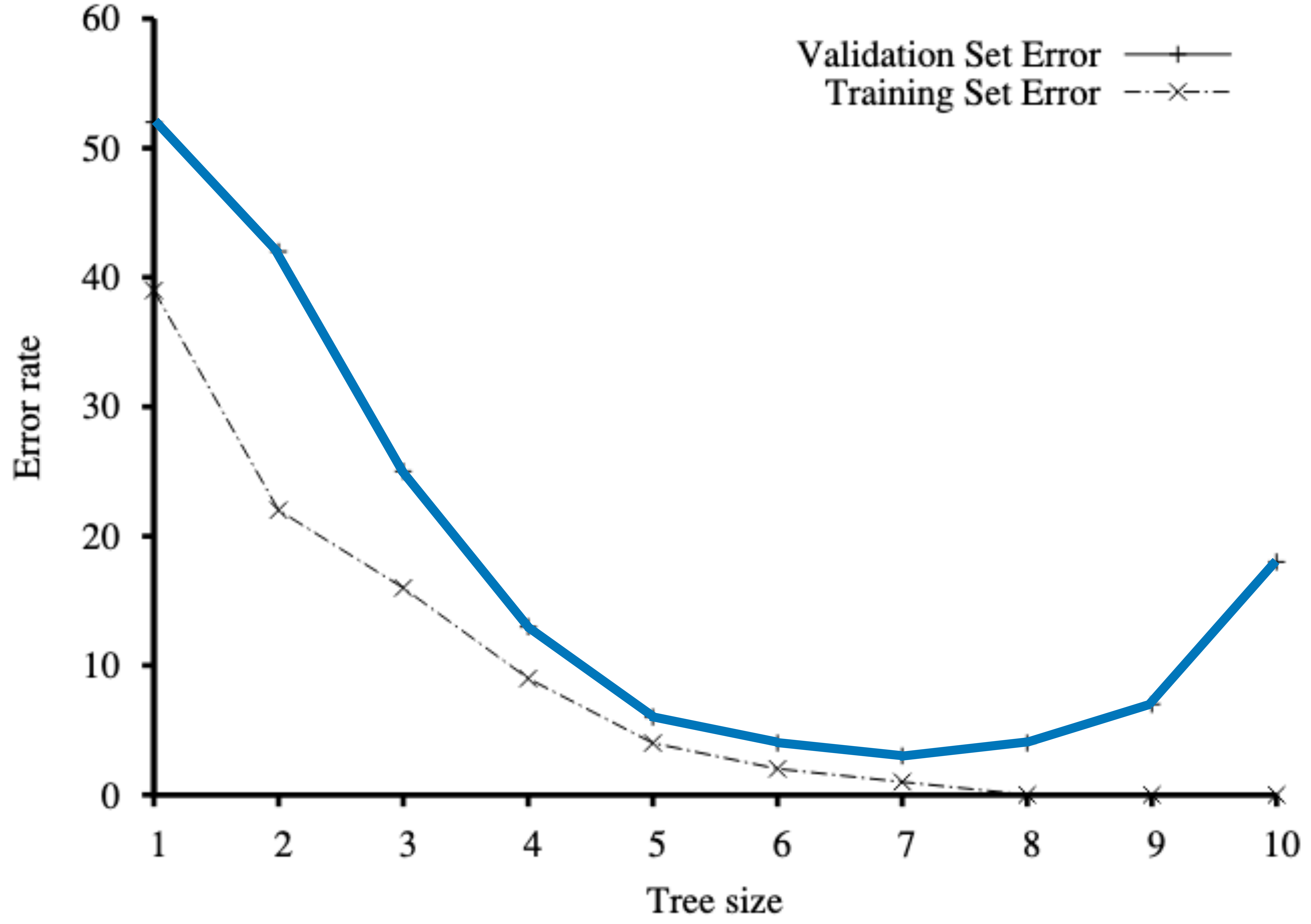


Figure 18.9 Error rates on training data (lower, dashed line) and validation data (upper, solid line) for different size decision trees. We stop when the training set error rate asymp-

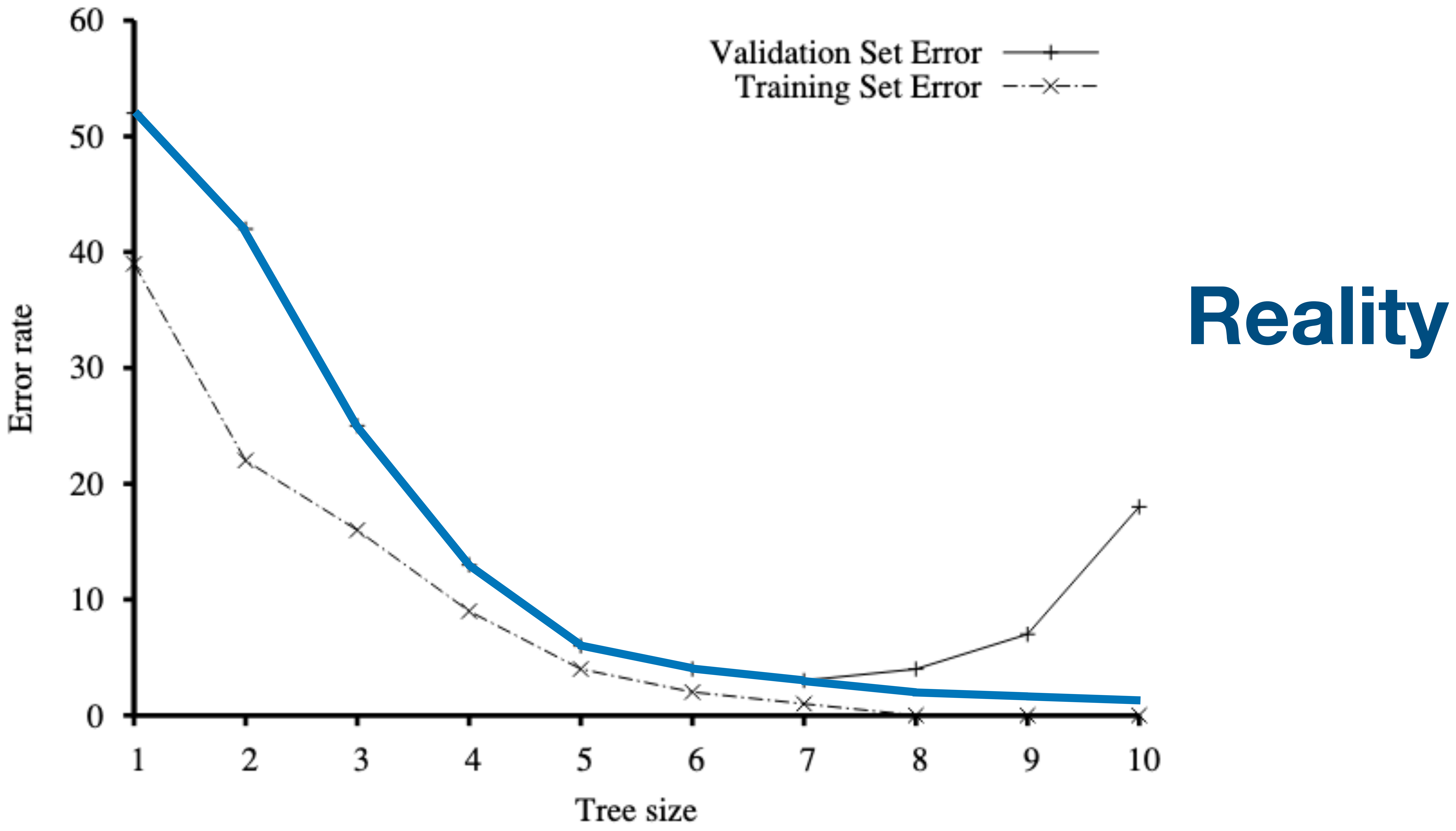


Figure 18.9 Error rates on training data (lower, dashed line) and validation data (upper, solid line) for different size decision trees. We stop when the training set error rate asymp- totes, and then choose the tree with minimal error on the validation set; in this case the tree

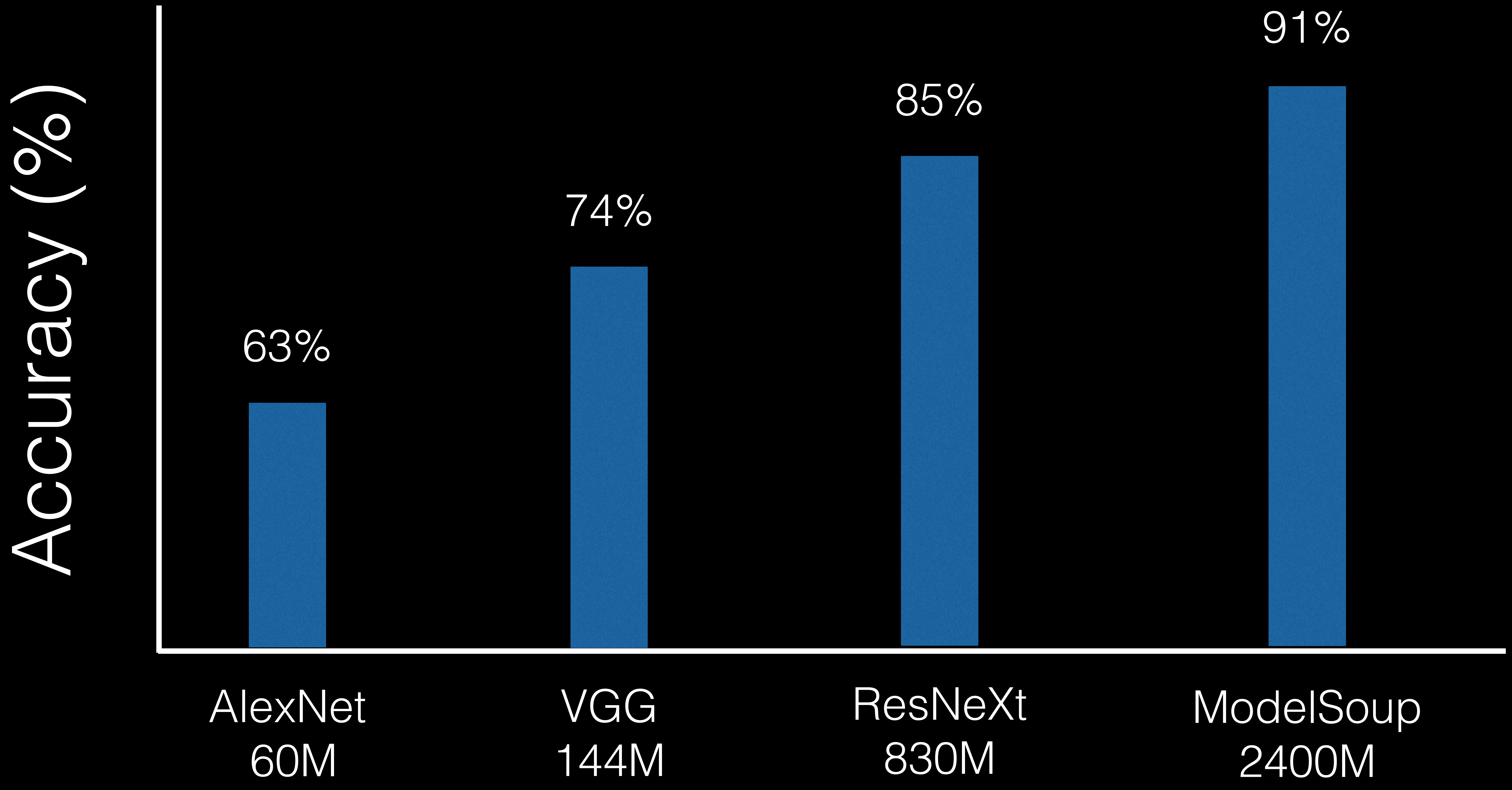
Why is machine learning inscrutable?

With four parameters I can fit an elephant, and with five I can make him wiggle his trunk.

- John Von Neumann

With four billion parameters
I can almost fit ImageNet.

- Modern ML Researchers


Accuracy (%)
63%
74%
85%
91%
AlexNet
60M
VGG
144M
ResNeXt
830M
ModelSoup
2400M

Things Fall Apart

Scaling Laws for Neural Language Models

Jared Kaplan *
Johns Hopkins University, OpenAI
jaredk@jhu.edu

Sam McCandlish*
OpenAI
sam@openai.com

Tom Henighan
OpenAI
henighan@openai.com

Tom B. Brown
OpenAI
tom@openai.com

Benjamin Chess
OpenAI
bchess@openai.com

Rewon Child
OpenAI
rewon@openai.com

Scott Gray
OpenAI
scott@openai.com

Alec Radford
OpenAI
alec@openai.com

Jeffrey Wu
OpenAI
jeffwu@openai.com

Dario Amodei
OpenAI
damodei@openai.com

Scaling Laws for Neural Language Models

But how are we going to get enough data to train these models?


Self-supervised machine
learning is the future
- Yann LeCun

What we currently understand:

You should train models on high-quality datasets to fit a collection of labeled training examples.

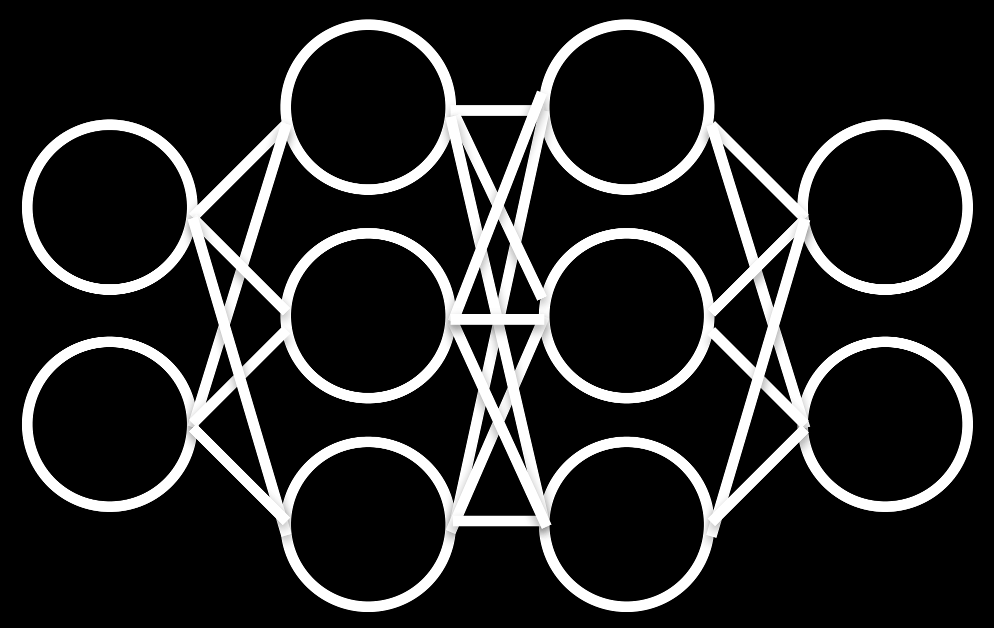
Self-supervised learning:

Take the last few things we *did* understand, and then stop doing them.

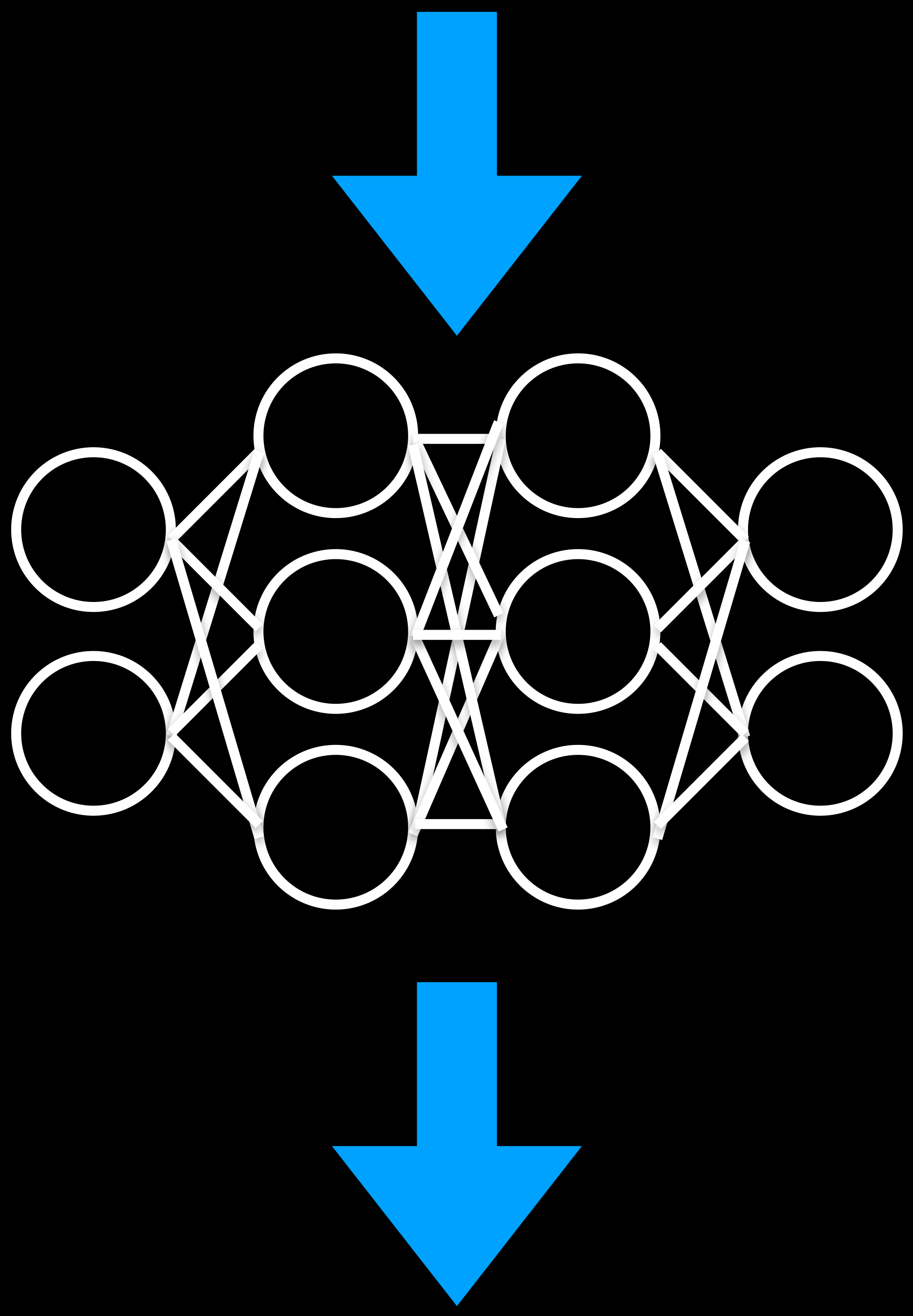
For example:

Sentiment Analysis

I loved this movie! It was the best movie I've ever seen in my life!	Positive
This was a total waste of time, there was nothing good at all.	Negative
This movie was entertaining, there wasn't anything bad about it.	Positive



I actually really liked this movie even though I heard bad things.



Positive

Self-supervised learning relies on "proxy tasks"

I loved this movie! It was the best movie I've ever seen in my life!	Positive
This was a total waste of time, there was nothing good at all.	Negative
This movie was entertaining, there wasn't anything bad about it.	Positive

I loved this movie! It was the best movie I've ever seen in my life!

This was a total waste of time, there was nothing good at all.

This movie was entertaining, there wasn't anything bad about it.

Radford, Jozefowicz, Sutskever. Learning to Generate Reviews and Discovering Sentiment. 2018.

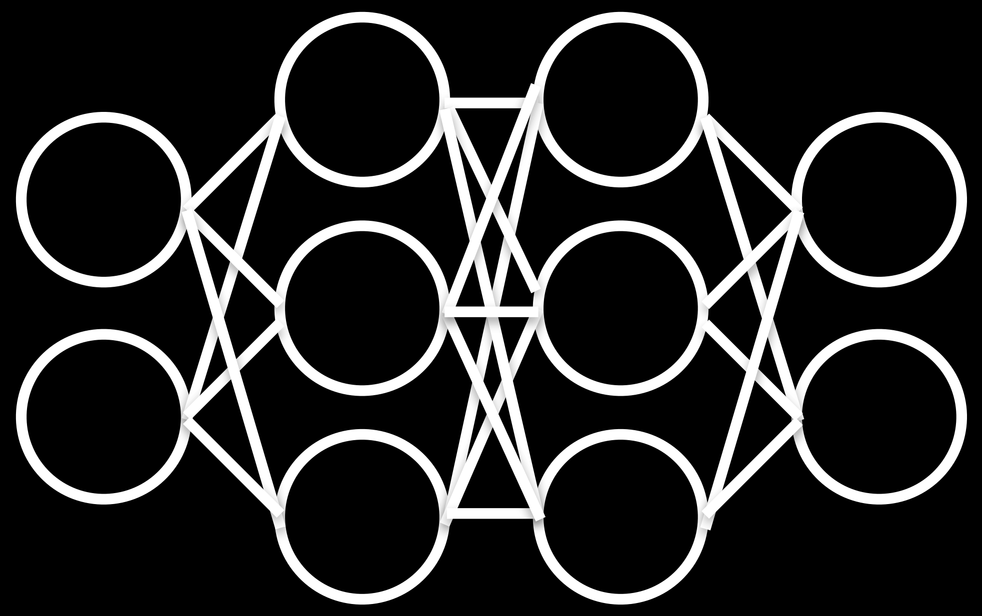
I loved this movie! It ____ the best
movie I've _________ in my life!

This was a total ________ time,
____ was nothing good at __.

This movie was ___________, there
wasn't ________ bad about it.

Radford, Jozefowicz, Sutskever. Learning to Generate Reviews and Discovering Sentiment. 2018.

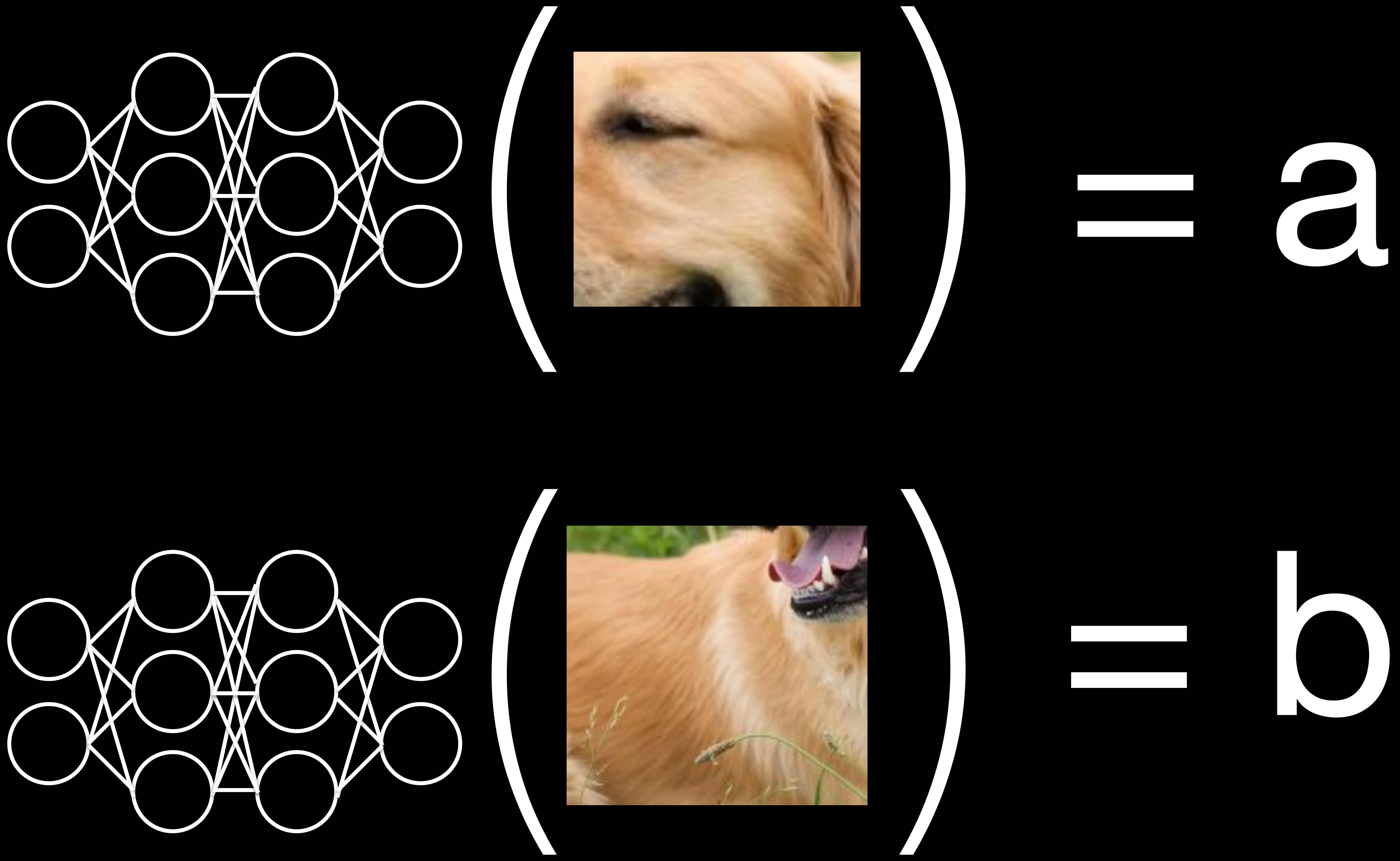
I actually really liked this movie even though I heard bad things. Overall my feelings are ______.



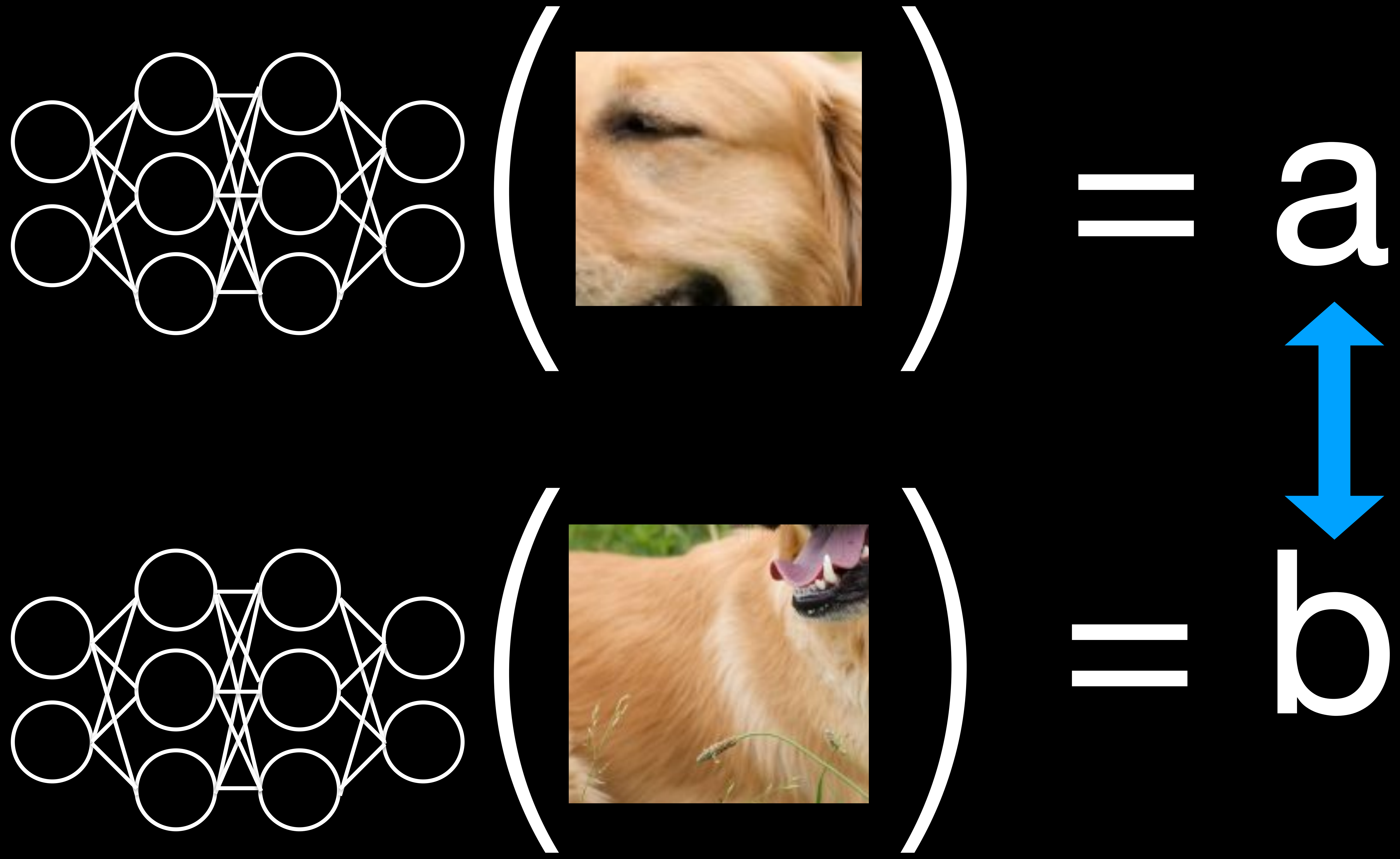
Positive

Chen et al. A Simple Framework for Contrastive Learning of Visual Representations. 2020.

Chen et al. A Simple Framework for Contrastive Learning of Visual Representations. 2020.

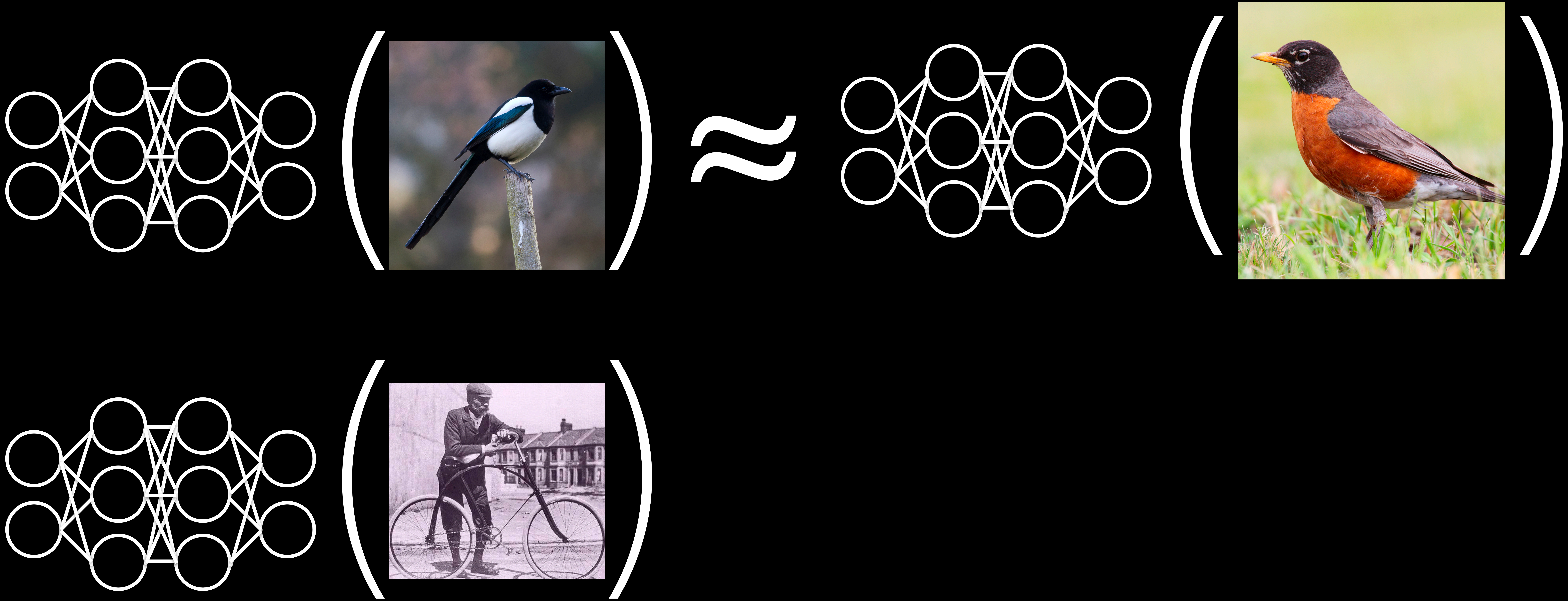


Chen et al. A Simple Framework for Contrastive Learning of Visual Representations. 2020.

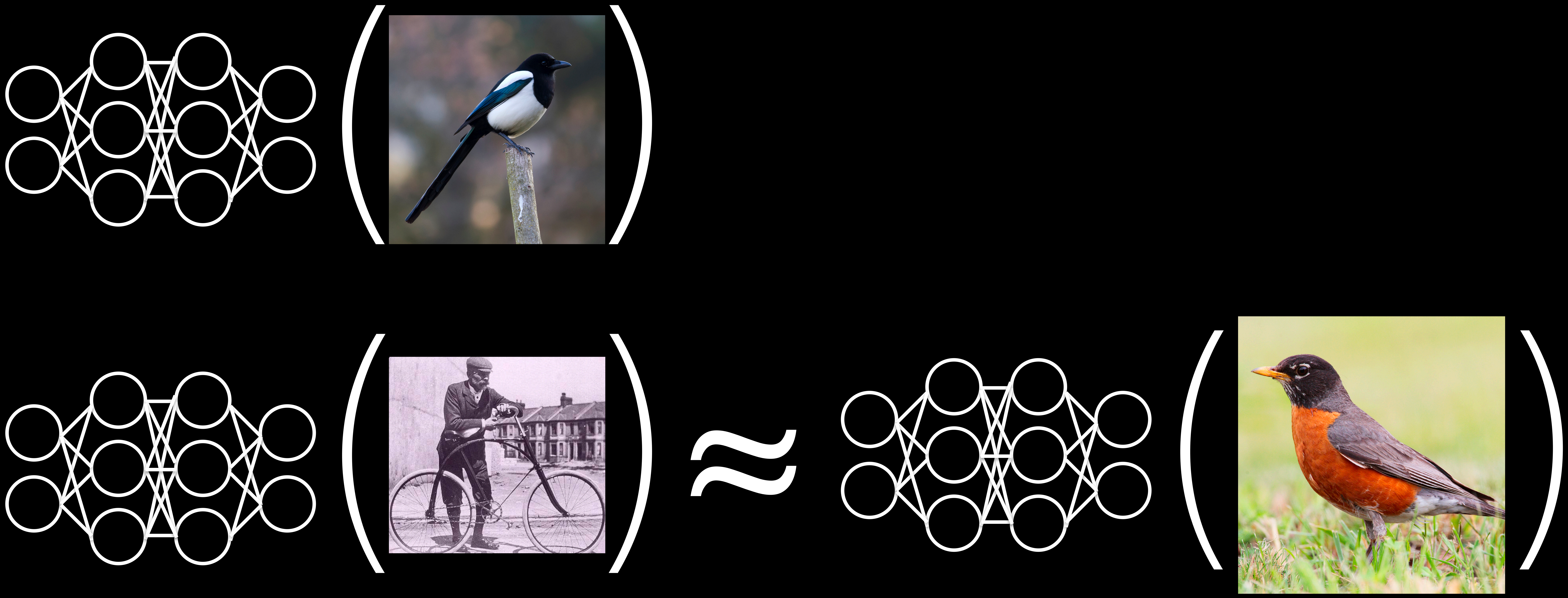


Chen et al. A Simple Framework for Contrastive Learning of Visual Representations. 2020.

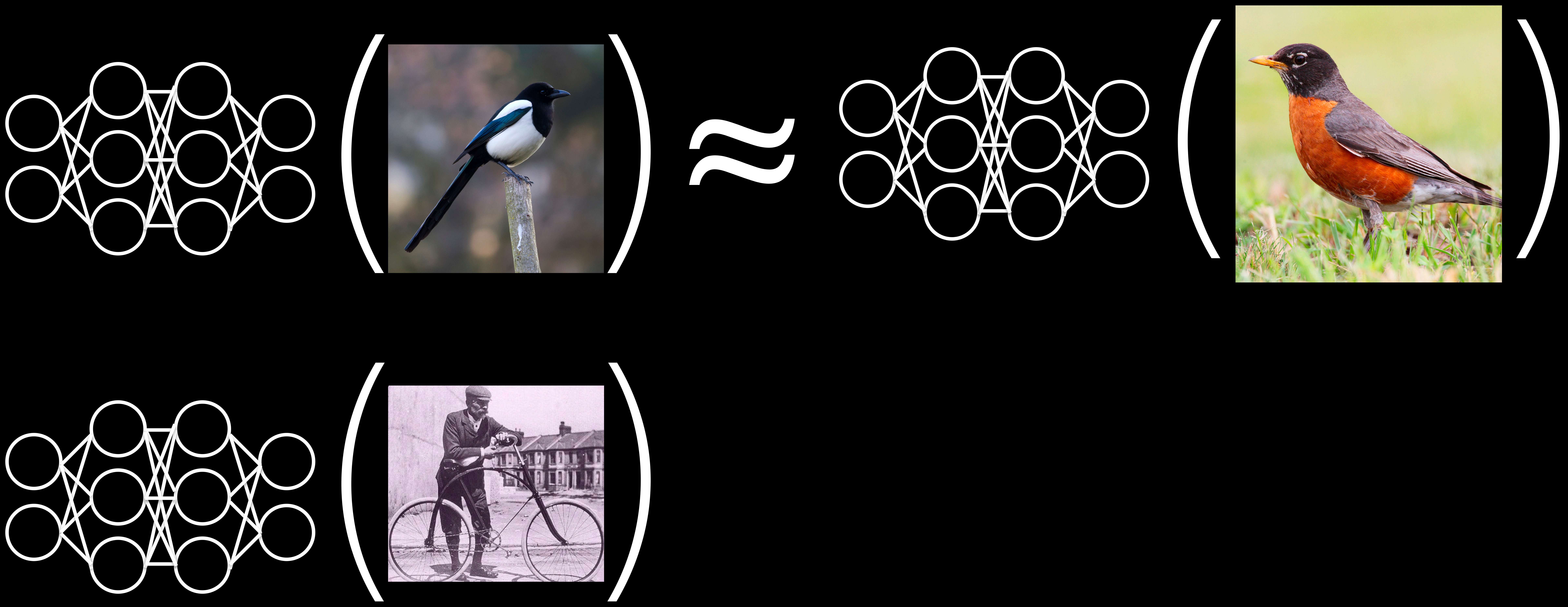
To classify:



To classify:

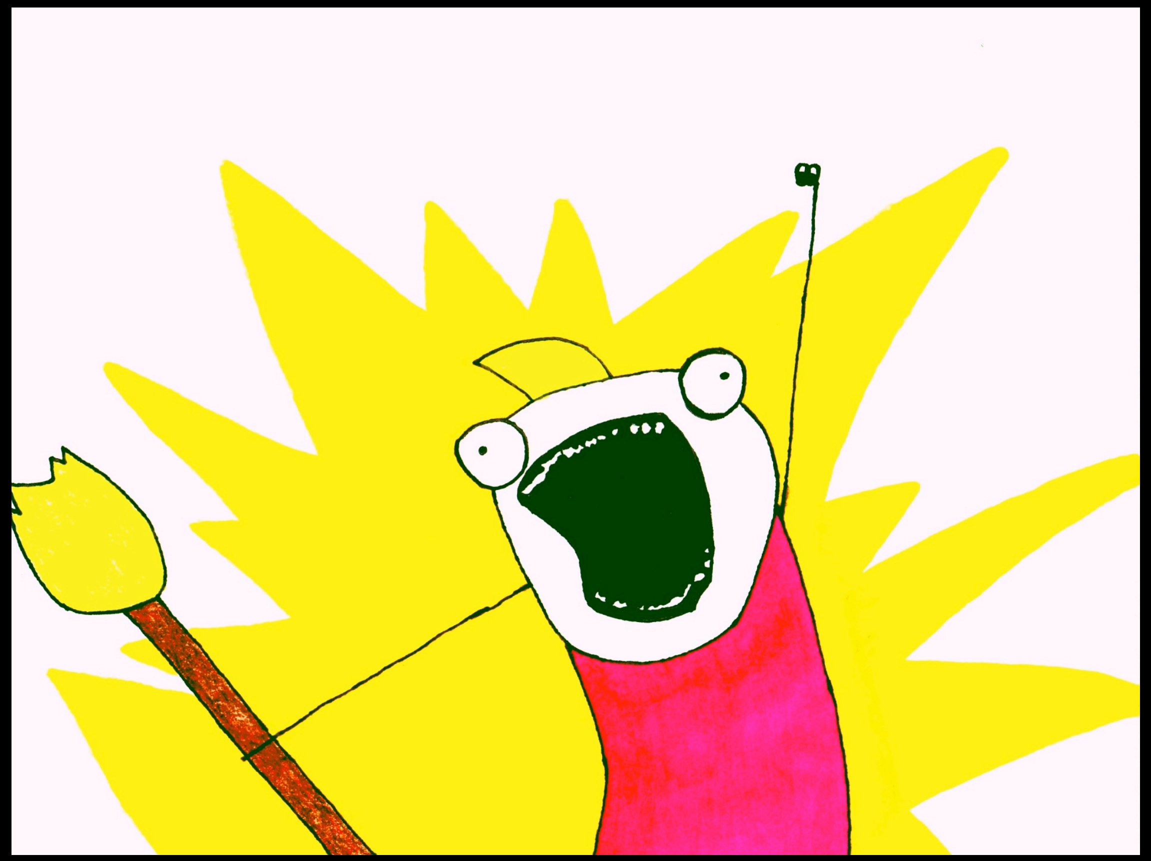


To classify:



To train a self-supervised model:

1. Crawl the internet
2. Collect ALL THE DATA!
3. Train on all of it



To train a self-supervised model:

1. Crawl the internet
2. Collect ALL THE DATA!
3. Train on all of it

Self-supervised learning:

Take the last few things we *did* understand,
and then stop doing them.

Labeled training data

~~Labeled training data~~

High-quality training data

High-quality training data

Even standard gradient descent begins to fail

ppl
13
12
11
10
9
8
Sat 13
Mon 15
Wed 17
Fri 19
Nov 21
Tue 23
Thu 25
Sat 27
Mon 29
December
Fri 03
Dec 05
Tue 07
Thu 09
Sat 11
Mon 13
Wed 15

Practically speaking what does this mean for reliability?

Poisoning attacks
become a real threat

Poisoning Attacks against Support Vector Machines

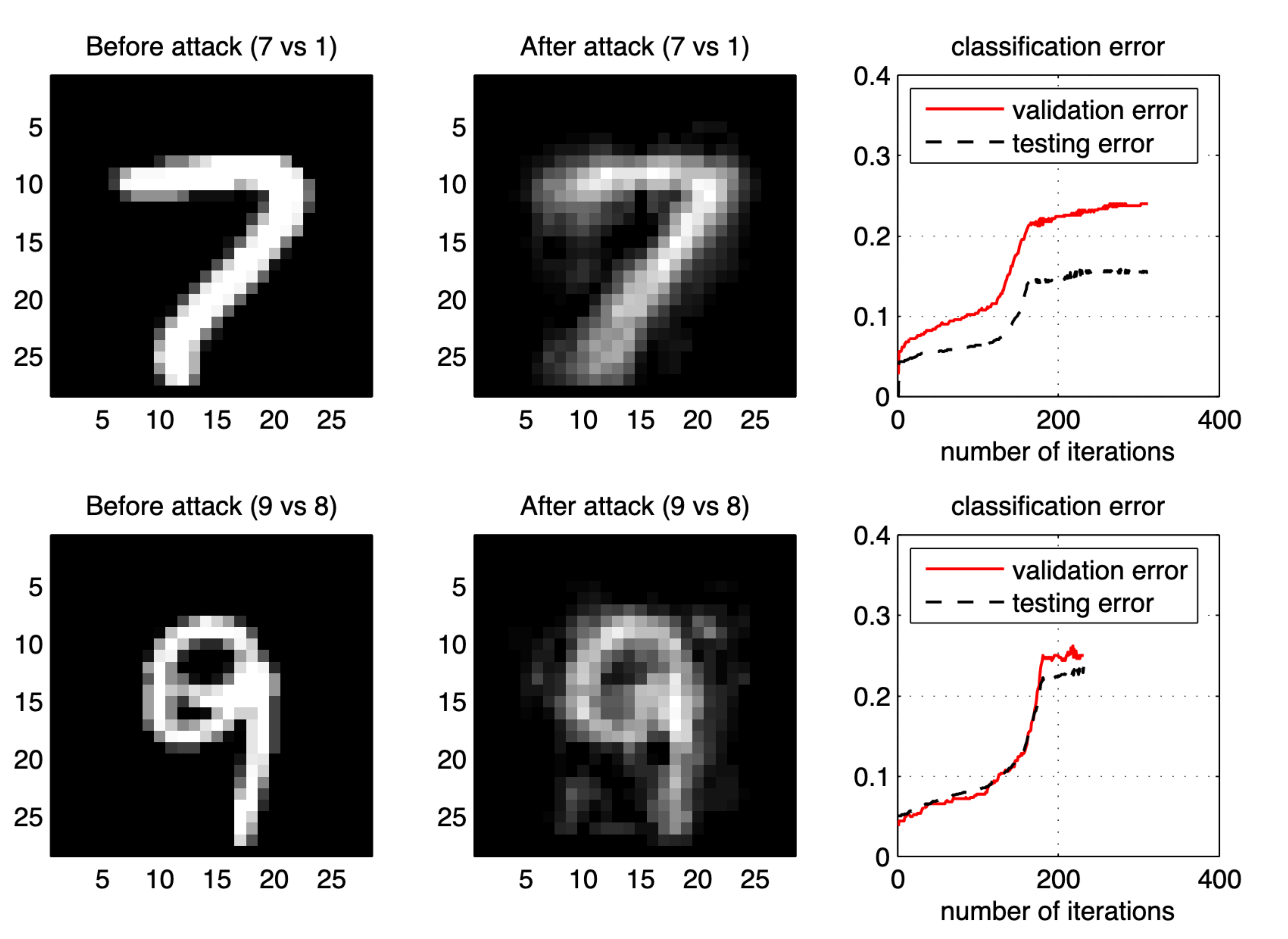
Battista Biggio BATTISTA.BIGGIO@DIEE.UNICA.IT
Department of Electrical and Electronic Engineering, University of Cagliari, Piazza d'Armi, 09123 Cagliari, Italy

Blaine Nelson BLAINE.NELSON@WSII.UNI-TUEBINGEN.DE
Pavel Laskov PAVEL.LASKOV@UNI-TUEBINGEN.DE
Wilhelm Schickard Institute for Computer Science, University of Tübingen, Sand 1, 72076 Tübingen, Germany

Poisoning Attacks against Support Vector Machines

Battista Biggio BATTISTA.BIGGIO@DIEE.UNICA.IT
Department of Electrical and Electronic Engineering, University of Cagliari, Piazza d'Armi, 09123 Cagliari, Italy

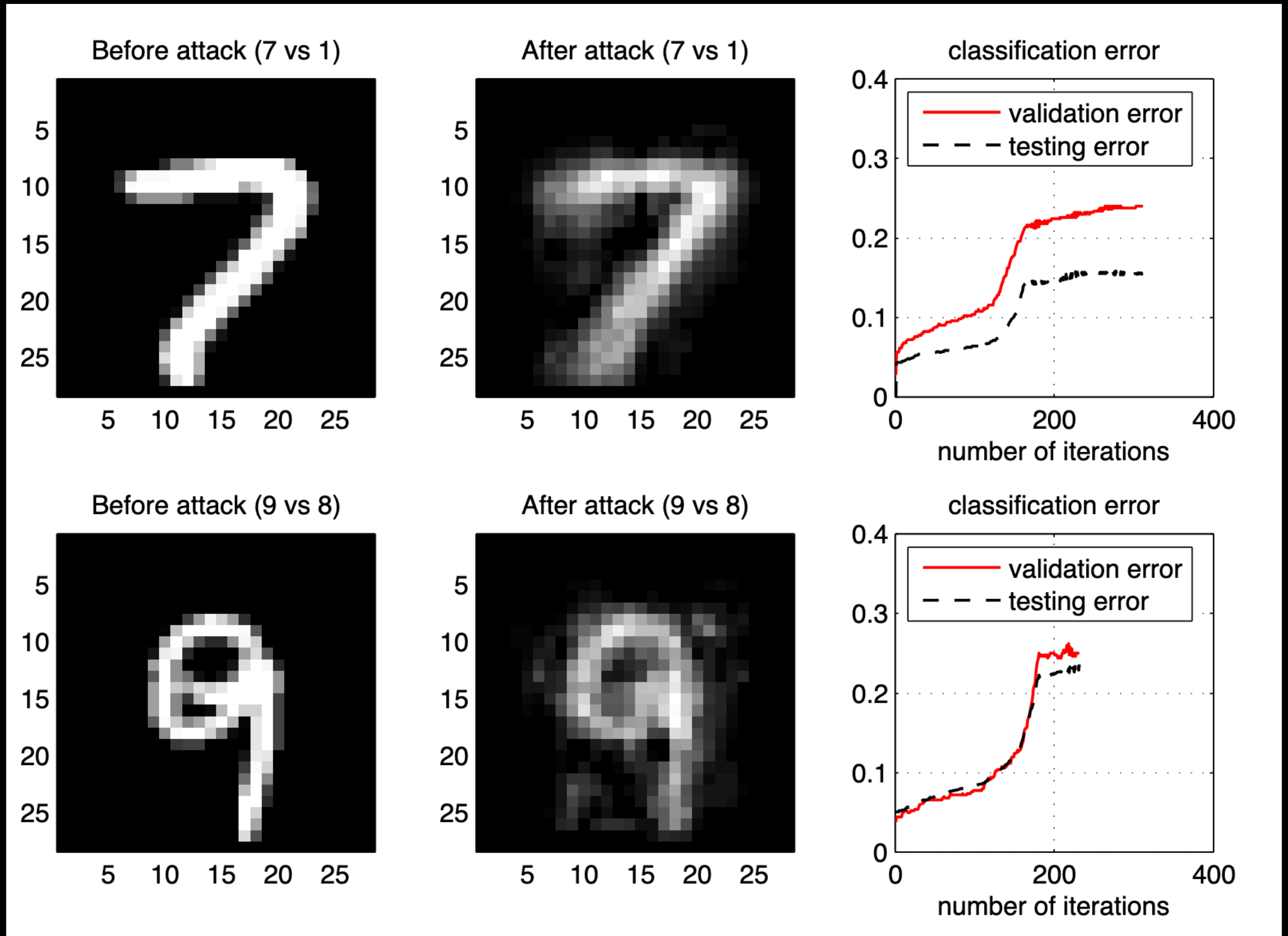
Blaine Nelson BLAINE.NELSON@WSII.UNI-TUEBINGEN.DE
Pavel Laskov PAVEL.LASKOV@UNI-TUEBINGEN.DE
Wilhelm Schickard Institute for Computer Science, University of Tübingen, Sand 1, 72076 Tübingen, Germany

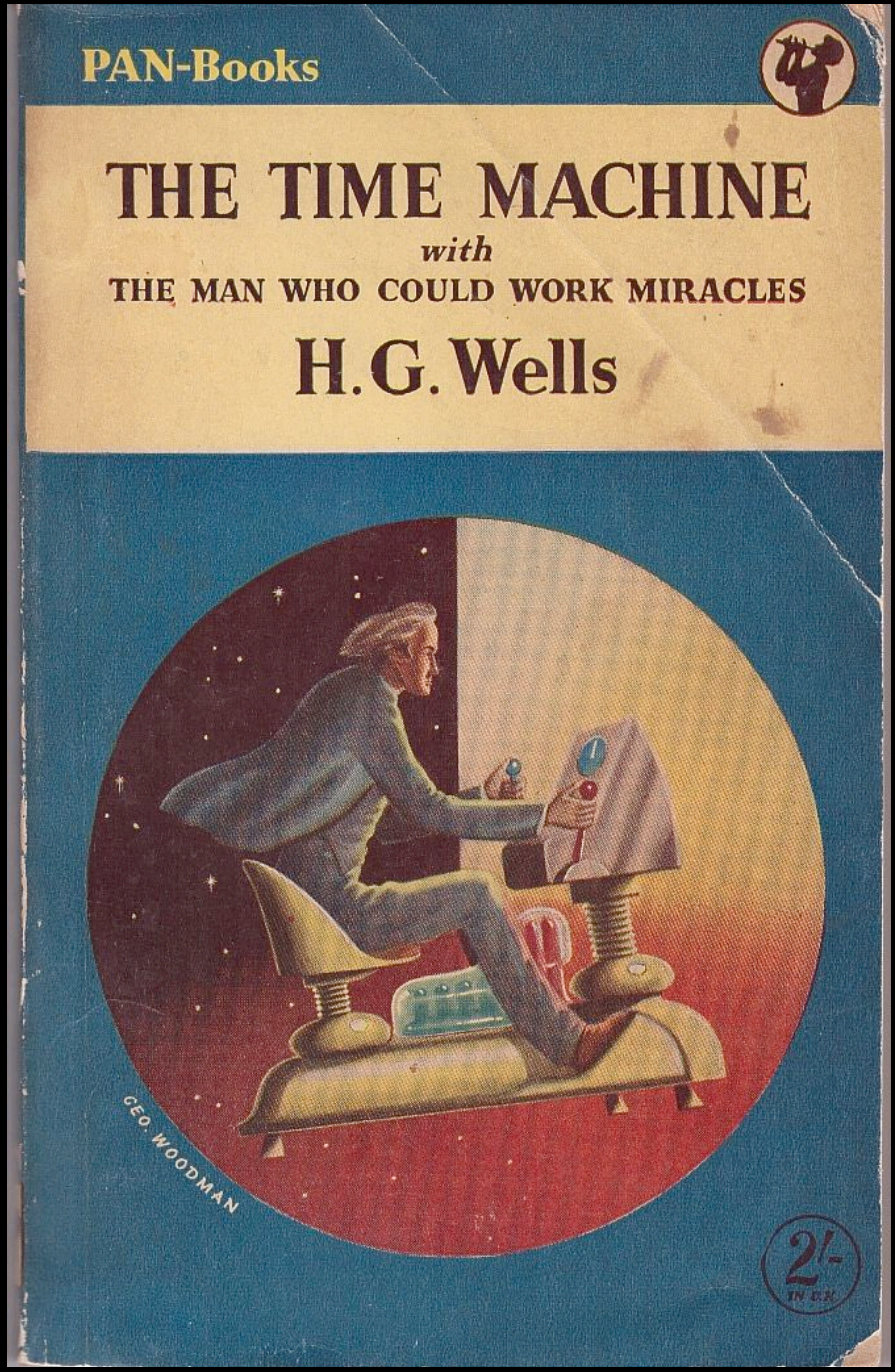


Poisoning Attacks against Support Vector Machines

Battista Biggio BATTISTA.BIGGIO@DIEE.UNICA.IT
Department of Electrical and Electronic Engineering, University of Cagliari, Piazza d'Armi, 09123 Cagliari, Italy

Blaine Nelson BLAINE.NELSON@WSII.UNI-TUEBINGEN.DE
Pavel Laskov PAVEL.LASKOV@UNI-TUEBINGEN.DE
Wilhelm Schickard Institute for Computer Science, University of Tübingen, Sand 1, 72076 Tübingen, Germany





Scaling Up Visual and Vision-Language Representation Learning With Noisy Text Supervision

Chao Jia [1] **Yinfei Yang** [1] **Ye Xia** [1] **Yi-Ting Chen** [1] **Zarana Parekh** [1] **Hieu Pham** [1] **Quoc V. Le** [1]
Yunhsuan Sung [1] **Zhen Li** [1] **Tom Duerig** [1]

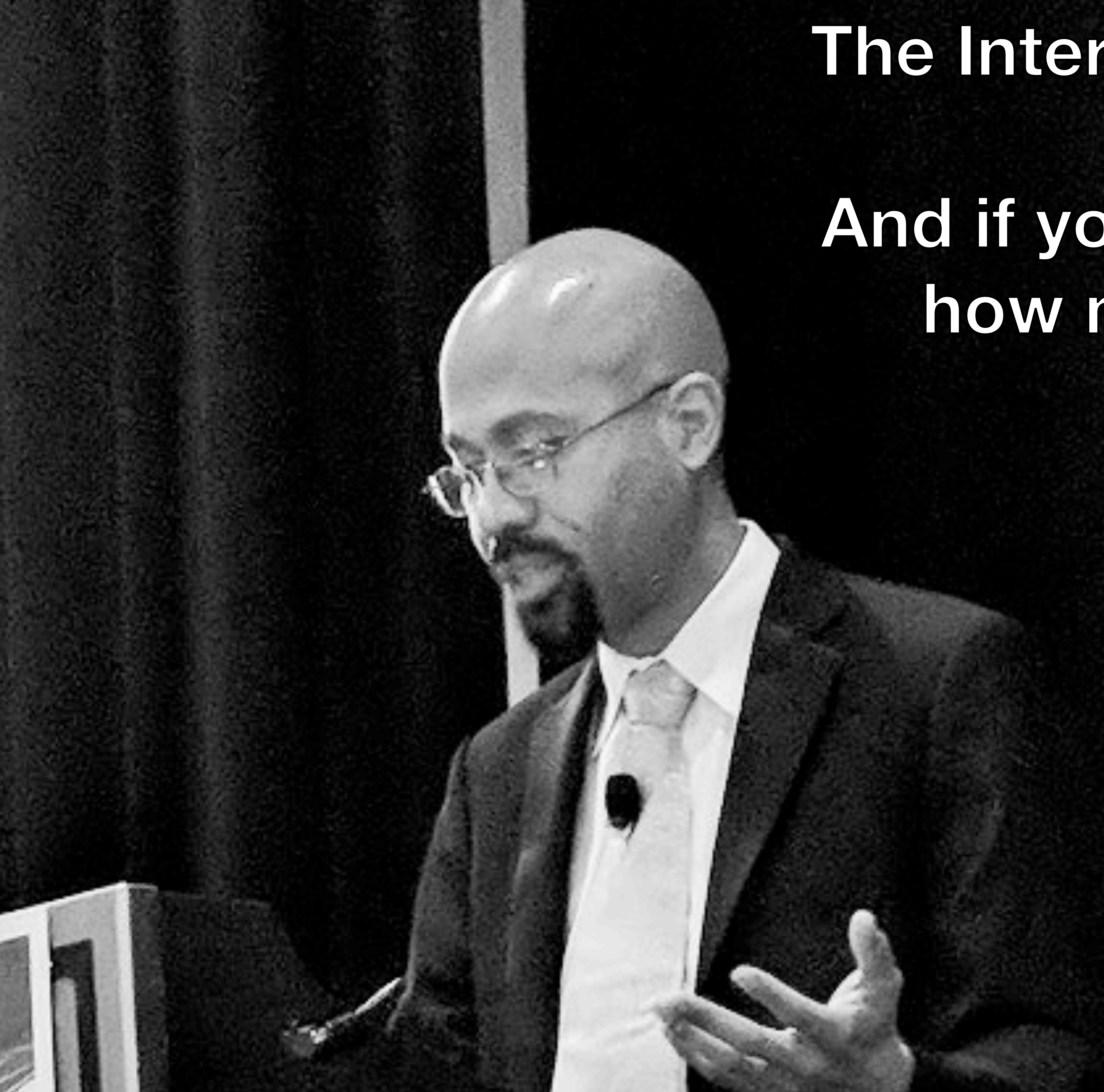
Divide and Contrast: Self-supervised Learning from Uncurated Data

Yonglong Tian [*]
MIT

Olivier J. Hénaff
DeepMind

Aaron van den Oord
DeepMind

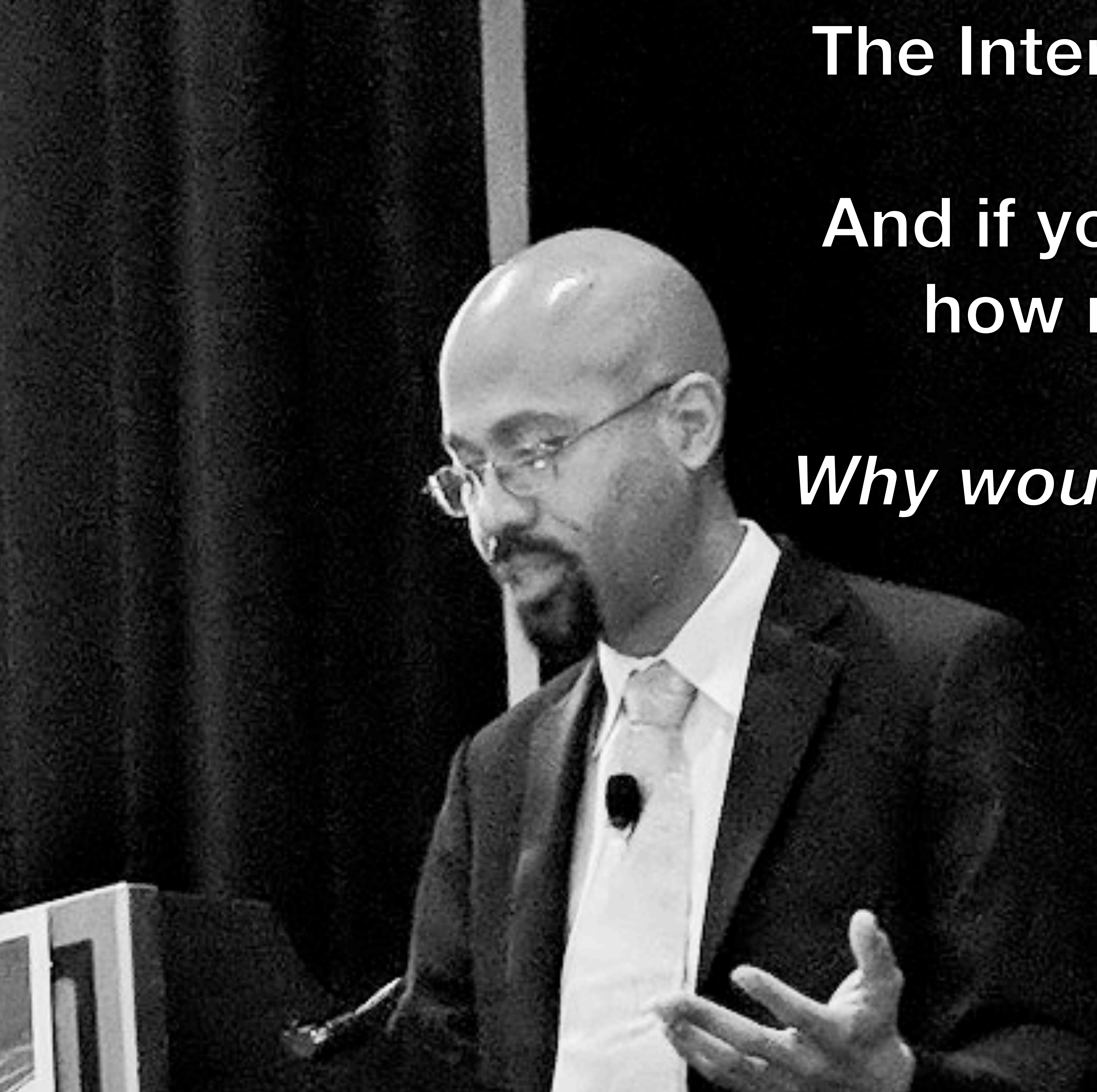

The Internet is a cauldron of evil,
- James Mickens



The Internet is a cauldron of evil,

And if you don't fully understand how machine learning works,

- James Mickens



The Internet is a cauldron of evil,

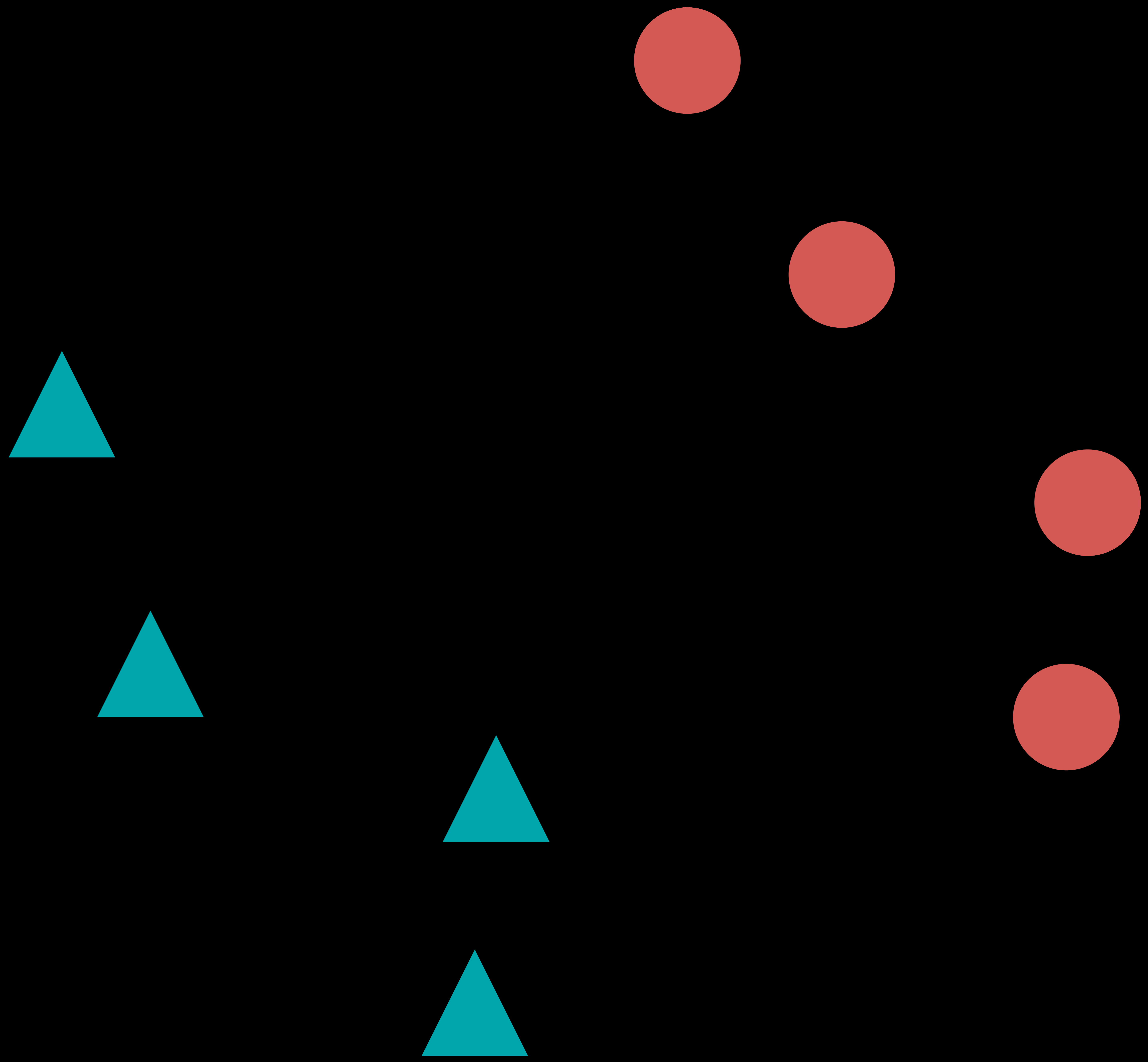
And if you don't fully understand how machine learning works,

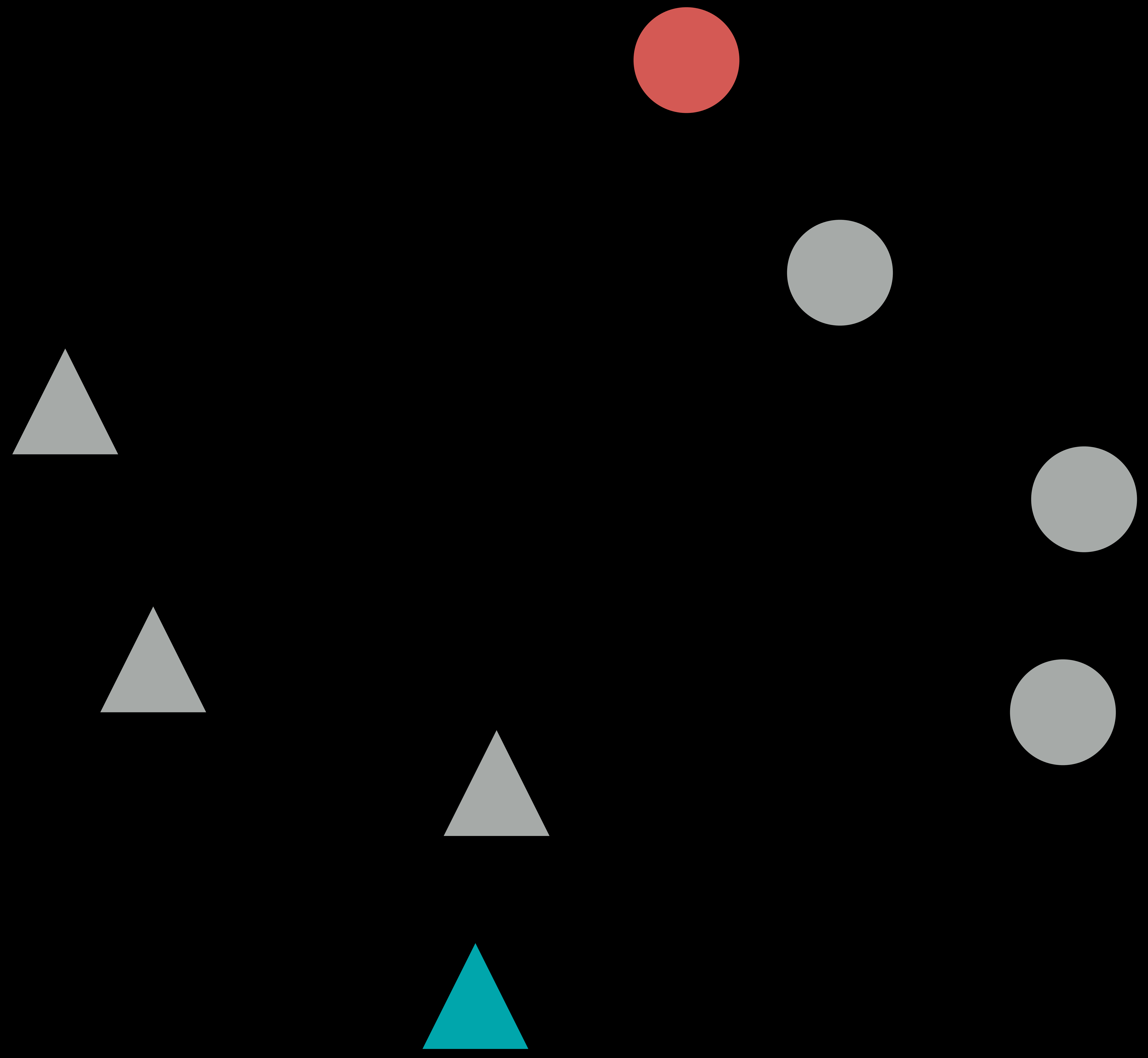
Why would you connect the two?

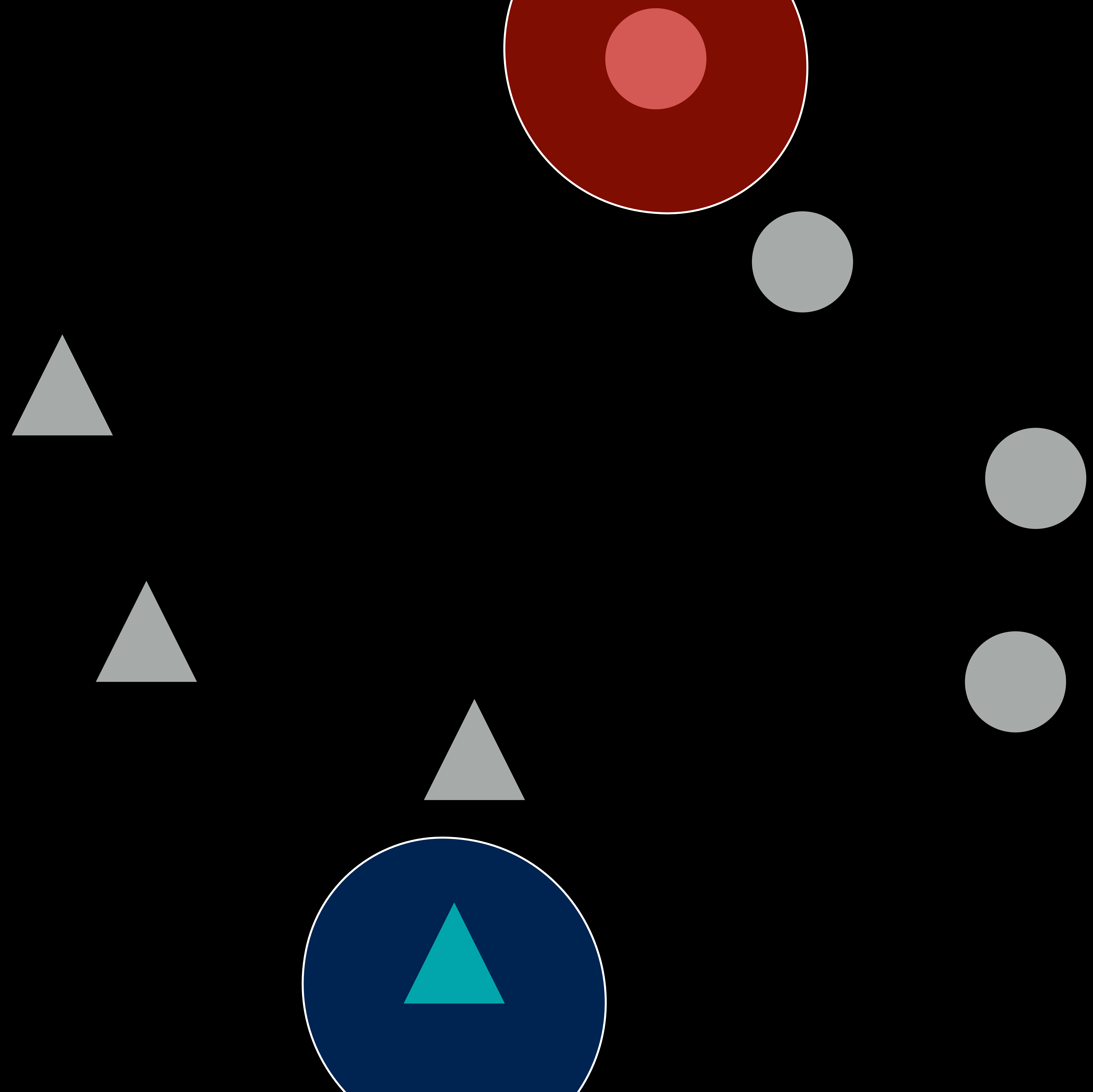
\- James Mickens

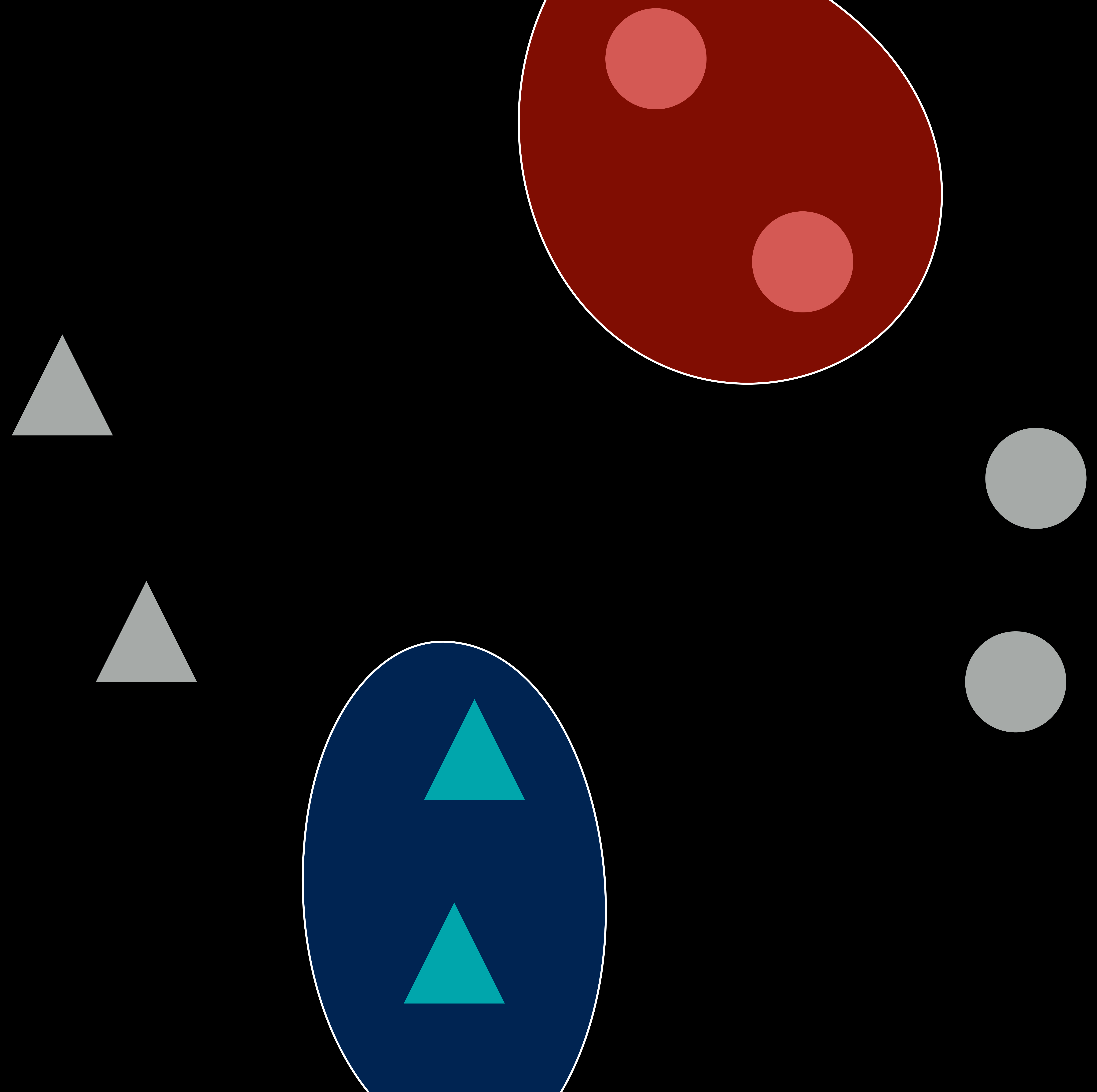
For example: Poisoning Semi-Supervised Learning

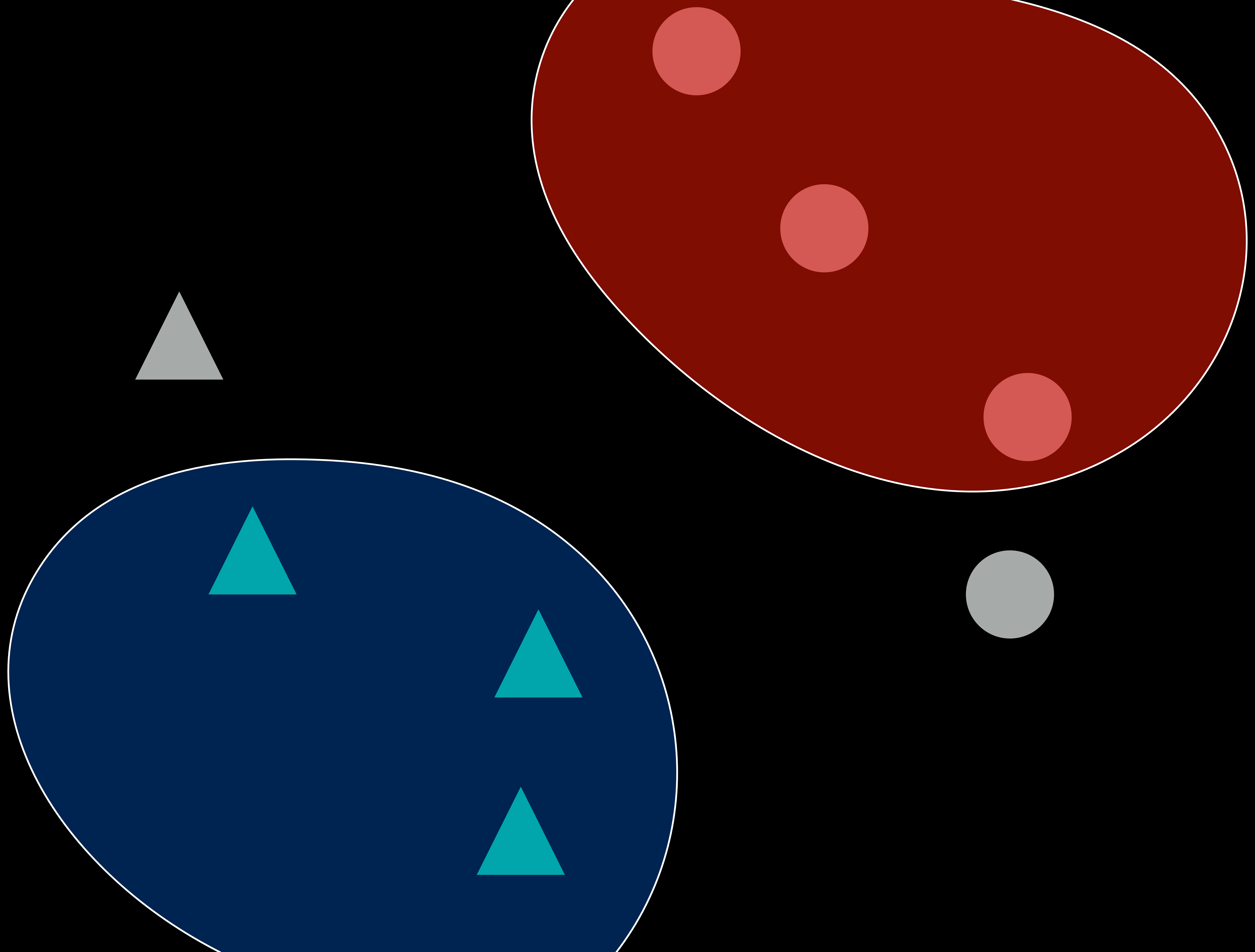
Carlini. Poisoning the Unlabeled Dataset of Semi-Supervised Learning. 2021.



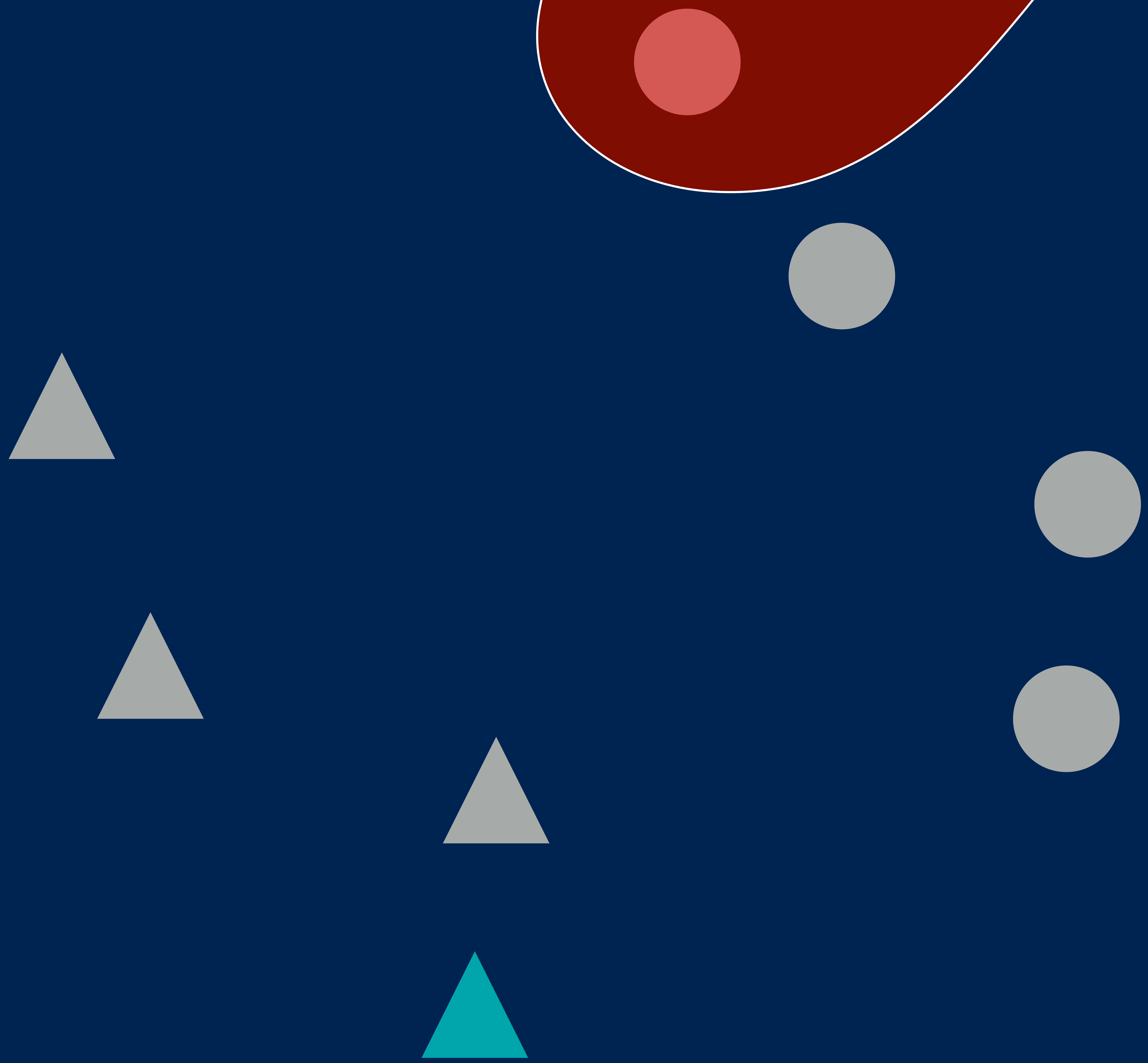








Why does the model learn this decision boundary?



Why does the model learn this decision boundary?

We don't know

Poisoning will be easy

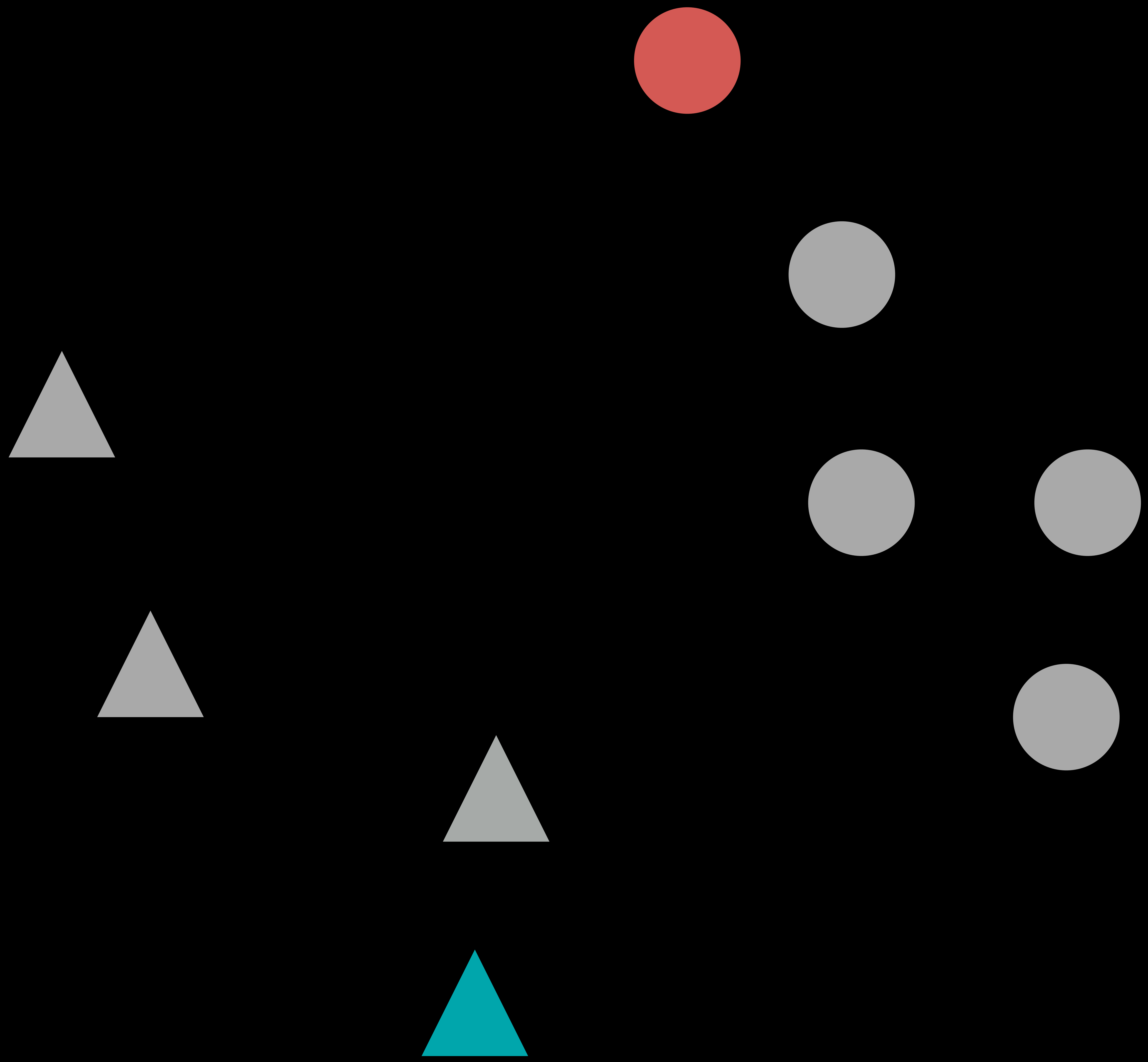

Desired
Error

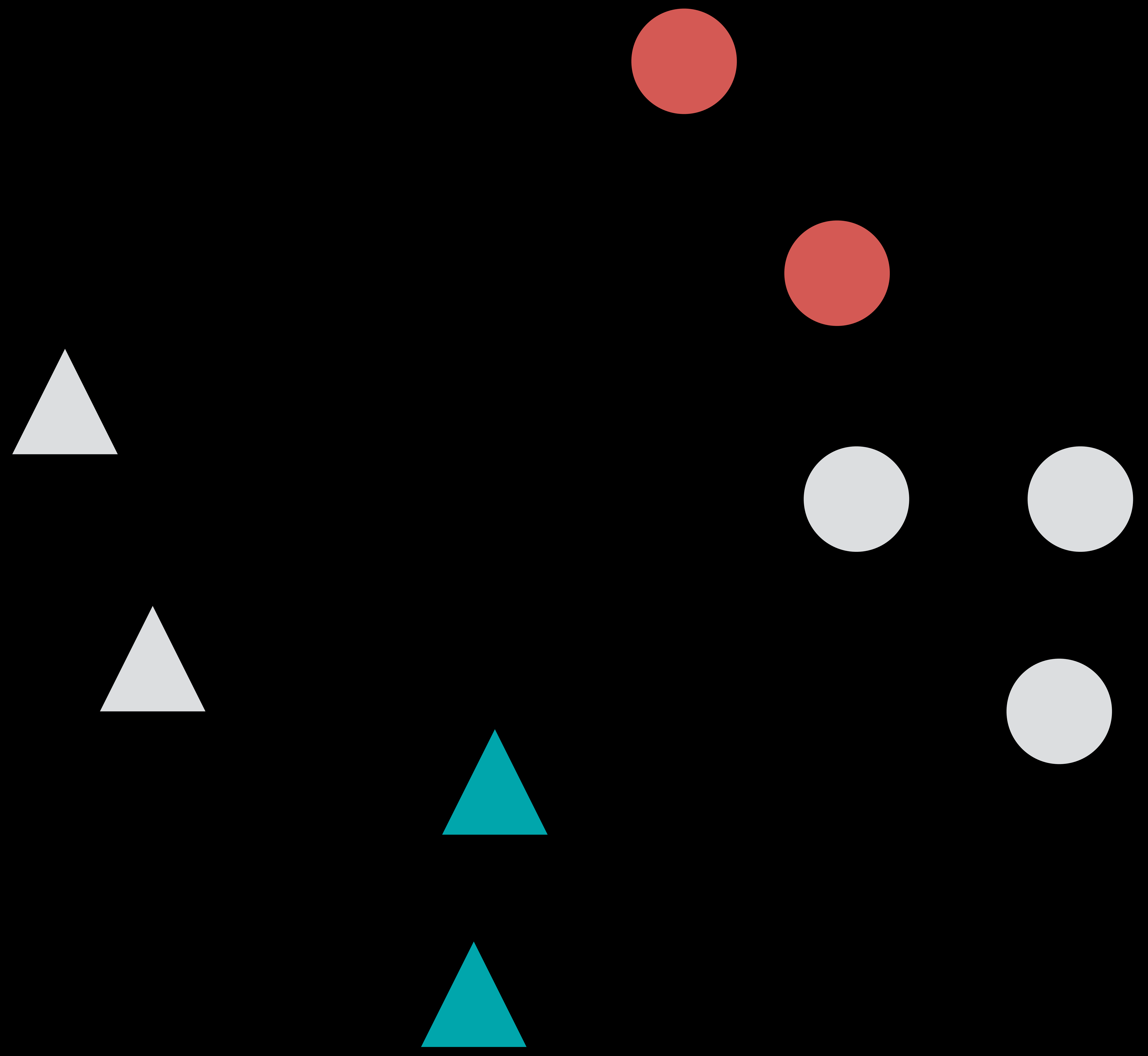
If we could insert *labeled* data

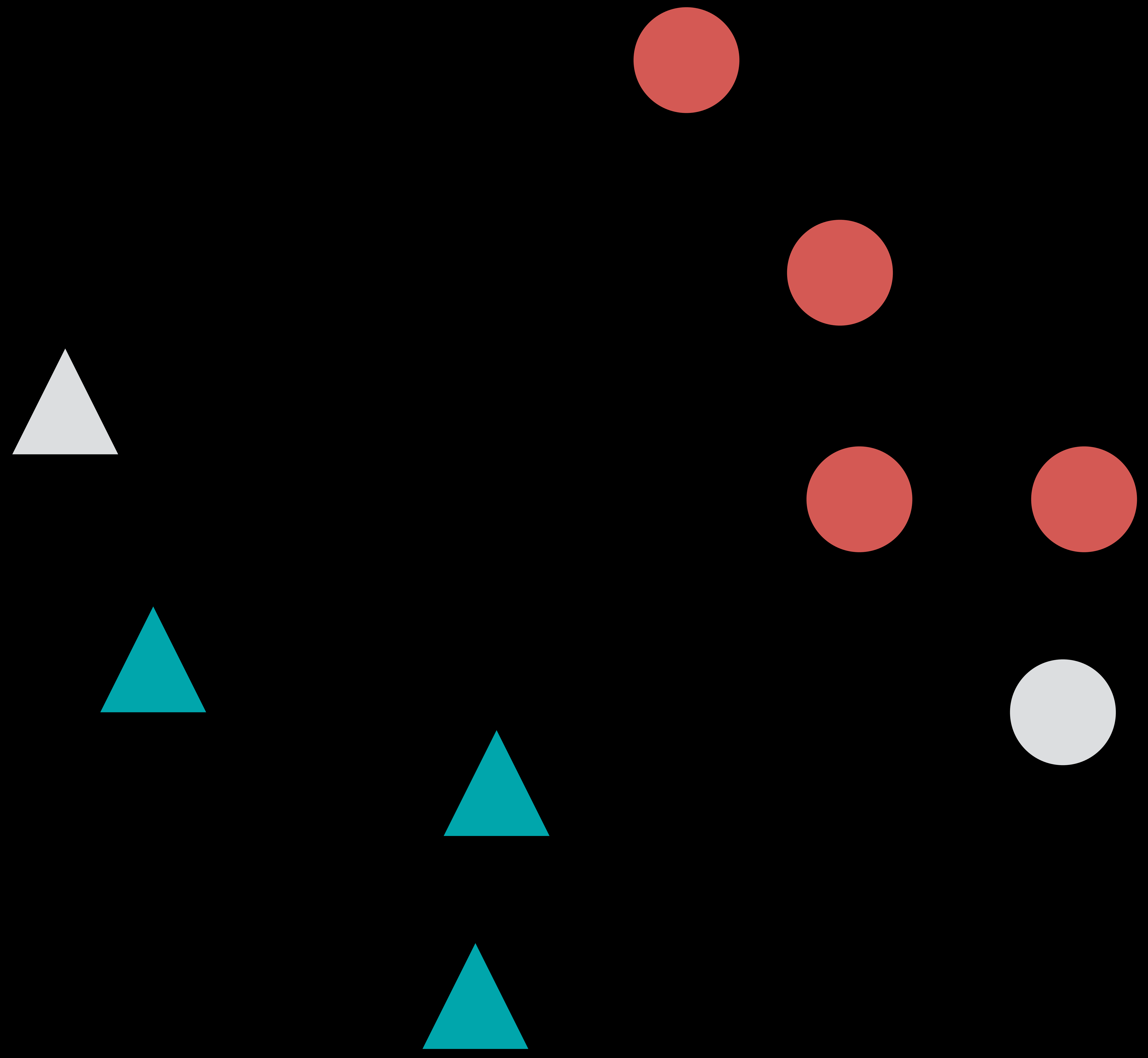

Poisoned
Sample

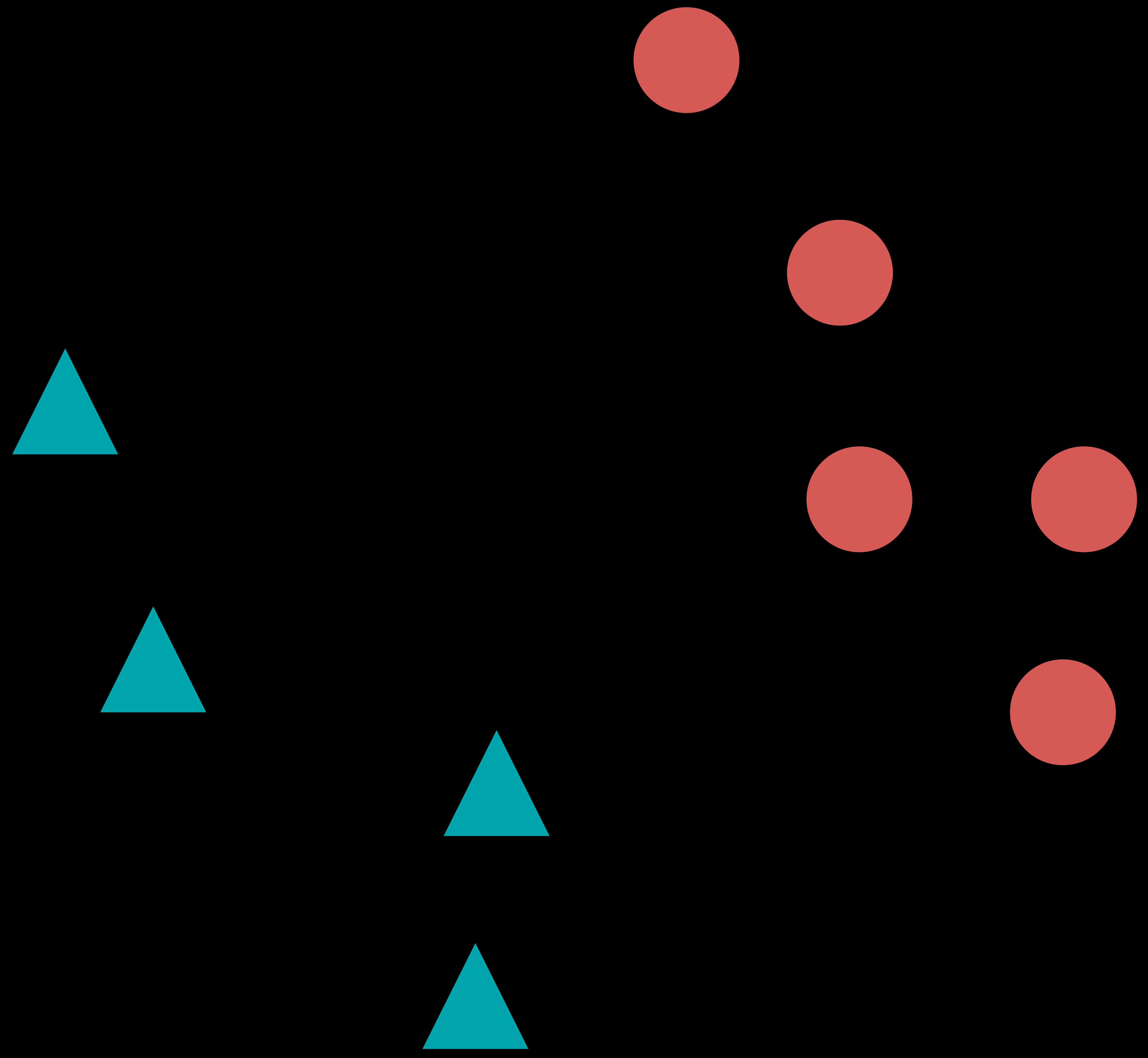

Desired
Error

but if we added *unlabeled* data …

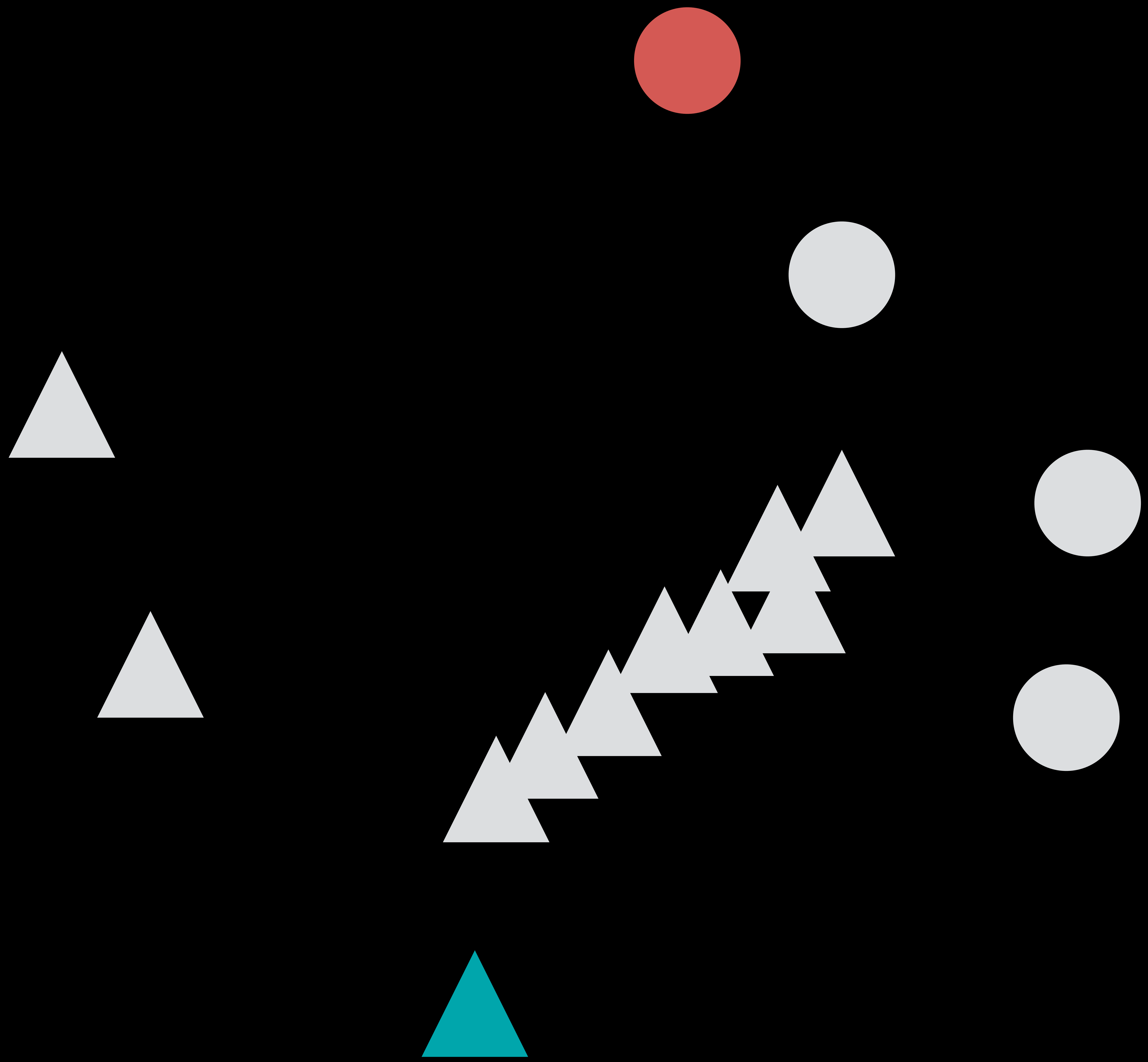


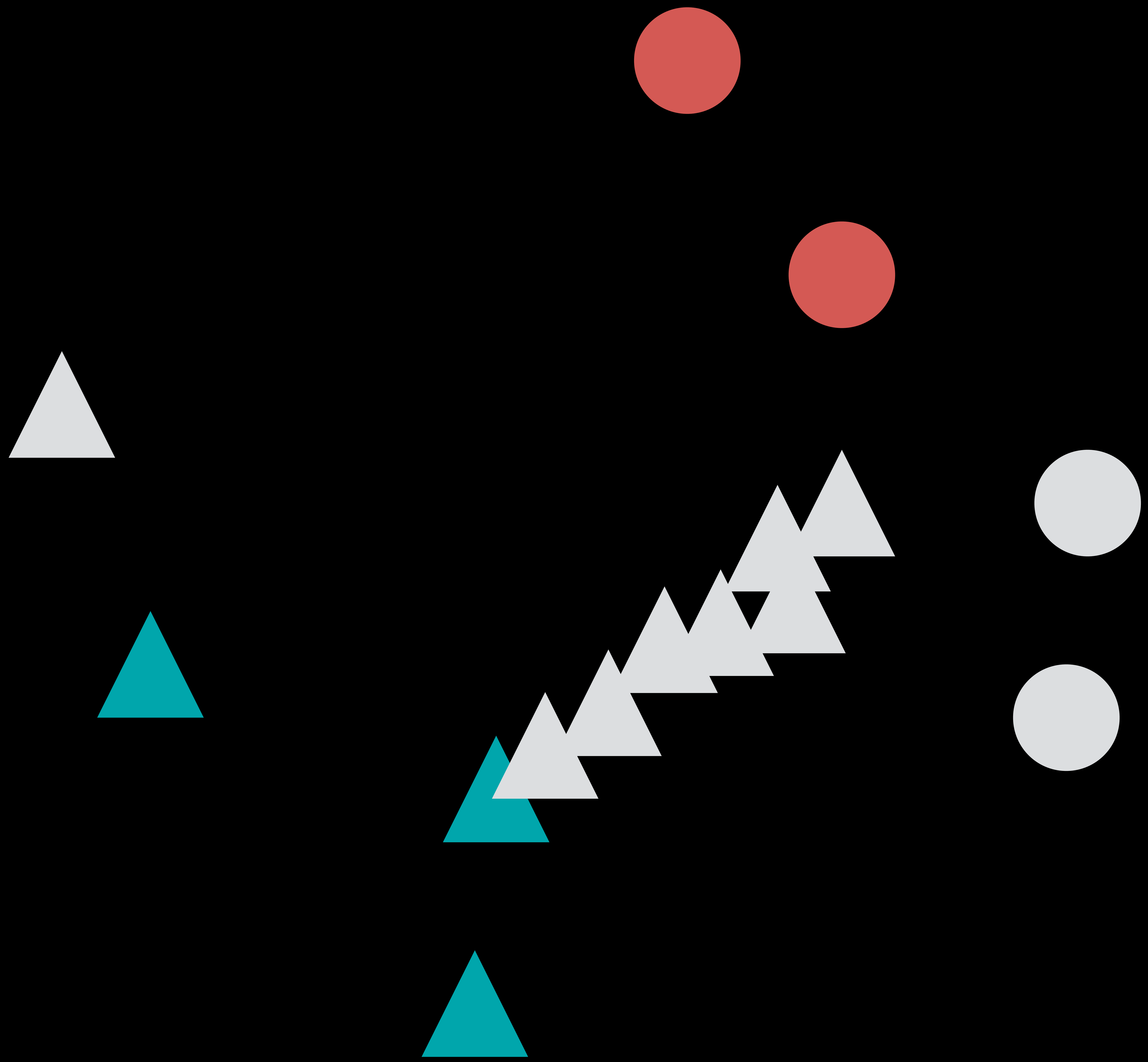


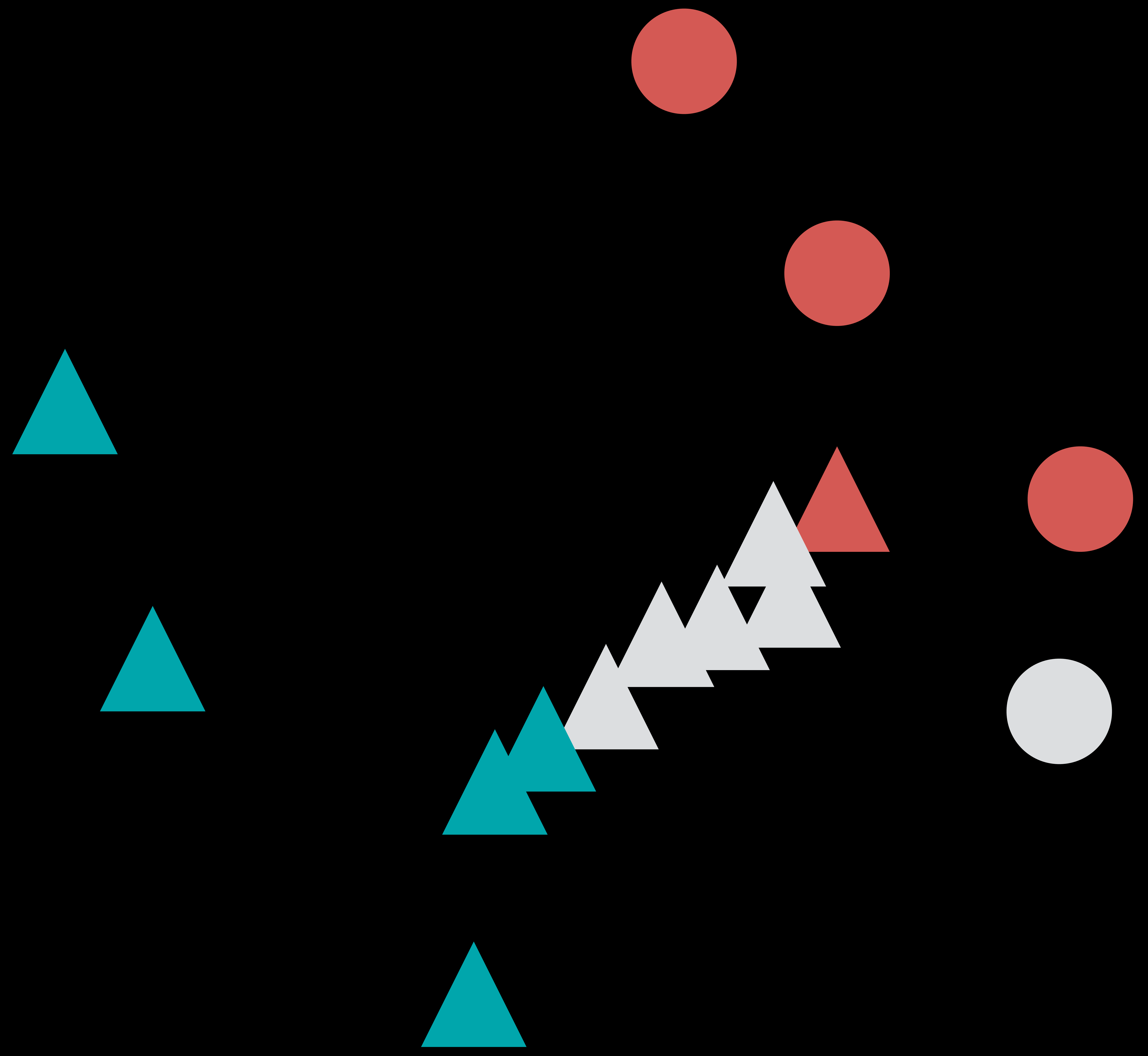


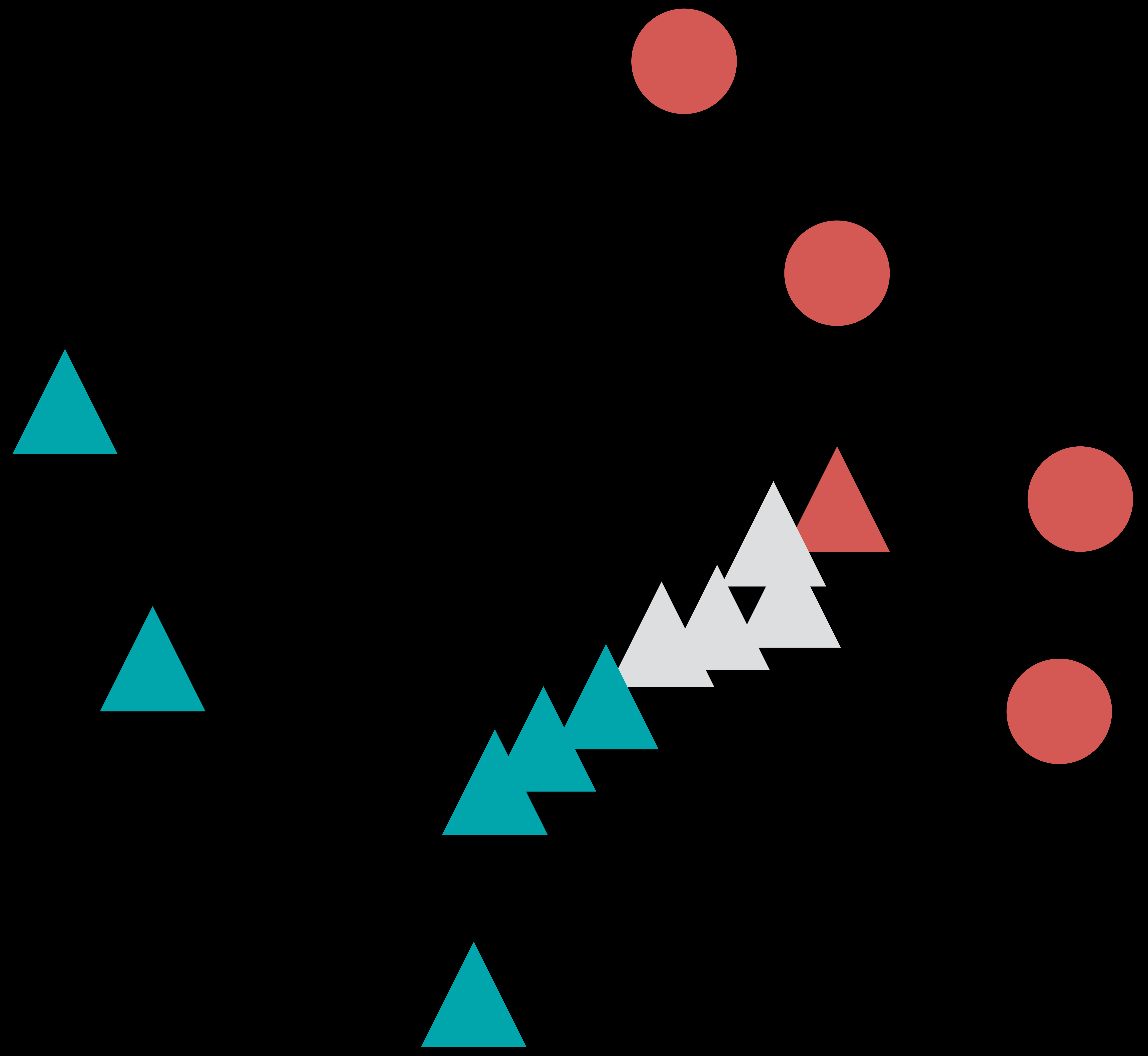


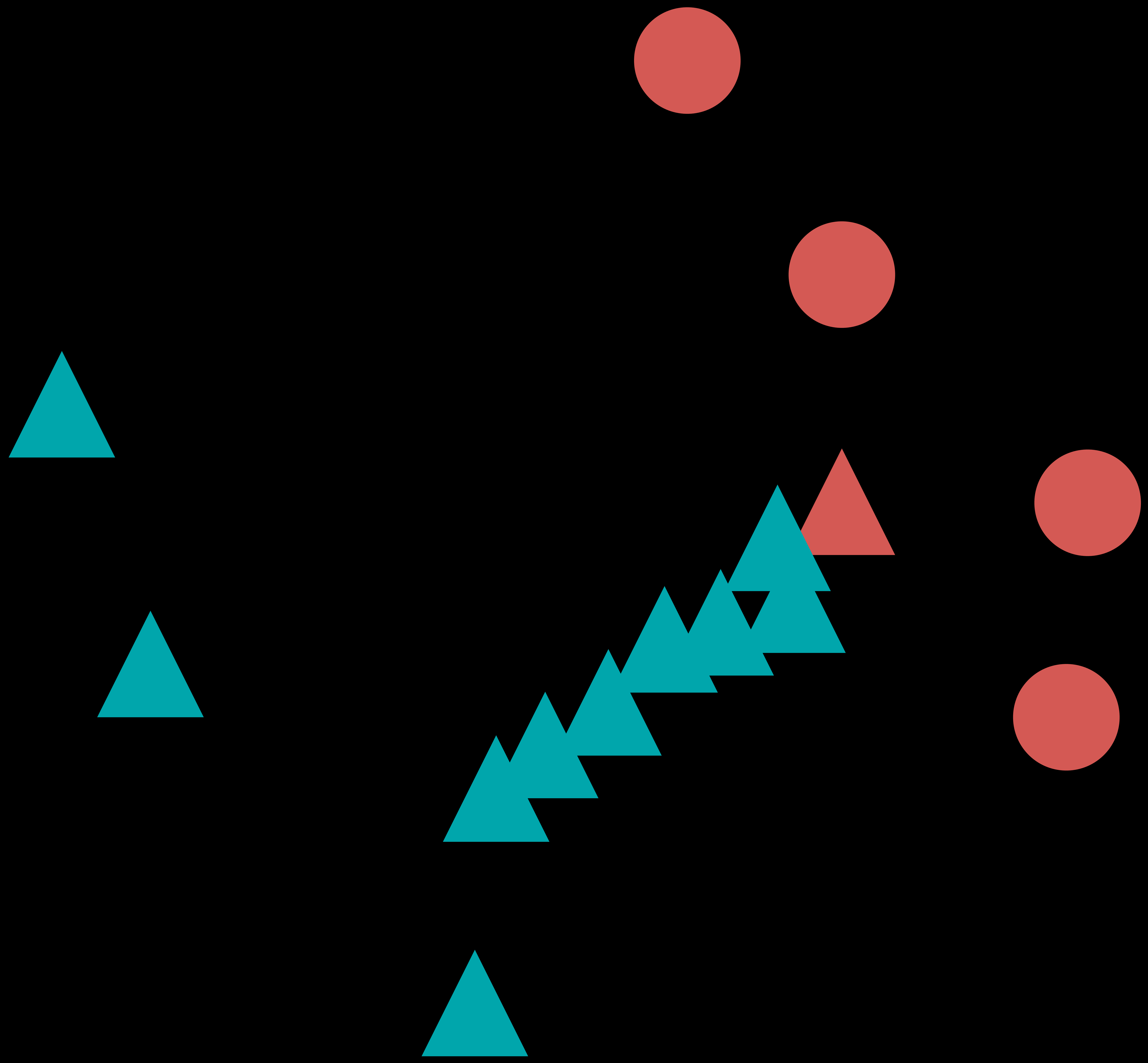
instead ...

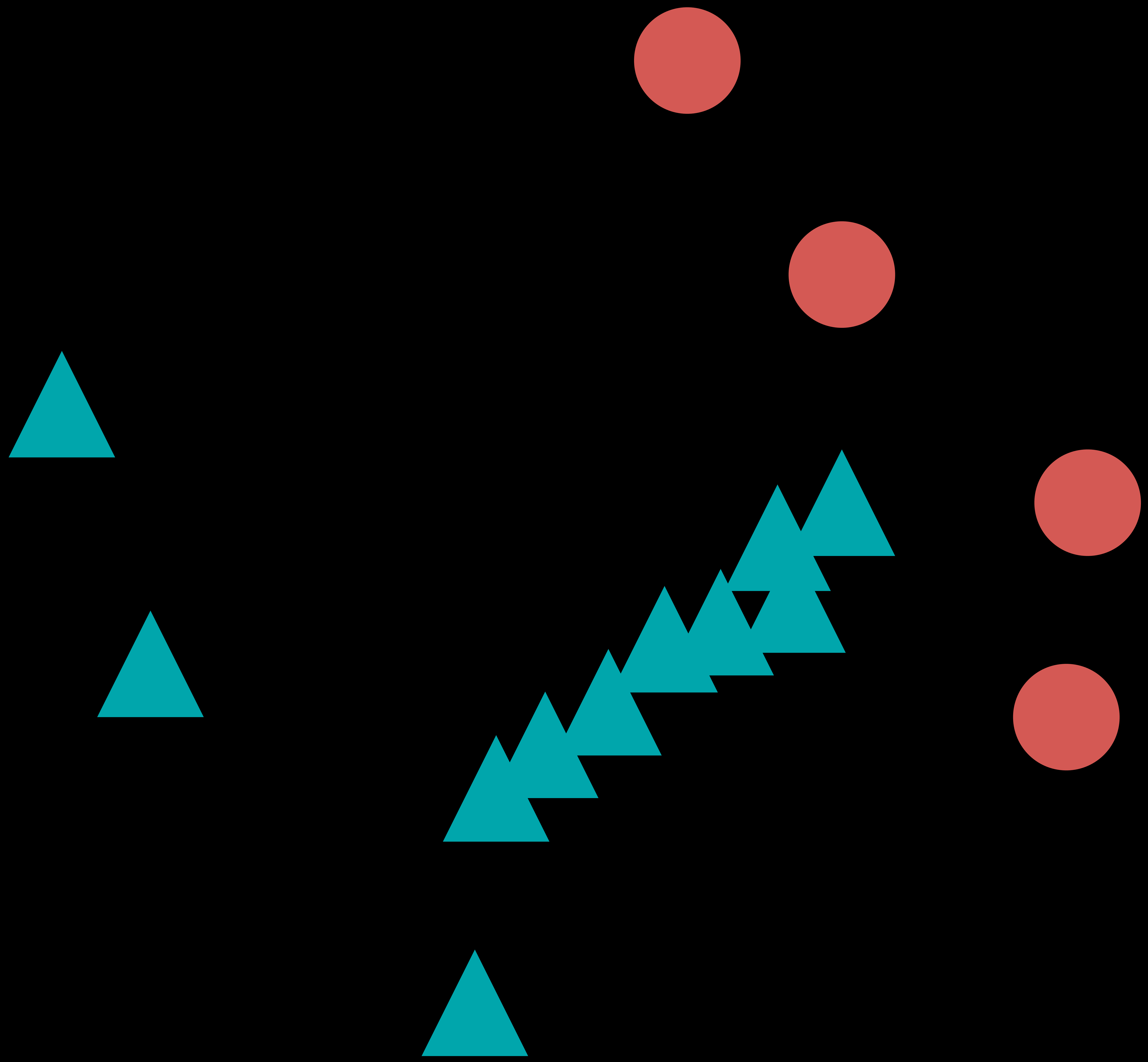













Success!

How much poisoning data?

How much poisoning data?

Fully supervised learning: 1%

How much poisoning data?

Fully supervised learning: 1%

Semi-supervised learning: 0.1%

How much poisoning data?

Fully supervised learning: 1%

Semi-supervised learning: 0.1%

Self-supervised learning:

How much poisoning data?

Fully supervised learning: 1%

Semi-supervised learning: 0.1%

Self-supervised learning: 0.00001%

Conclusion

Lessons for the Future of Machine Learning

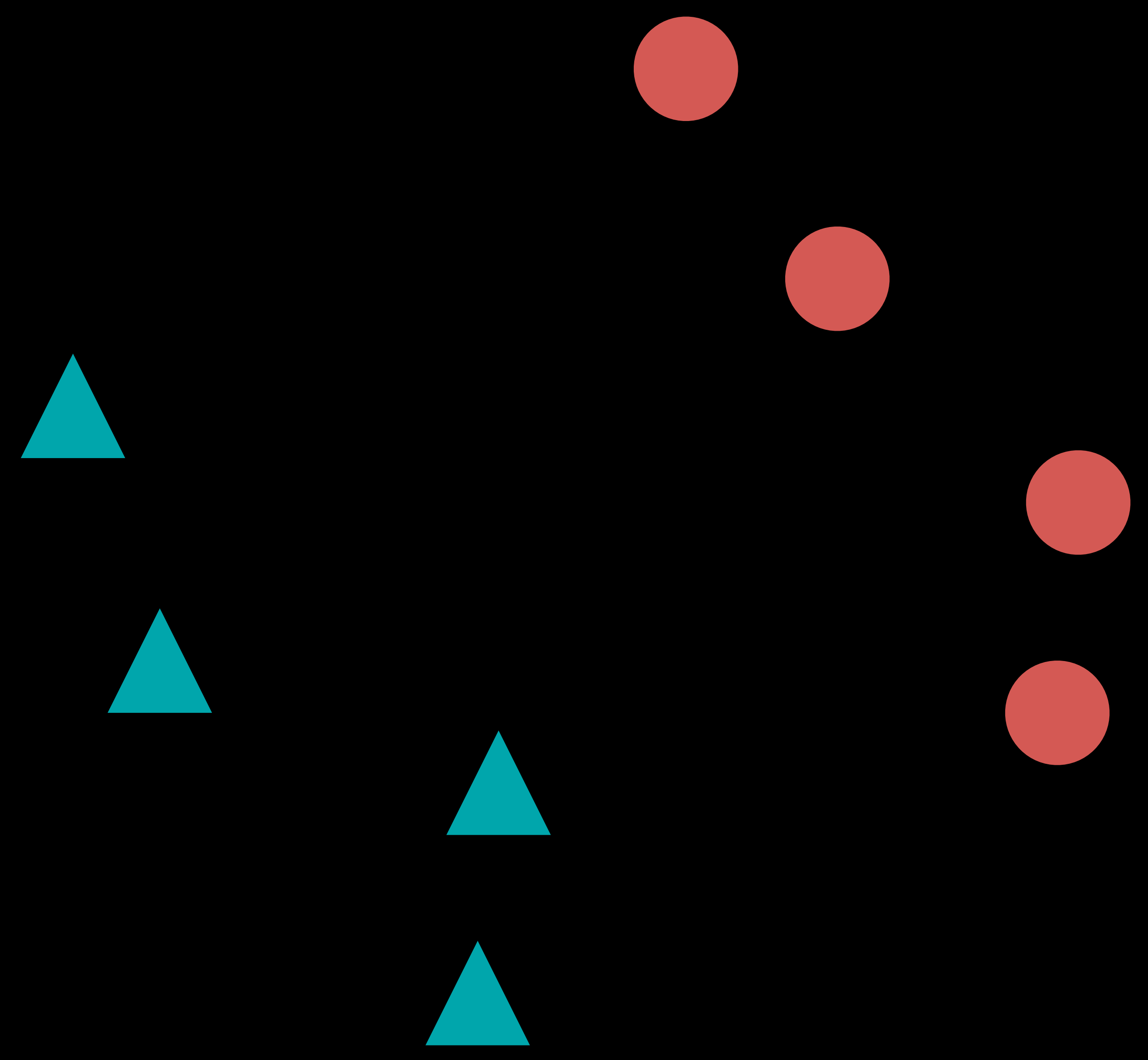
How

```
def is_triangle(x):
  u = np.sum(x[:len(x)//2])
  l = np.sum(x[len(x)//2]:)
  if u < l/2:
      return "triangle"
  else:
      return "circle"
```

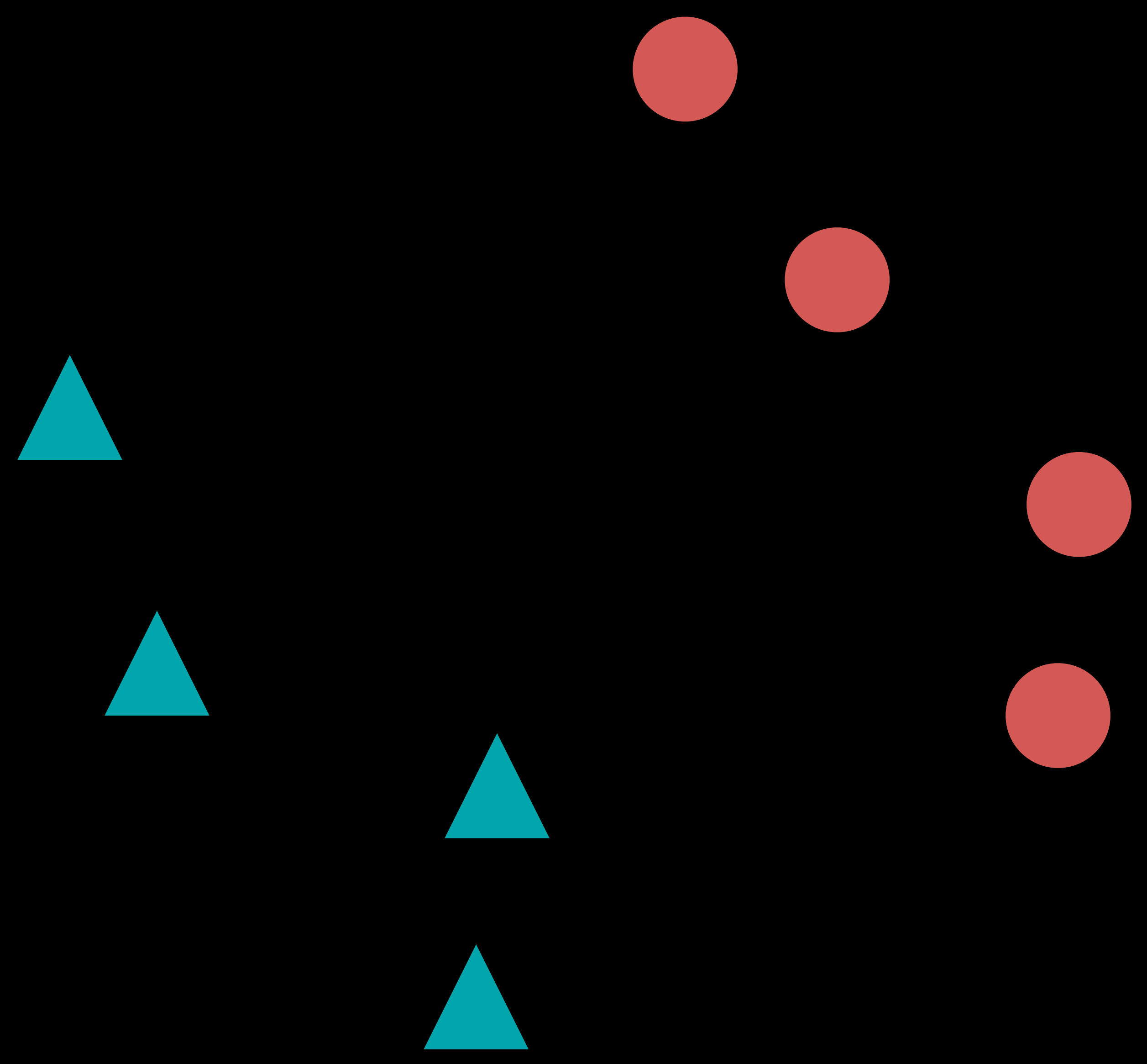
How

```
def is_triangle(x):
  u = np.sum(x[:len(x)//2])
  l = np.sum(x[len(x)//2]:)
  if u < l/2:
      return "triangle"
  else:
      return "circle"
```

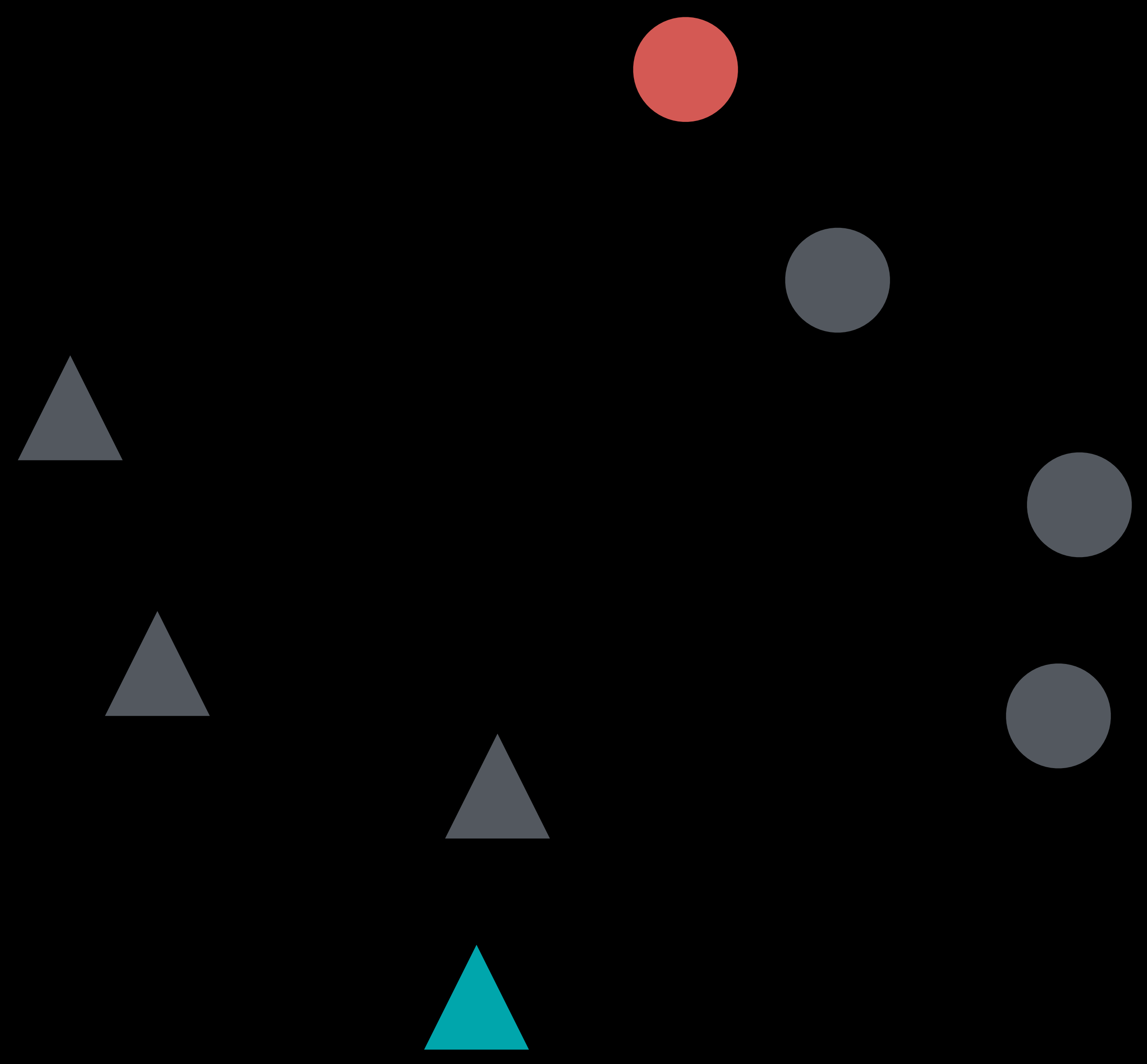
What



What



(not-even)What



Questions

https://nicholas.carlini.com nicholas@carlini.com