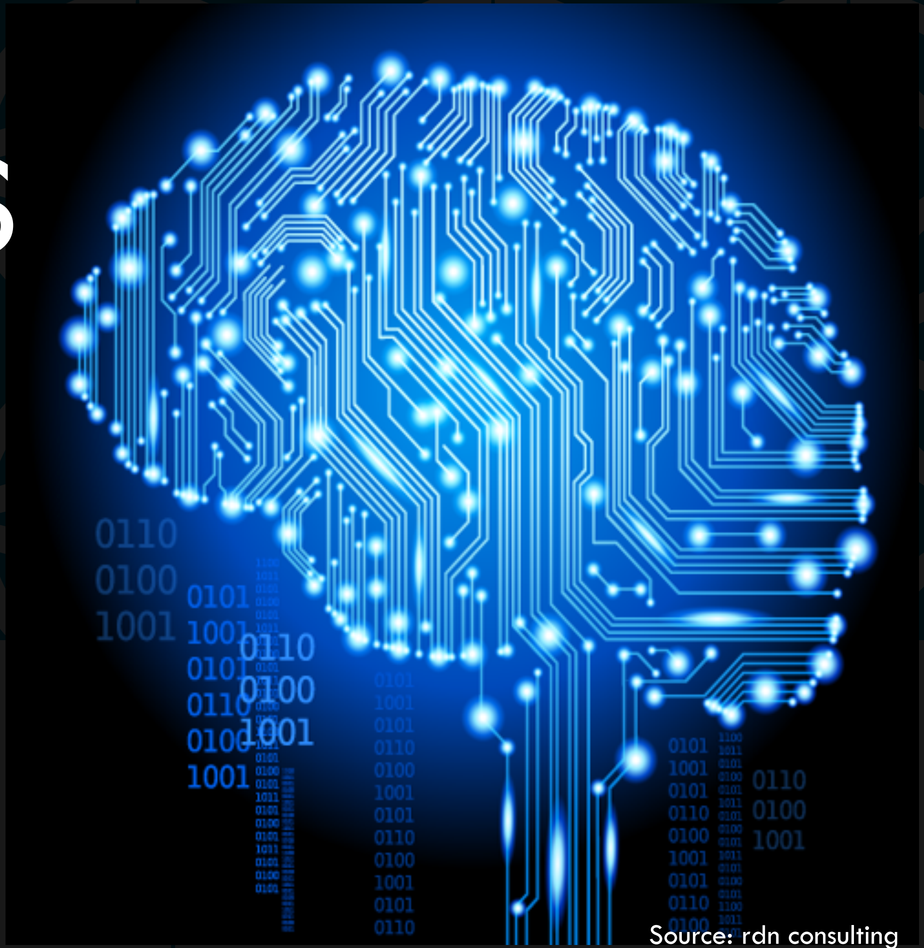
DEEP LEARNING FOR DETECTING ANOMALIES AND SOFTWARE VULNERABILITIES

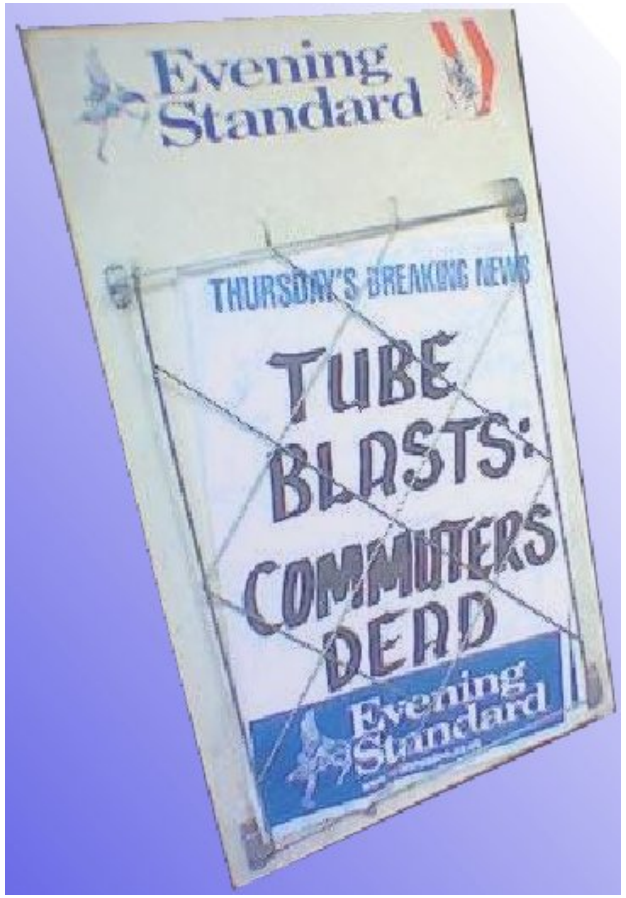


Trần Thế Truyền
Deakin University

truyen.tran@deakin.edu.au
prada-research.net/~truyen
tranthetruyen
letdataspeak.blogspot.com
@truyenoz
goo.gl/3jJ100

Hanoi, Jan 17th 2017

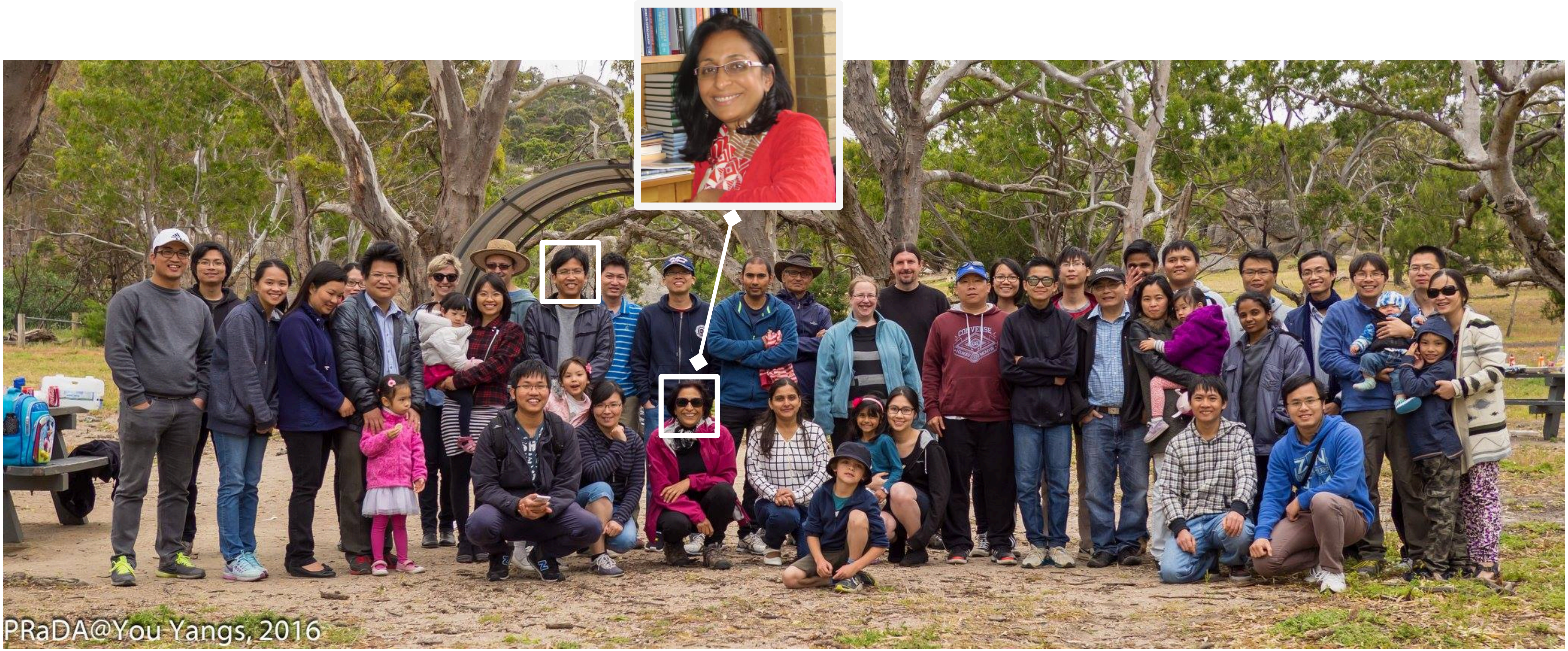
REAL-WORLD FAILURE OF ANOMALY DETECTION



London, July 7, 2005

Real world - what are the operators monitoring?

PRADA @ DEAKIN, MELBOURNE



OUR APPROACH TO SECURITY: (DEEP) MACHINE LEARNING

Usual detection of attacks are based on profiling and human skills

But attacking tactics change overtime, creating zero-day attacks

Systems are very complex now, and no humans can cover all

→ It is best to use machine to learn continuously and automatically.

→ Humans can provide feedbacks for the machine to correct itself.

→ Deep learning is on the rise.

For now: It is best for human and machine to co-operate.

SOLVING REAL WORLD PROBLEMS IS REWARDING

iCetana

iHosp™

SAMSUNG

Startups

Contracts

AGENDA

Part I: Introduction to deep learning

- A brief history
- Top 3 architectures
- Unsupervised learning

Part II: Anomaly detection

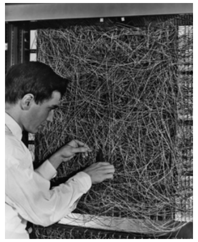
Part III: Software vulnerabilities

Yann LeCun

1988

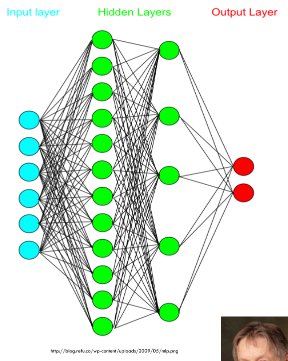
Geoff Hinton

2006



Rosenblatt's perceptron

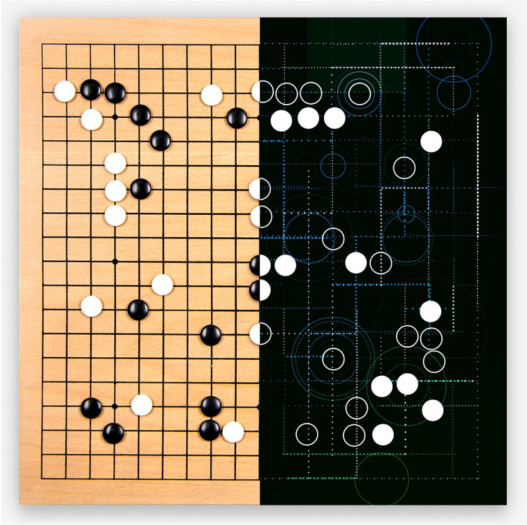
1958



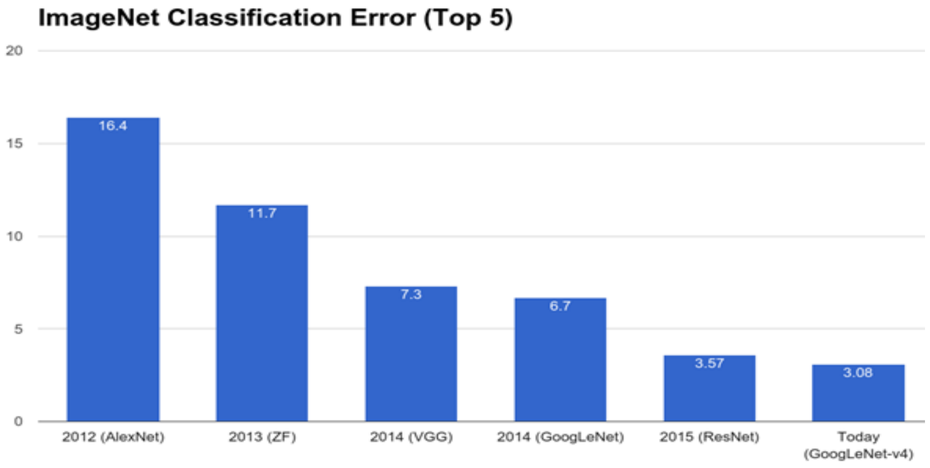
1986

2012

AlphaGo

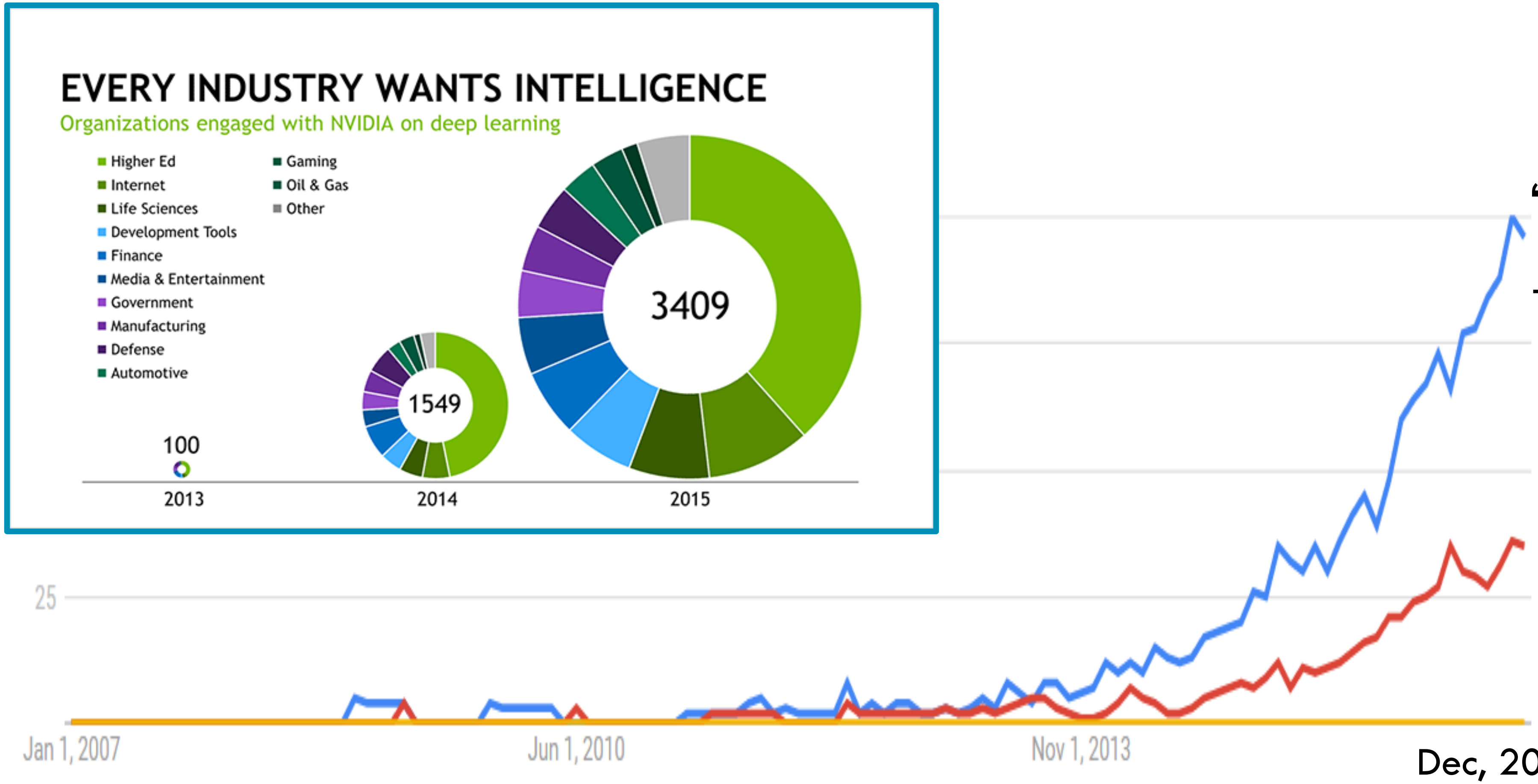


2016-2017



http://redcatlabs.com/2016-07-30_FifthElephant-DeepLearning-Workshop/#/

DEEP LEARNING IS SUPER HOT



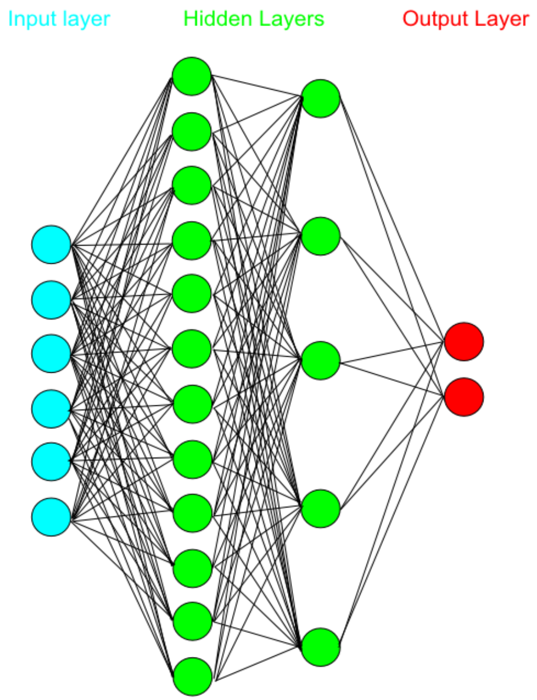
"deep learning" + **data**

"deep learning" + **intelligence**

Dec, 2016

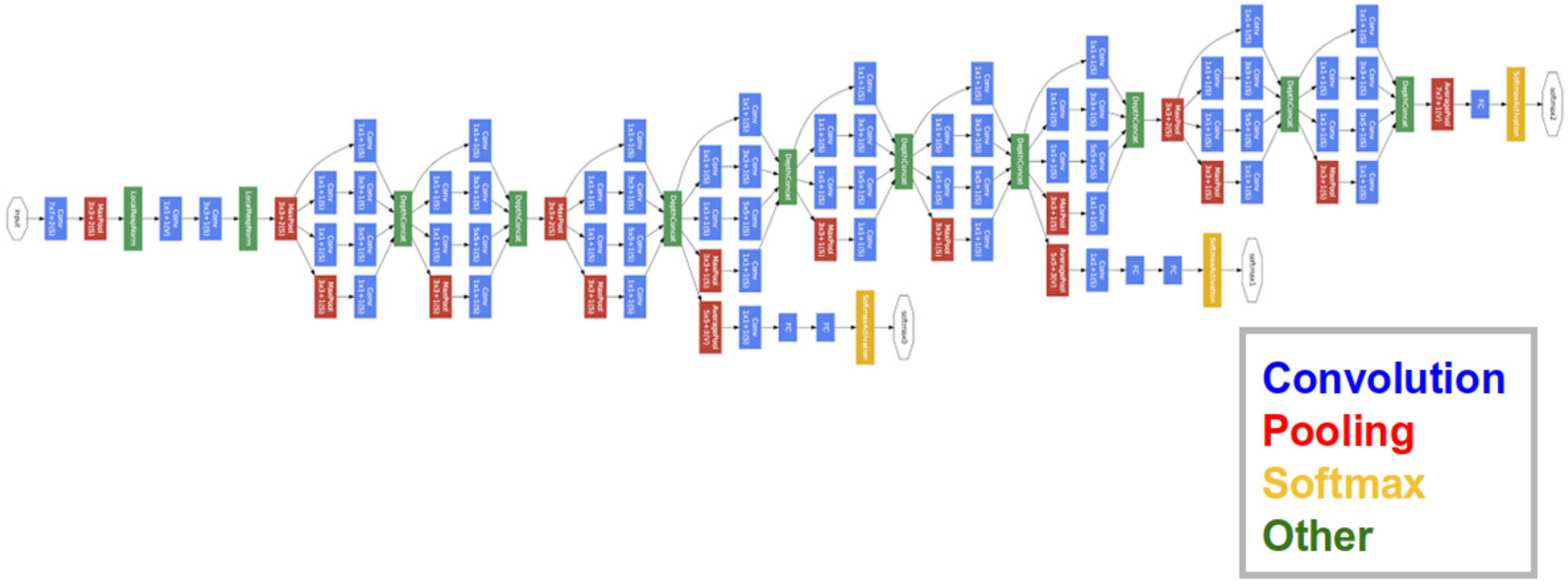
DEEP LEARNING IS NEURAL NETS, BUT …

1986



http://blog.refu.co/wp-content/uploads/2009/05/mlp.png

2016



TWO LEARNING PARADIGMS

Supervised learning

(mostly machine)

A → B

Will be quickly solved for "easy" problems (Andrew Ng)

Unsupervised learning

(mostly human)

$$\mathbf{v} \sim P_{model}(\mathbf{v})$$

$$P_{model}(\mathbf{v}) \approx P_{data}(\mathbf{v})$$

KEY IN MACHINE LEARNING: FEATURE ENGINEERING

In typical machine learning projects, 80-90% effort is on feature engineering

- A right feature representation doesn't need much work. Simple linear methods often work well.

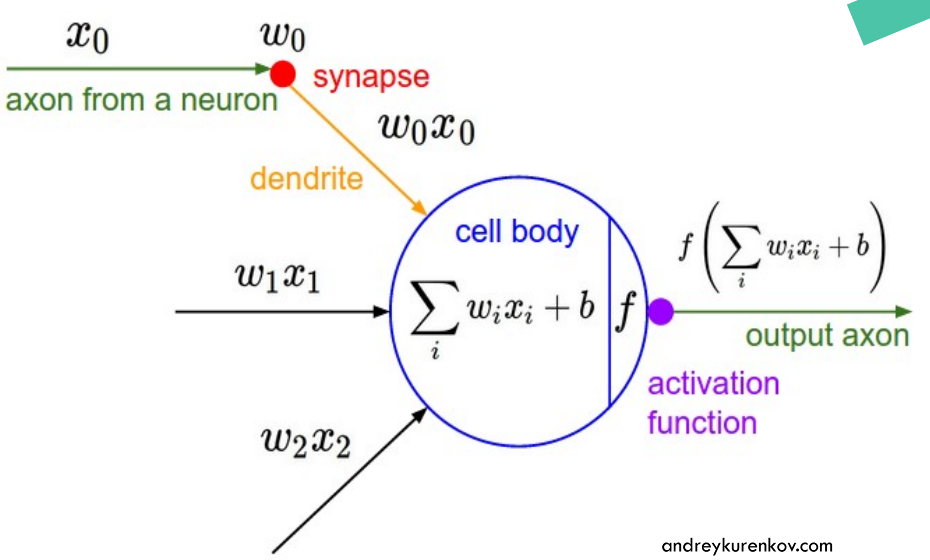
Text: BOW, n-gram, POS, topics, stemming, tf-idf, etc.

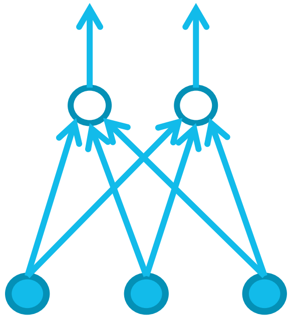
Software: token, LOC, API calls, #loops, developer reputation, team complexity, report readability, discussion length, etc.

Try yourself on Kaggle.com!

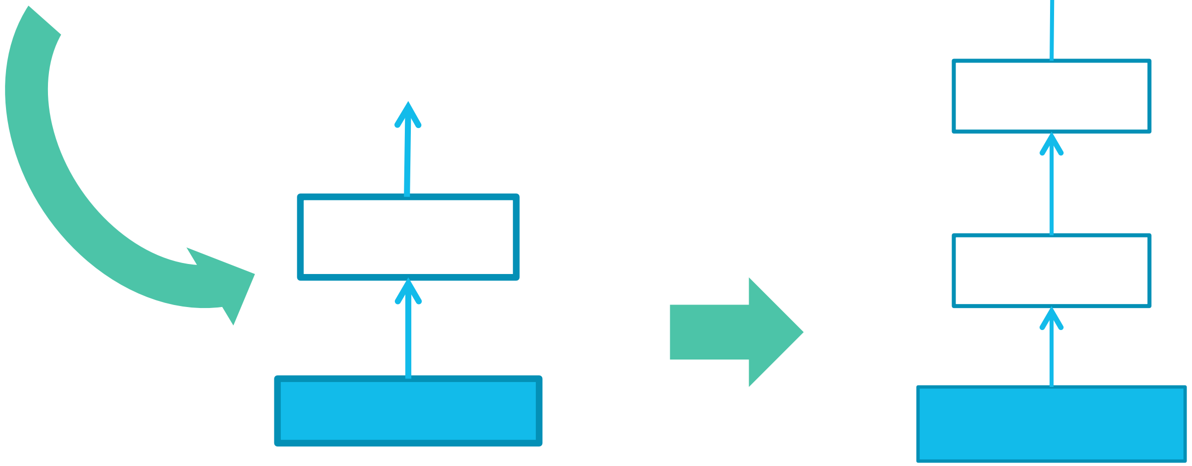
FEEDFORWARD NETS: FEATURE DETECTION

Integrate-and-fire neuron



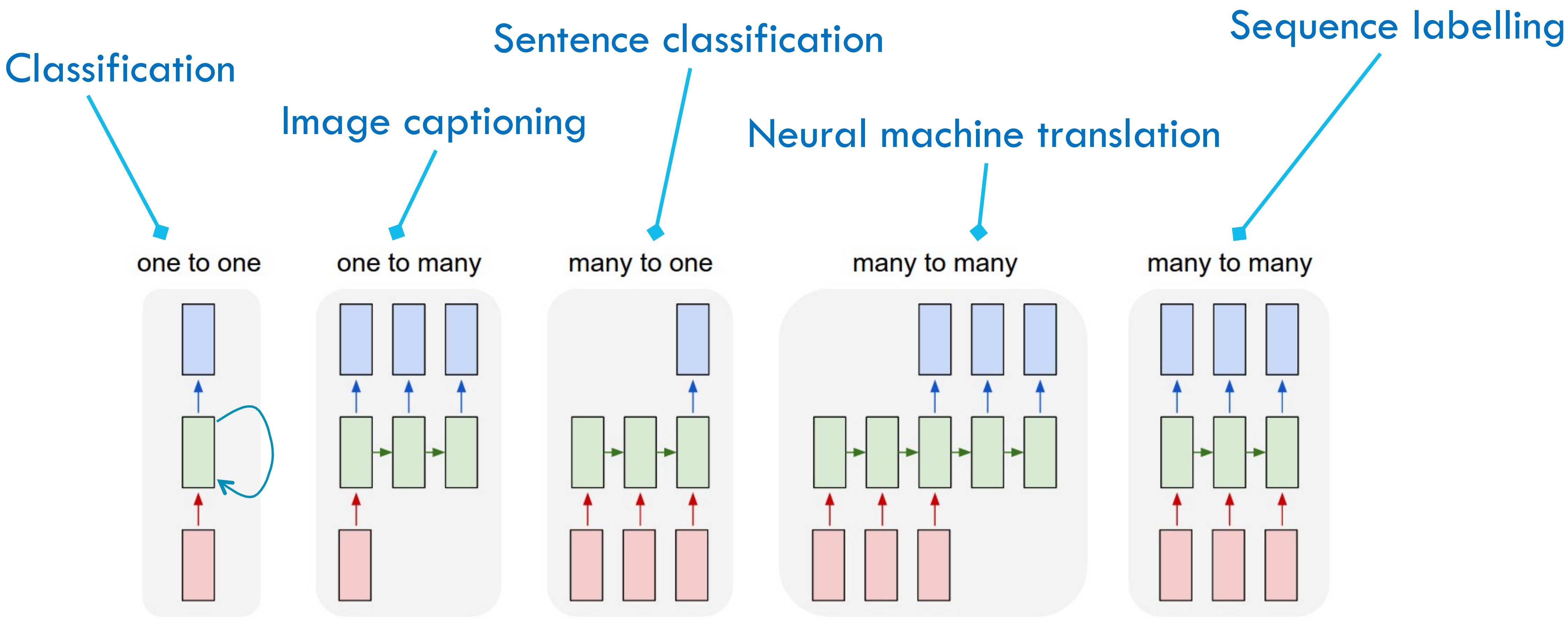


Feature detector



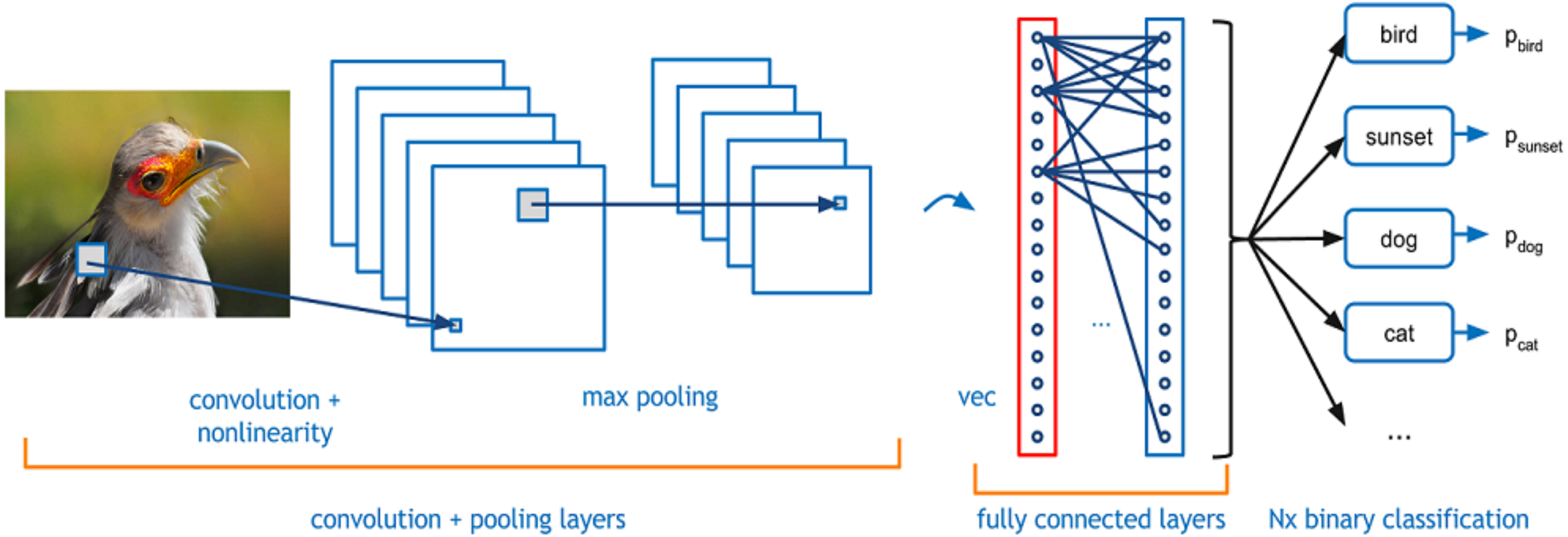
Block representation

RECURRENT NEURAL NETWORKS: TEMPORAL DYNAMICS

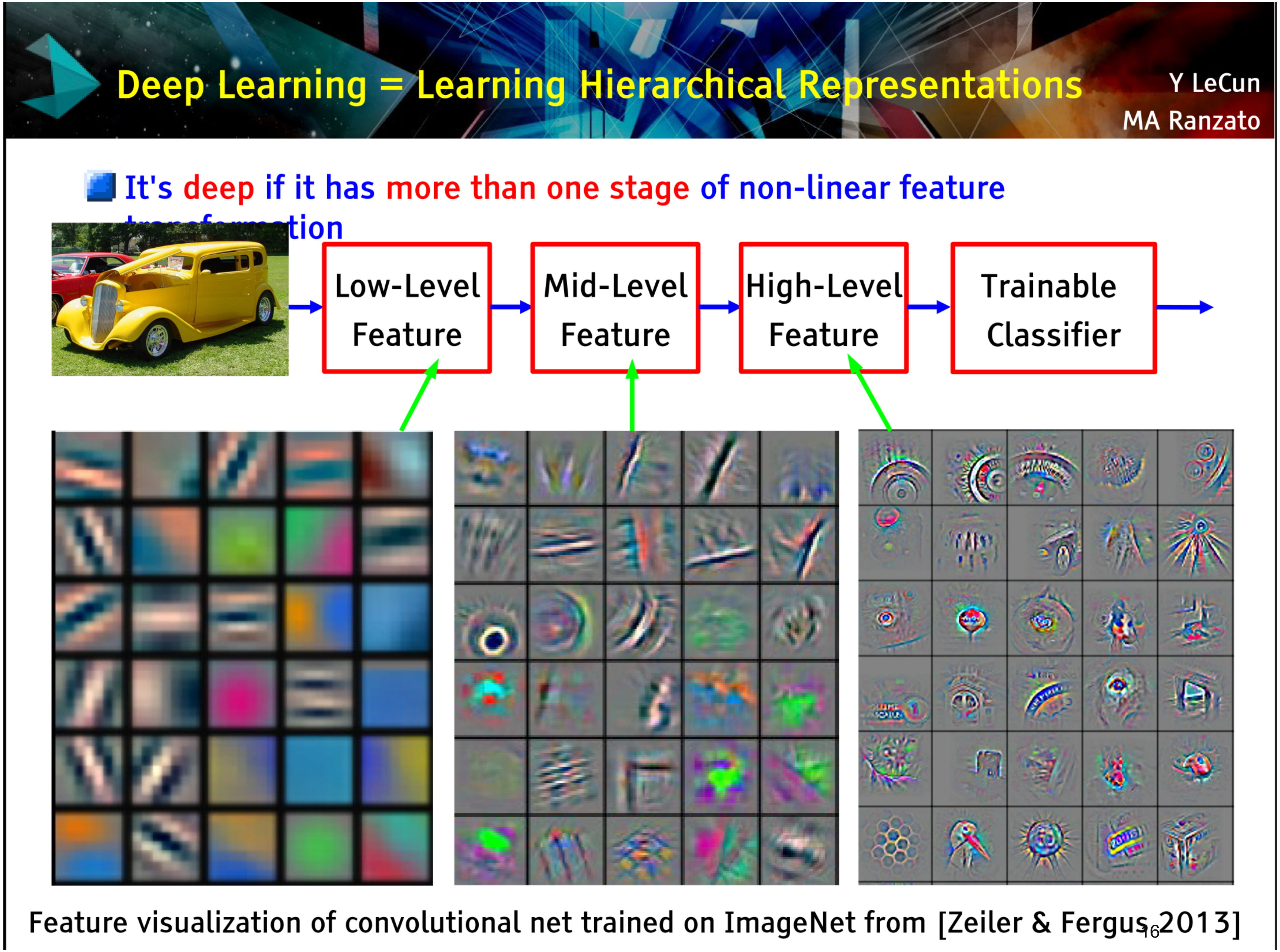


Source: http://karpathy.github.io/assets/rnn/diags.jpeg

CONVOLUTIONAL NETS: MOTIF DETECTION



adeshpande3.github.io



Slide from Yann LeCun

UNSUPERVISED LEARNING

WE WILL BRIEFLY COVER

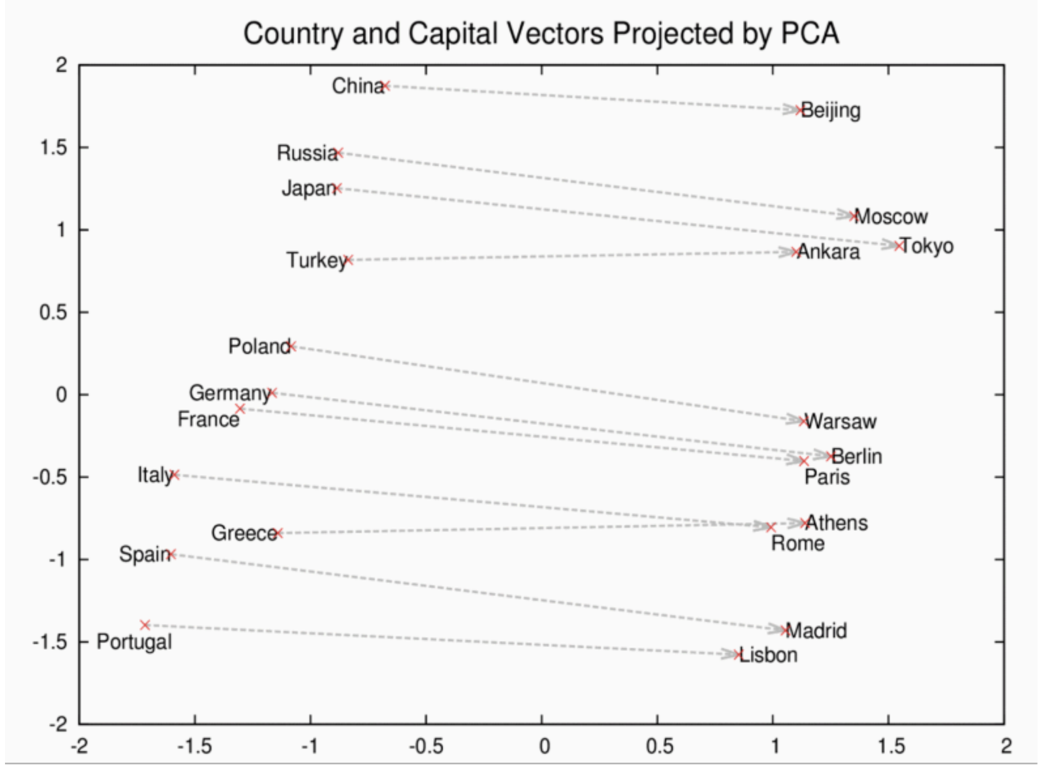
Word embedding

Deep autoencoder

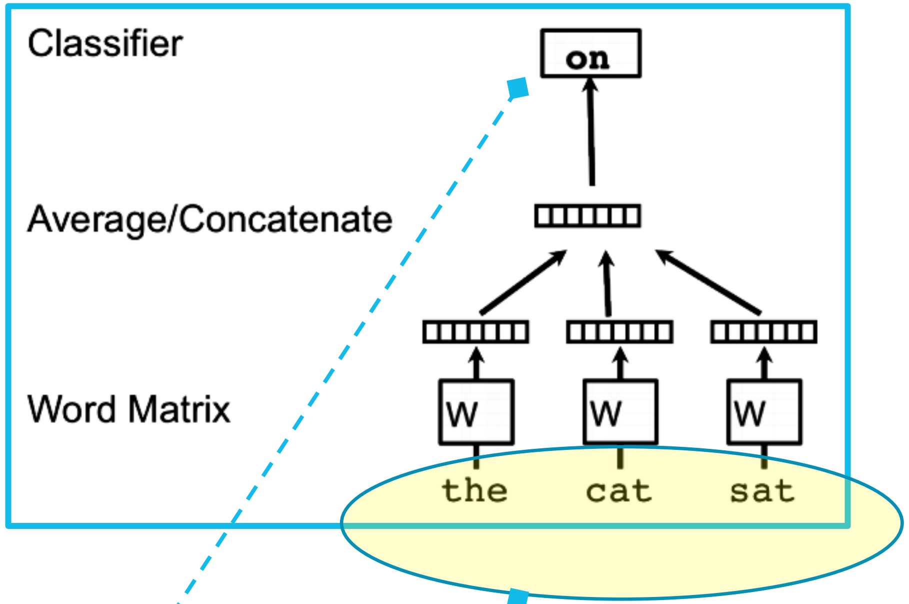
RBM → DBN → DBM

Generative Adversarial Net (GAN)

WORD EMBEDDING



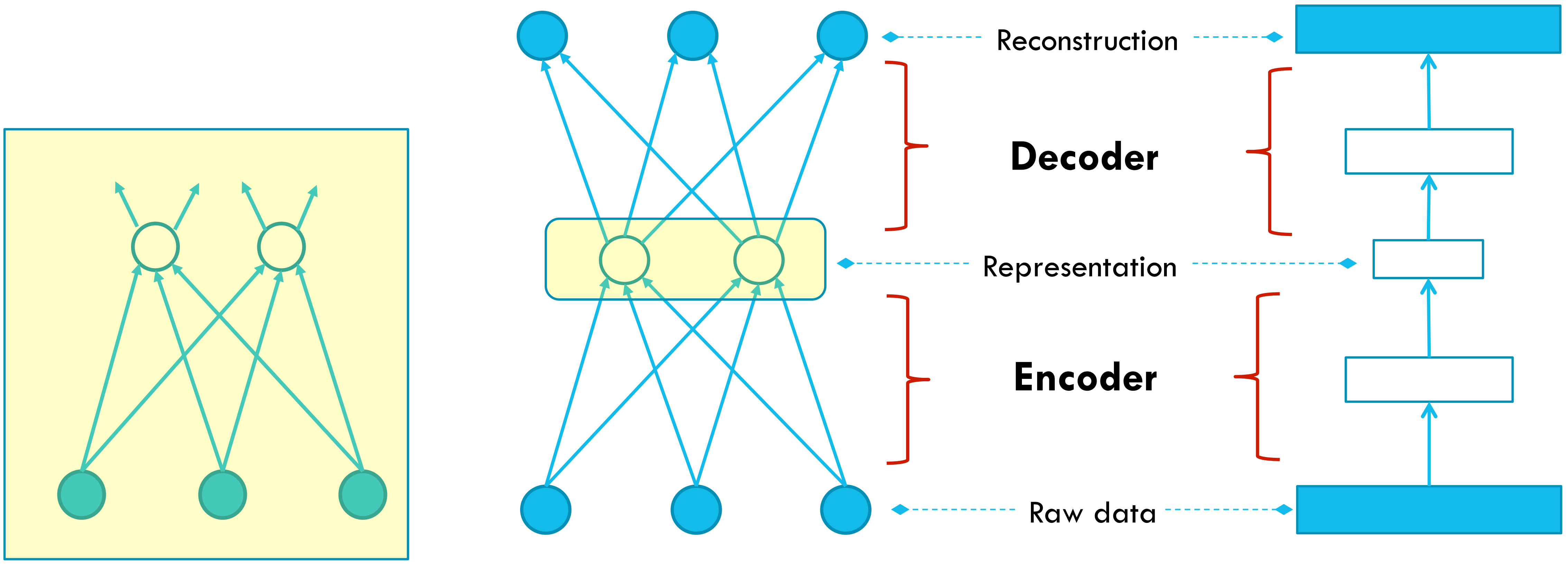
(Mikolov et al, 2013)



$$P\left(w_t \mid C_t\right) = \frac{e^{V_{w_t}^{\top} f(C_t)}}{\sum_{w \in Vocab} e^{V_w^{\top} f(C_t)}}$$

$$f(C_t) = \frac{1}{|C_t|} \sum_{w \in C_t} W_w$$

DEEP AUTOENCODER – SELF RECONSTRUCTION OF DATA



Feature detector

Auto-encoder

Deep Auto-encoder

GENERATIVE MODELS

$$\mathbf{v} \sim P_{model}(\mathbf{v})$$

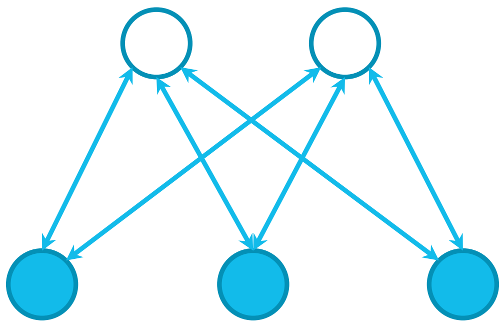
$$P_{model}(\mathbf{v}) \approx P_{data}(\mathbf{v})$$

Many applications:

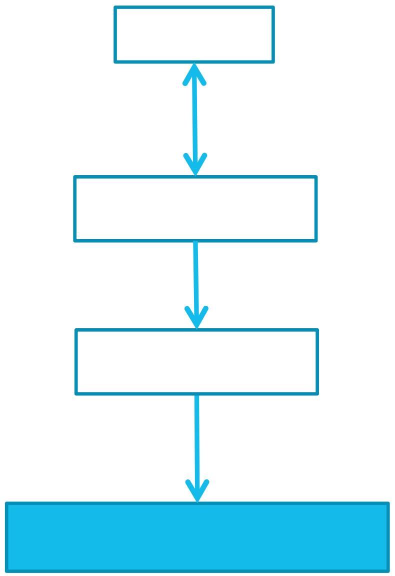
- Text to speech
- Simulate data that are hard to obtain/share in real life (e.g., healthcare)
- Generate meaningful sentences conditioned on some input (foreign language, image, video)
- Semi-supervised learning
- Planning

A FAMILY: RBM → DBN → DBM

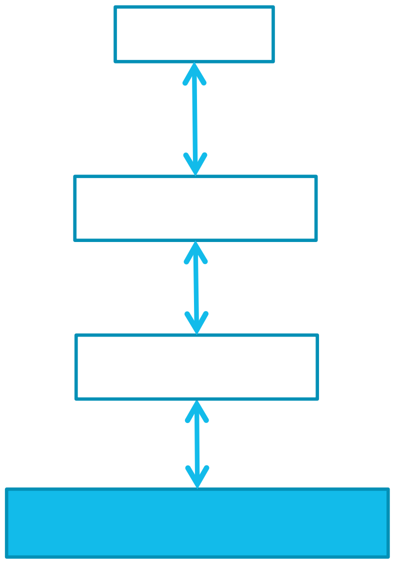
$$p\left(\mathbf{v},\mathbf{h};\psi\right) \propto \exp\left[-\underbrace{E\left(\mathbf{v},\mathbf{h};\psi\right)}_{\textit{energy}}\right]$$



Restricted Boltzmann Machine
(~1994, 2001)



Deep Belief Net
(2006)



Deep Boltzmann Machine
(2009)

GAN: GENERATIVE ADVERSARIAL NETS
(GOODFELLOW ET AL, 2014)

Yann LeCun: *GAN is one of best idea in past 10 years!*

*Instead of modeling the entire distribution of data, learns to map ANY random distribution into the region of data, so that **there is no discriminator that can distinguish sampled data from real data.***

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

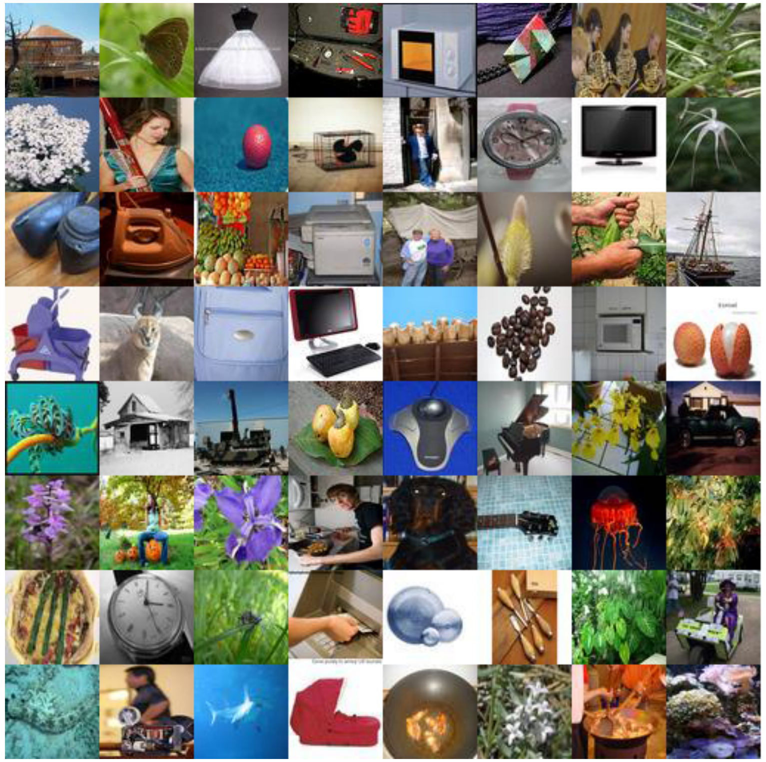
Binary discriminator, usually a neural classifier

Any random distribution in any space

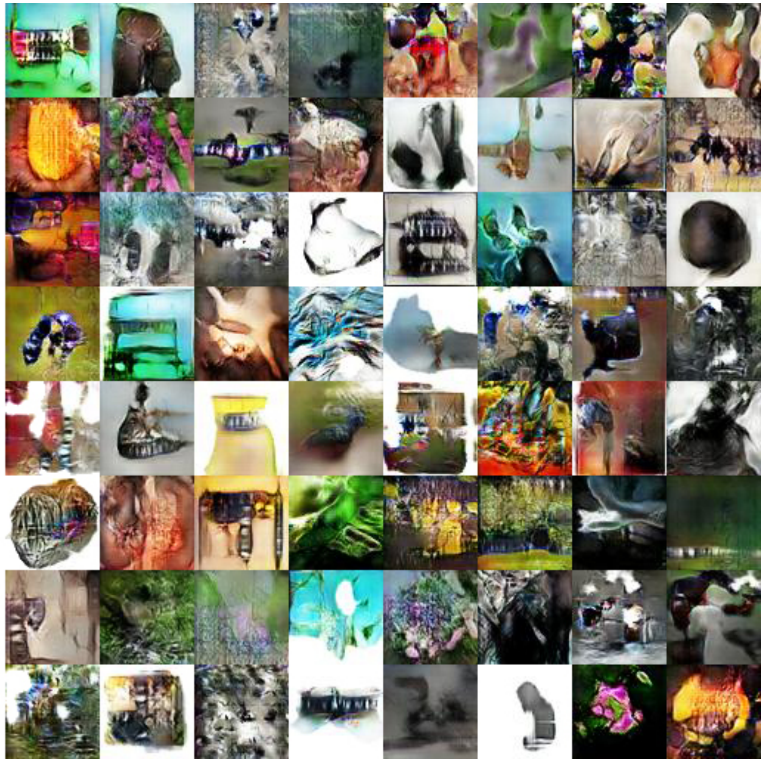
Neural net that maps z → x

GAN: GENERATED SAMPLES

The best quality pictures generated thus far!



Real



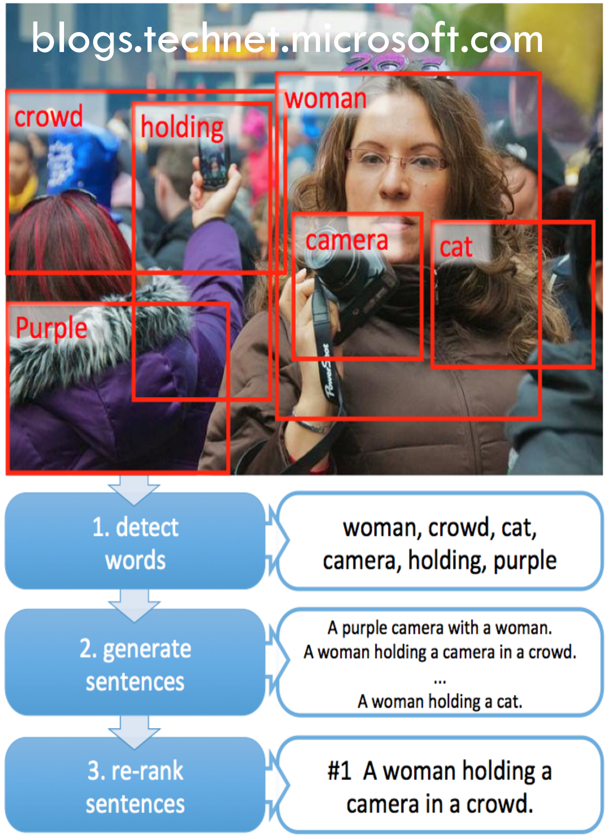
Generated

http://kvfrans.com/generative-adversial-networks-explained/

DEEP LEARNING IN COGNITIVE DOMAINS

Google Translate

Where human can recognise, act or answer accurately within seconds

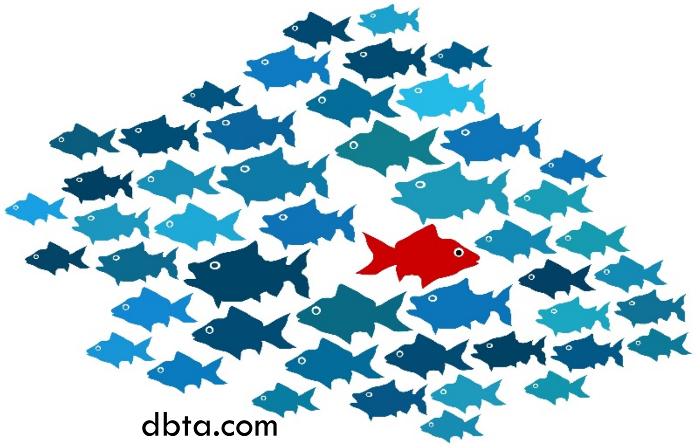


DEEP LEARNING IN NON-COGNITIVE DOMAINS

- **Where humans need extensive training to do well**
- Domains that demand transparency & interpretability.

… healthcare

… security



… genetics, foods, water …

END OF PART I

dbta.com

ANOMALY DETECTION USING UNSUPERVISED LEARNING

This work is partially supported by the Telstra-Deakin Centre of Excellence in Big Data and Machine Learning

AGENDA

Part I: Introduction to deep learning

Part II: Anomaly detection

- Multichannel
- Unusual mixed-data co-occurrence
- Object lifetime model

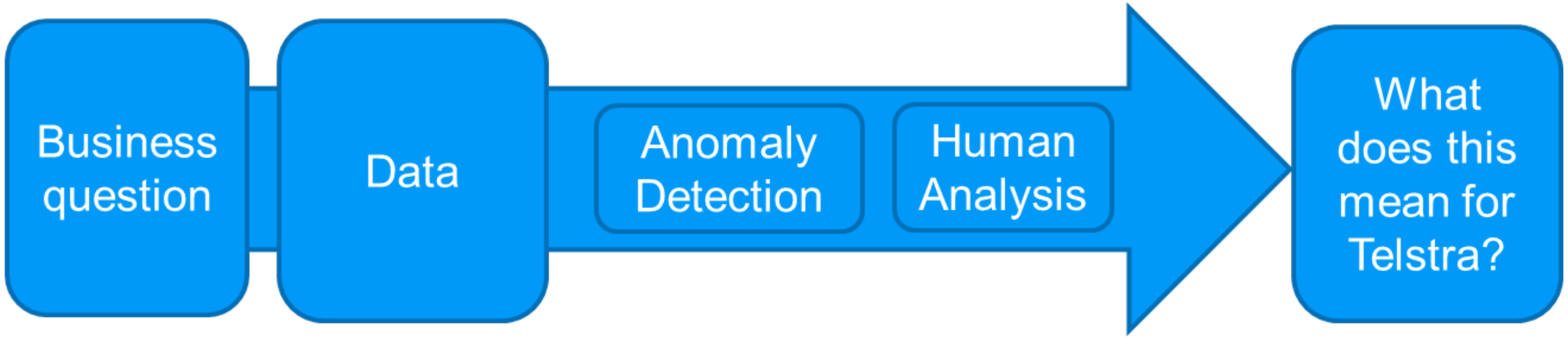
Part III: Software vulnerabilities

Strategy: learn normality, anything does not fit in is abnormal

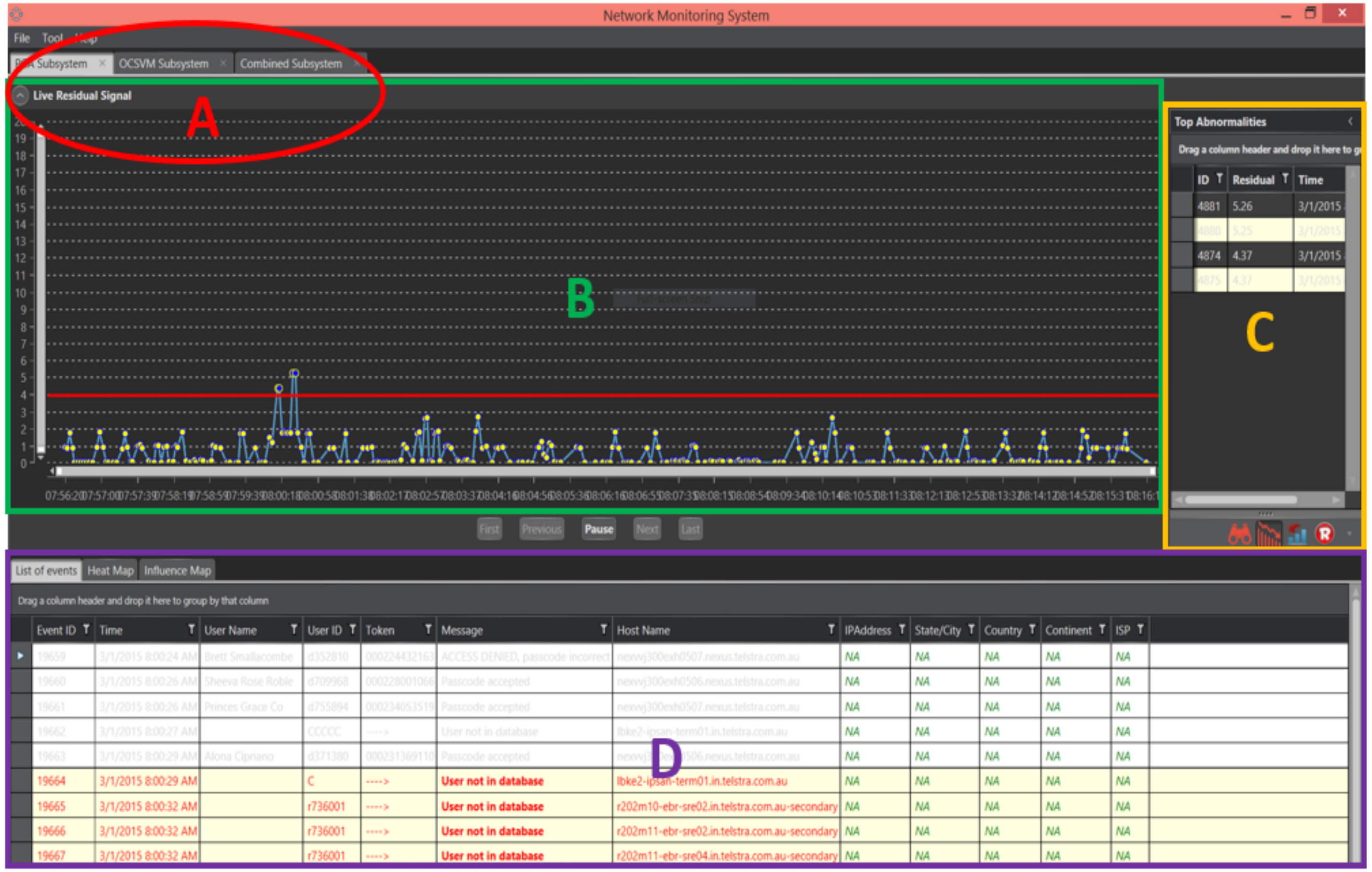
PROJECT: DISCOVERY IN TELSTRA SECURITY OPERATIONS

We use smart people and smart tools to discover unknown malicious or risky behaviour to inform and protect Telstra and its customers.

Discovery workflow:



SOFTWARE: SNAPSHOT



A) Anomaly detection systems

B) The main screen showing the residual signal, and the threshold for anomaly detection

C) Top anomalies

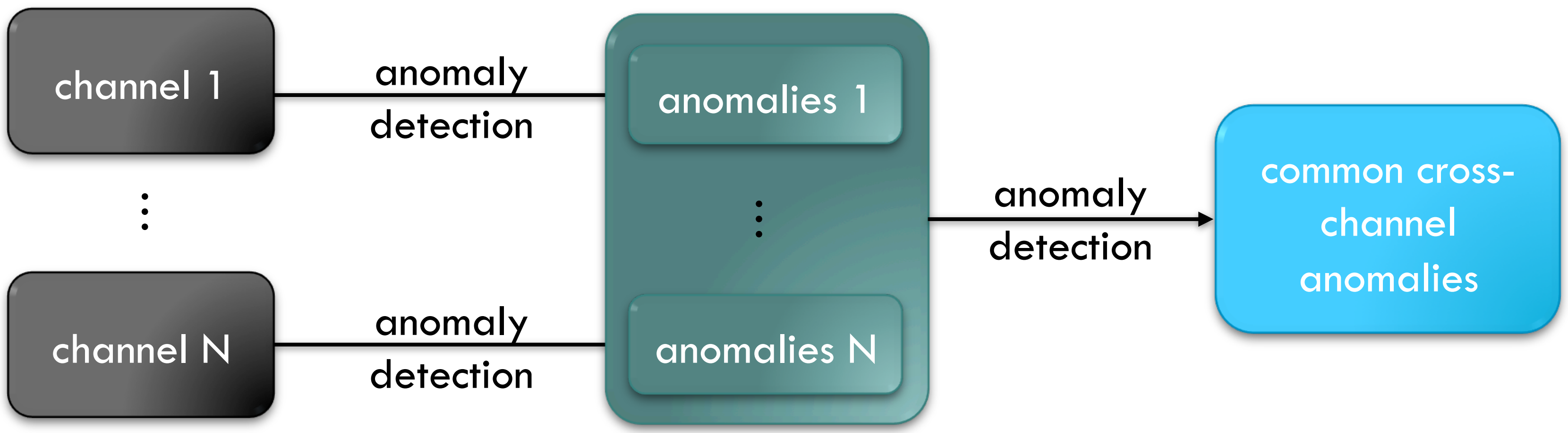
D) Event details for selected anomaly

MULTICHANNEL FRAMEWORK

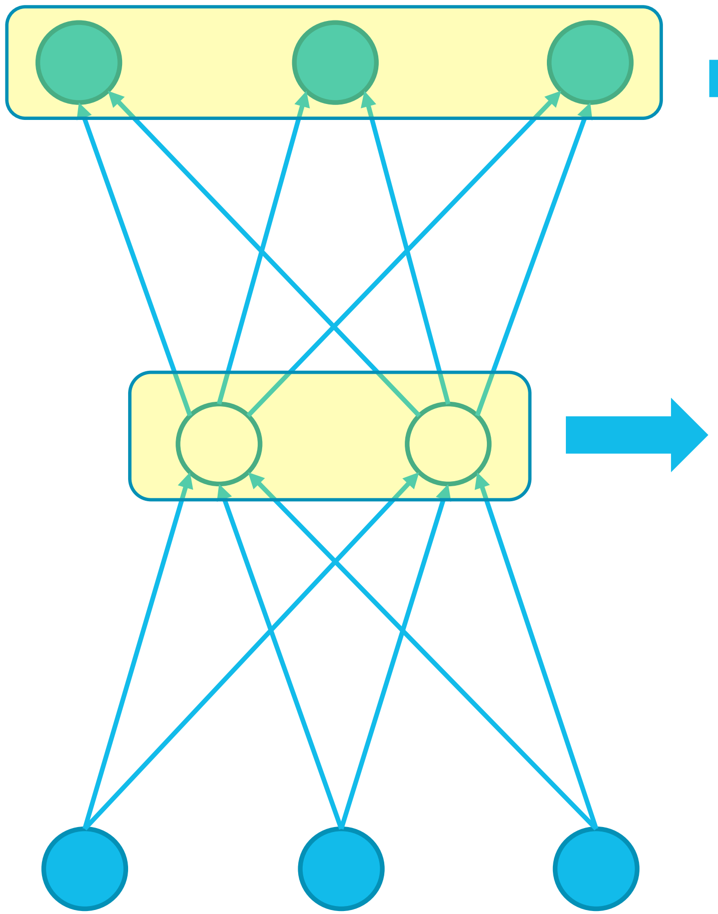
Detect common anomalous events that happen across multiple information channels

1. Cross-channel Autoencoder (CC-AE)

General framework



DETECTION METHOD: AUTOENCODER



Reconstruction error

External outlier detector?

METHOD: CROSS-CHANNEL AUTOENCODER

1. Single channel anomaly detection
   - For each channel, model the data with an autoencoder
   - Determine the anomalies by analysing the reconstruction errors
2. Augmenting the reconstruction errors
   - Augment the reconstruction errors across channels
   - Model the reconstruction errors with an autoencoder
3. Cross-channel anomaly detection
   - Determine the cross-channel anomalies by analysing the reconstruction errors

RESULTS 1 – NEWS DATA

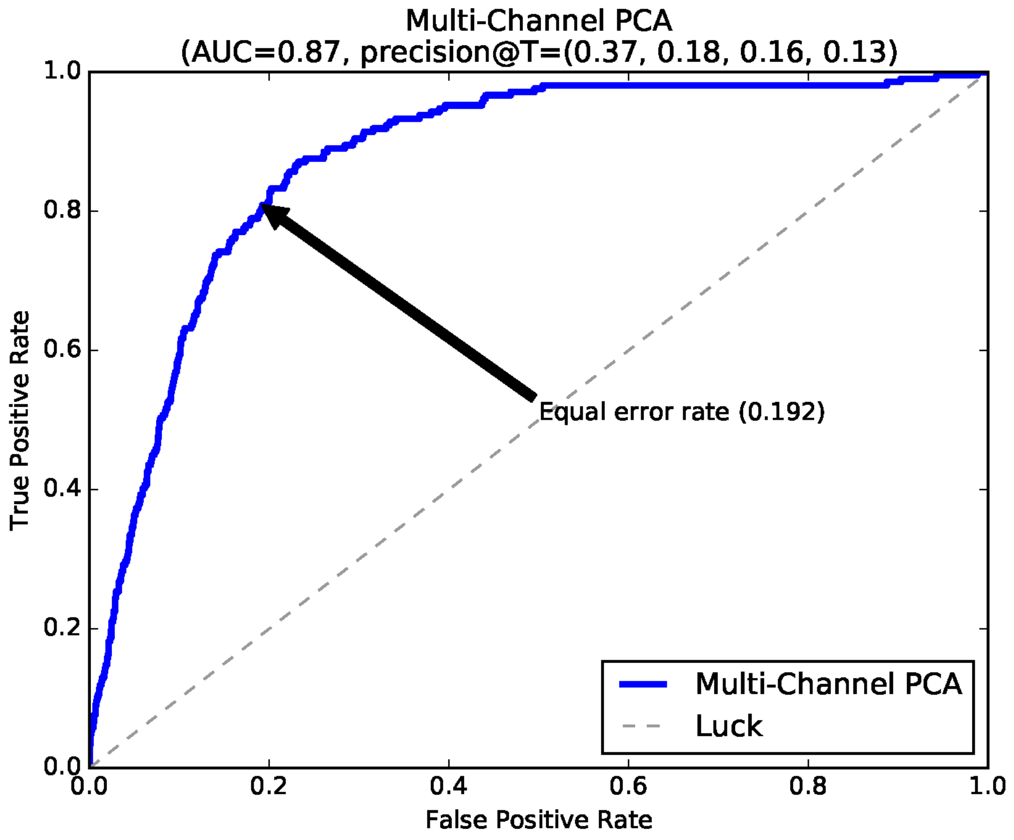
A channel is defined to be the stream of articles about a specific topic published by a news agency, e.g. economy-related articles from BBC

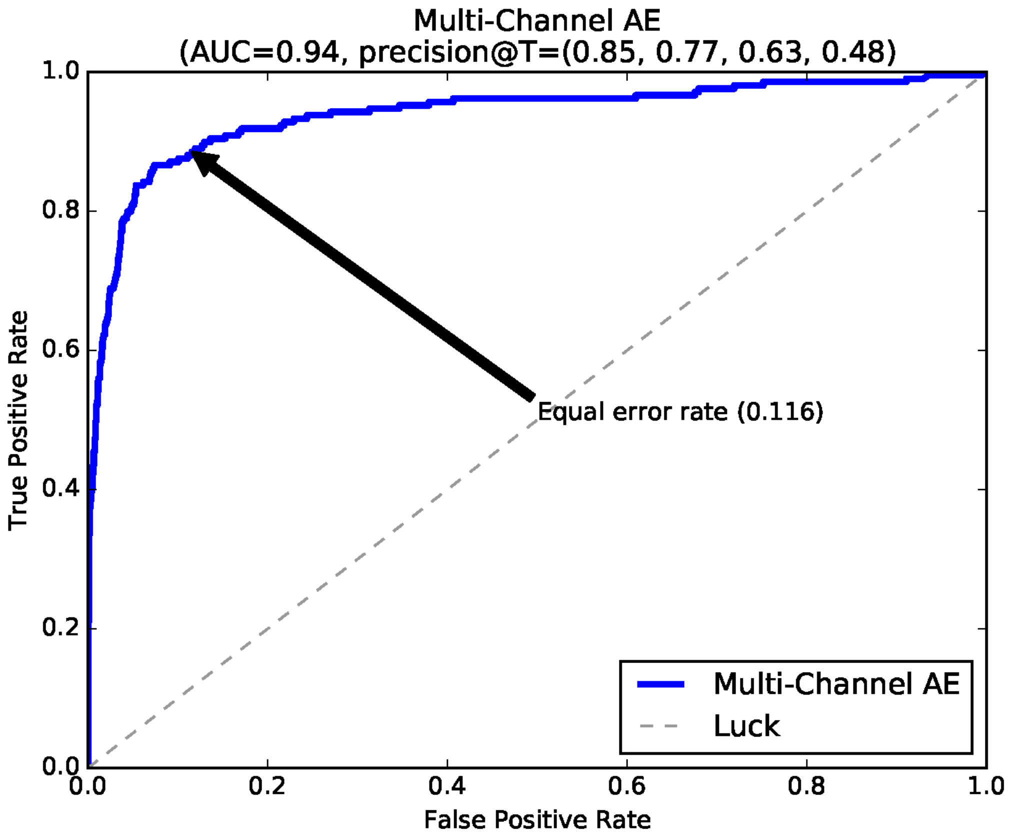
- 3 news agencies: BBC, Reuters, and CNN
- 9 predefined topics: politics, sports, health, entertainment, world-news, technology, and Asian news
- Free-form text data

Feature extraction: Bag of words representation

Anomaly injection: Breastfeeding articles

RESULTS 1 – NEWS DATA





RESULTS 2 – DEAKIN SQUID DATA

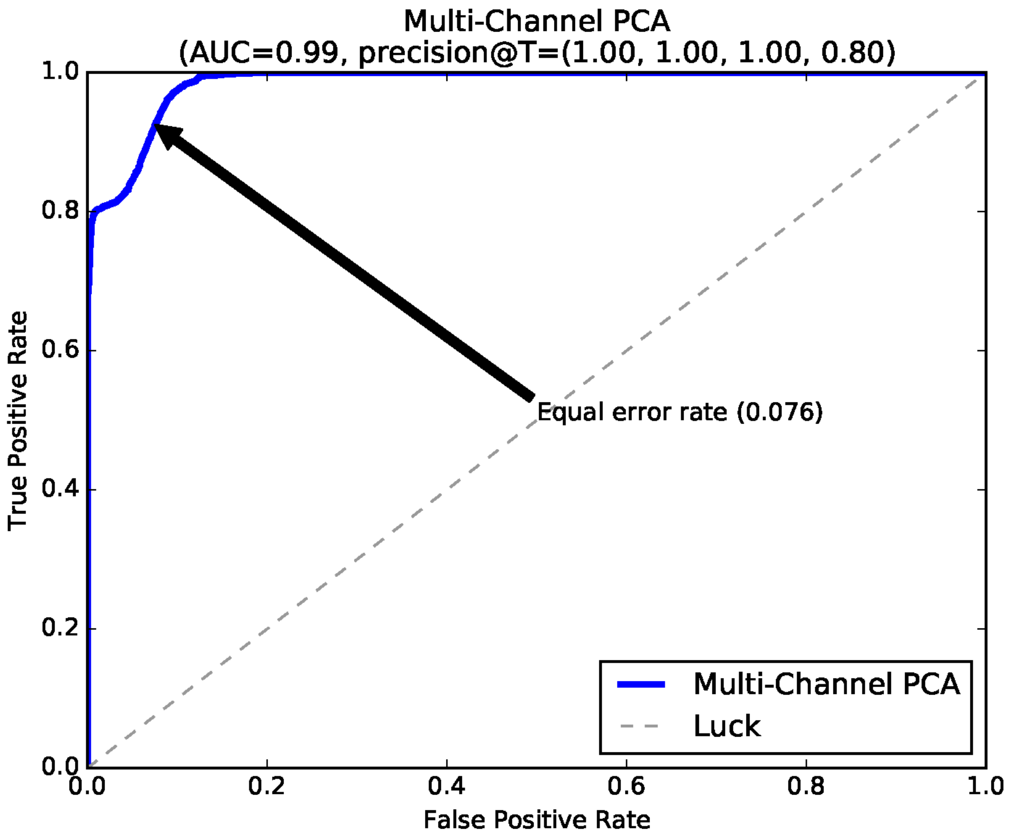
Squid is a caching and forwarding web proxy.

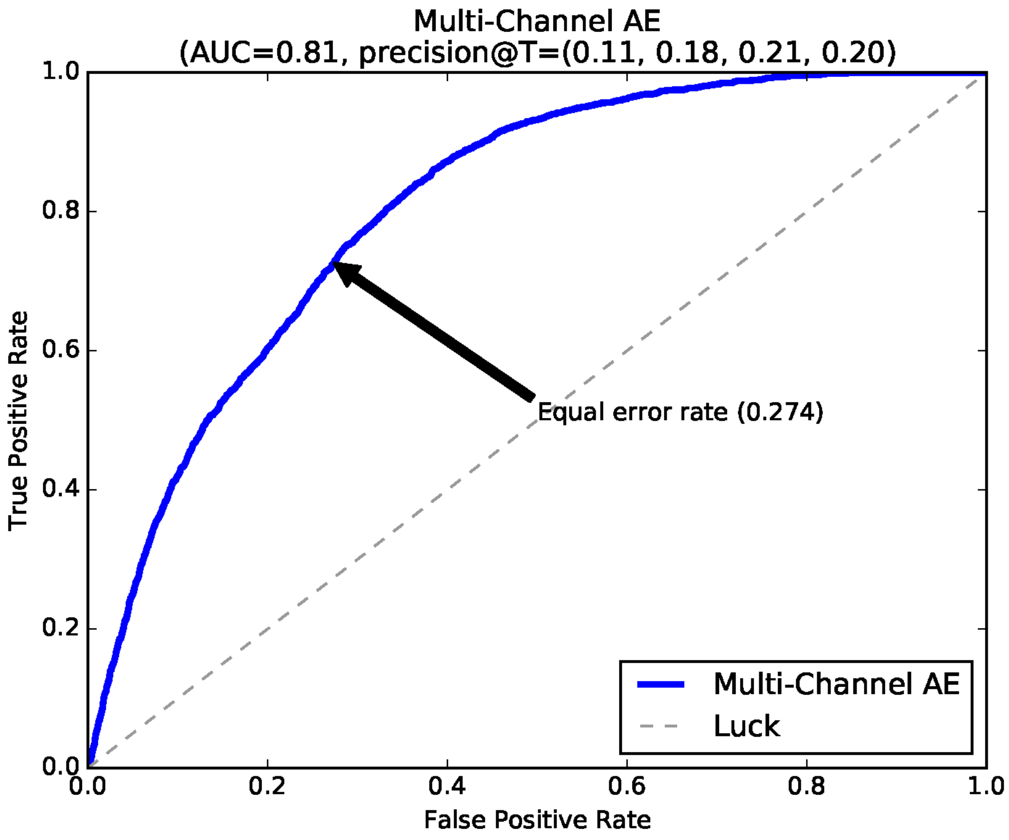
- Each server is defined to be a channel
- 7 channels with inbound and outbound network data
- Sample datapoint:

```
      1          2          3                4          5      6              7
1469032590.233  19  10.132.169.158  TCP_HIT/200  4771  GET http://Europe.cnn.com/
EUROPE/potd/2001/04/17/tz.pulltizer.ap.jp  -  NONE/-  image/jpeg
               7 cont'd                    8     9        10
```

- Bag of words representation
- Anomaly Injection: URLs from URLBlackList.com

RESULTS 2 – DEAKIN SQUID DATA





AGENDA

Part I: Introduction to deep learning

Part II: Anomaly detection

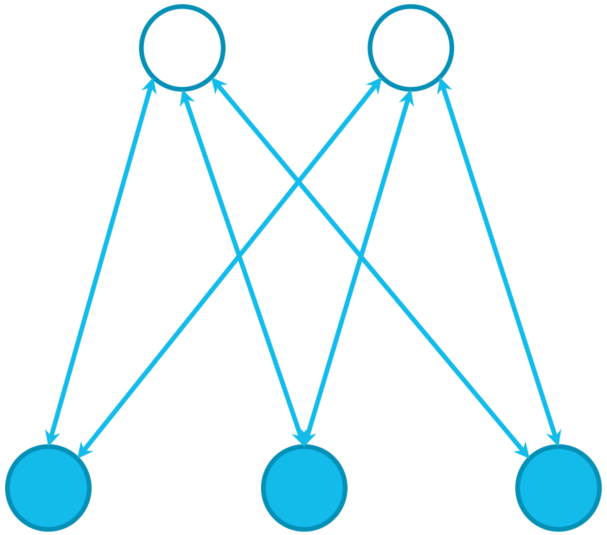
- Multichannel
- Unusual mixed-data co-occurrence
- Object lifetime model

Part III: Software vulnerabilities

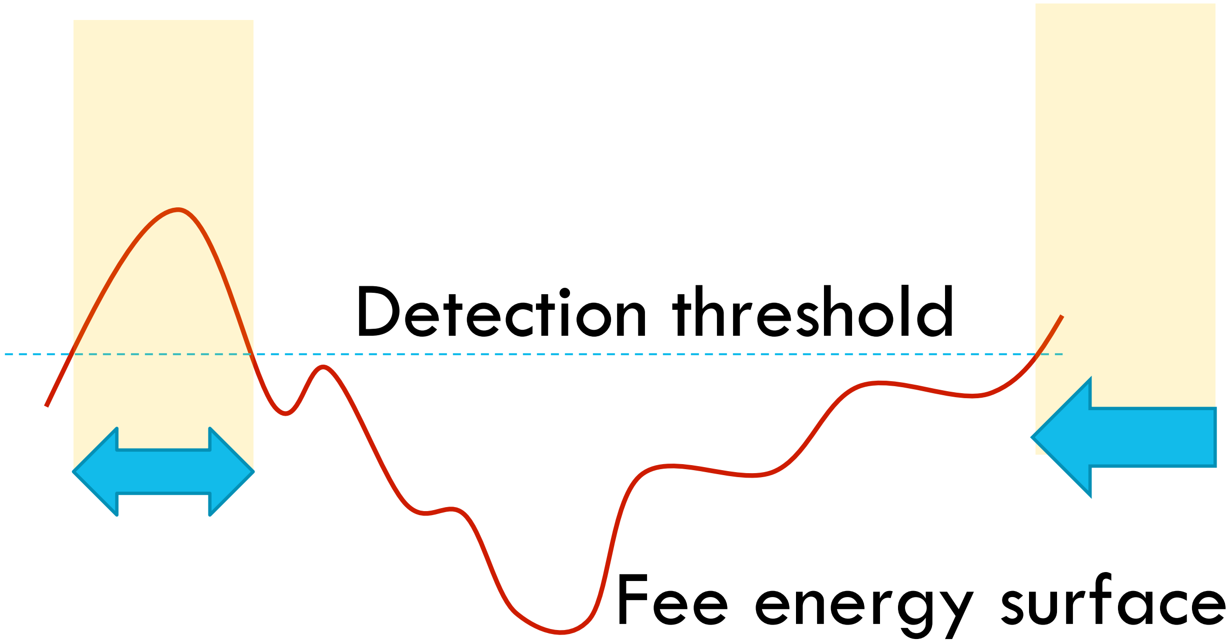
MIXED DATA

	A	B	C	D	E	F	G	H	I	J
1	Age	Sex	Chest pain type	Resting blood pressure	Serum cholestoral (mg/dl)	Fasting blood sugar > 120 mg/dl ?	Resting electrocardiographic result	Maximum heart rate achieved	Exercise induced angina	oldpeak = ST depression induced by exercise relative to rest
2	70	male	asymptomatic (4)	130.0	322.0	no	2	109.0	no	2.4
3	67	female	non-anginal pain (3)	115.0	564.0	no	2	160.0	no	1.6
4	57	male	atypical angina (2)	124.0	261.0	no	0	141.0	no	0.3
5	64	male	asymptomatic (4)	128.0	263.0	no	0	105.0	yes	0.2
6	74	female	atypical angina (2)	120.0	269.0	no	2	121.0	yes	0.2
7	65	male	asymptomatic (4)	120.0	177.0	no	0	140.0	no	0.4
8	56	male	non-anginal pain (3)	130.0	256.0	yes	2	142.0	yes	0.6
9	59	male	asymptomatic (4)	110.0	239.0	no	2	142.0	yes	1.2
10	60	male	asymptomatic (4)	140.0	293.0	no	2	170.0	no	1.2
11	63	female	asymptomatic (4)	150.0	407.0	no	2	154.0	no	4.0
12	59	male	asymptomatic (4)	135.0	234.0	no	0	161.0	no	0.5
13	53	male	asymptomatic (4)	142.0	226.0	no	2	111.0	yes	0.0
14	44	male	non-anginal pain (3)	140.0	235.0	no	2	180.0	no	0.0
15	61	male	typical angina (1)	134.0	234.0	no	0	145.0	no	2.6
16	57	female	asymptomatic (4)	128.0	303.0	no	2	159.0	no	0.0
17	71	female	asymptomatic (4)	112.0	149.0	no	0	125.0	no	1.6
18	46	male	asymptomatic (4)	140.0	311.0	no	0	120.0	yes	1.8
19	53	male	asymptomatic (4)	140.0	203.0	yes	2	155.0	yes	3.1
20	64	male	typical angina (1)	110.0	211.0	no	2	144.0	yes	1.8
21	40	male	typical angina (1)	140.0	199.0	no	0	178.0	yes	1.4
22	67	male	asymptomatic (4)	120.0	229.0	no	2	129.0	yes	2.6

ENERGY-BASED METHOD

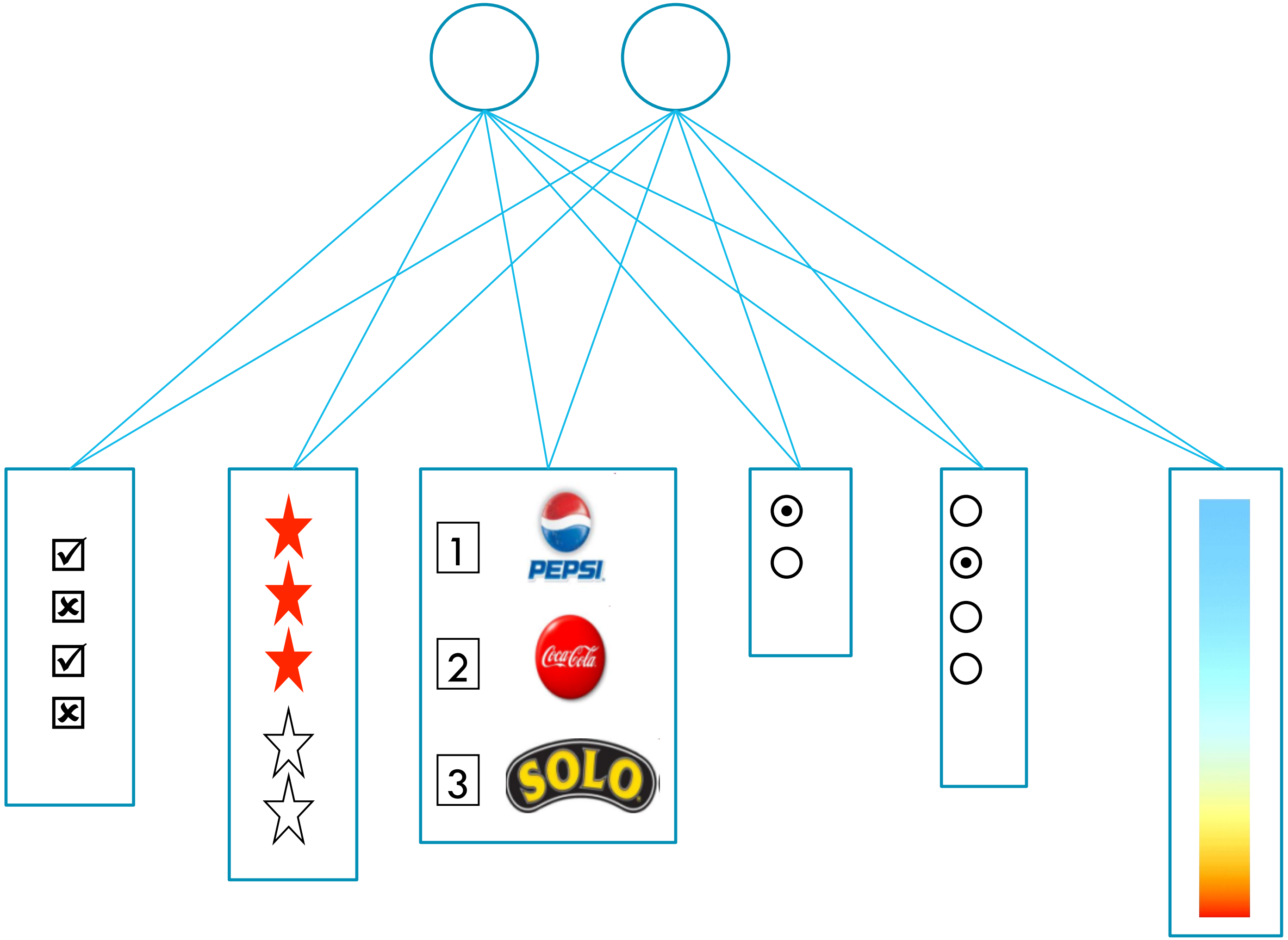


Restricted Boltzmann Machine



$$F(\boldsymbol{x}) = -\sum_i \left(a_i x_i + \sum_k \log(1 + \exp(x_i W_{ik} + b_k)) \right)$$

MIXED-VARIATE RBM (TRAN ET AL, 2011)

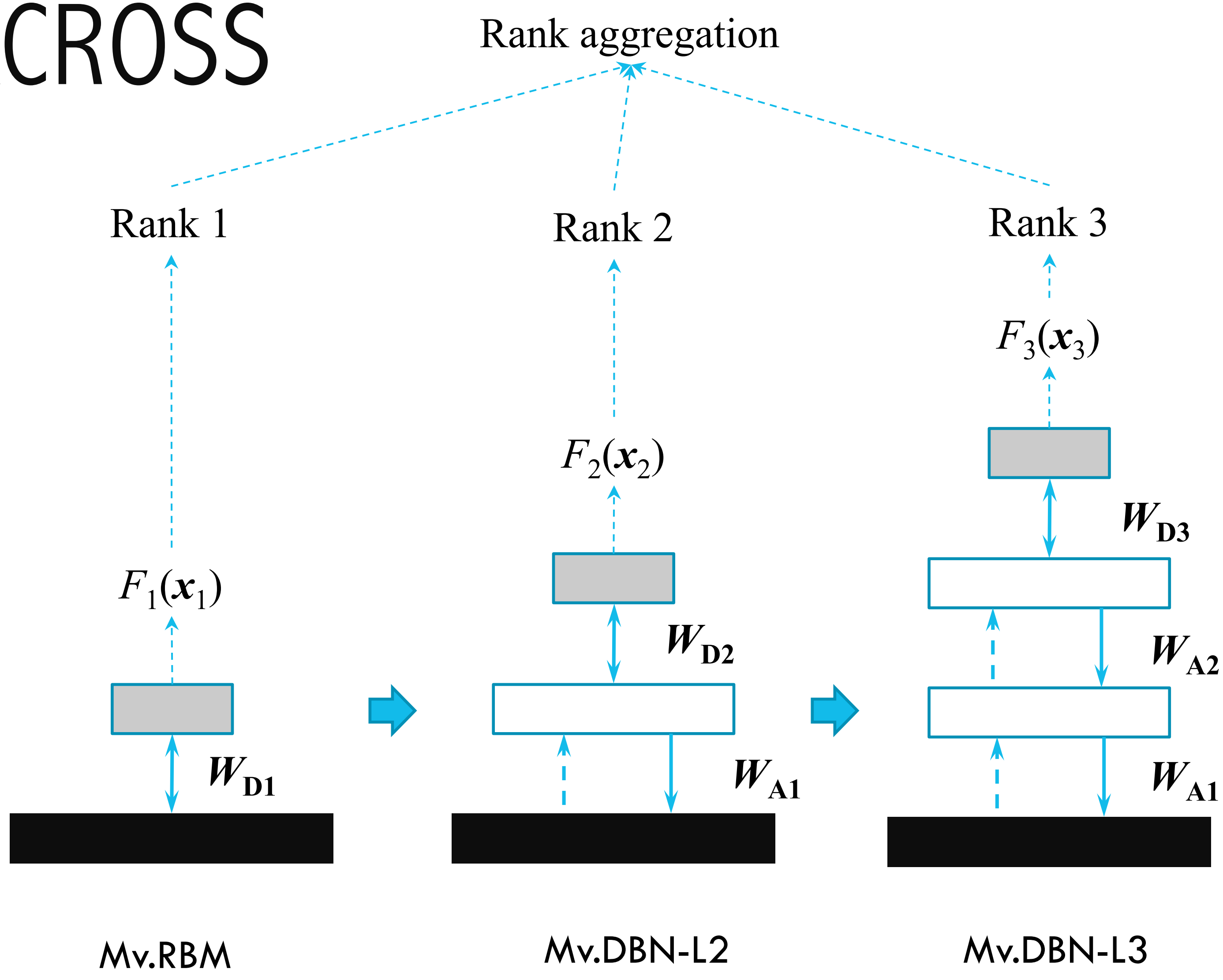


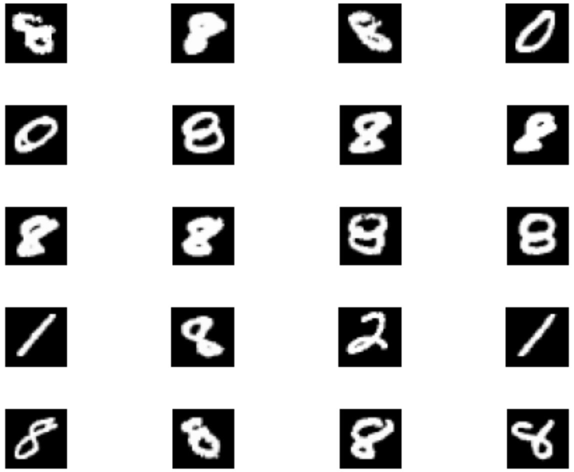
RESULTS OVER REAL DATASETS

Dataset	Single type			mixed-type			
	GMM	OCSVM	PPCA	BMM	ODMAD	GLM-t	Mv.RBM
KDD99-10	0.42	0.54	0.55	–	–	–	**0.71**
Australian Credit	0.74	0.84	0.38	**0.972**	0.942	–	0.90
German Credit	0.86	0.86	0.02	0.934	0.810	–	**0.95**
Heart	0.89	0.76	0.64	0.872	0.630	0.72	**0.94**
Thoracic Surgery	0.71	0.71	0.70	**0.939**	0.879	–	0.90
Auto MPG	**1.00**	**1.00**	0.67	0.625	0.575	0.64	**1.00**
Contraceptive	0.62	0.84	0.02	0.673	0.523	–	**0.91**
Average	*0.75*	*0.79*	*0.43*	*0.84*	*0.73*	*0.68*	***0.91***

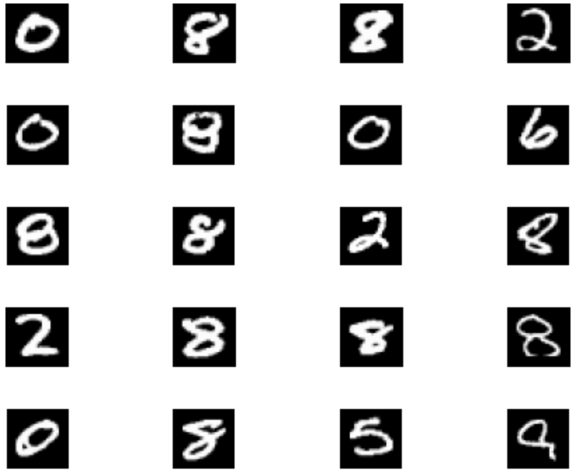
ABNORMALITY ACROSS ABSTRACTIONS

$$\bar{r}_i(p) = \left(\sum_{l=1}^{L} r_{li}^{p}\right)^{1/p}$$

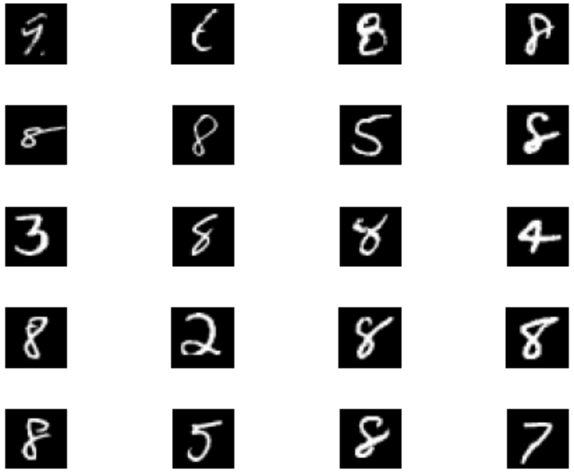




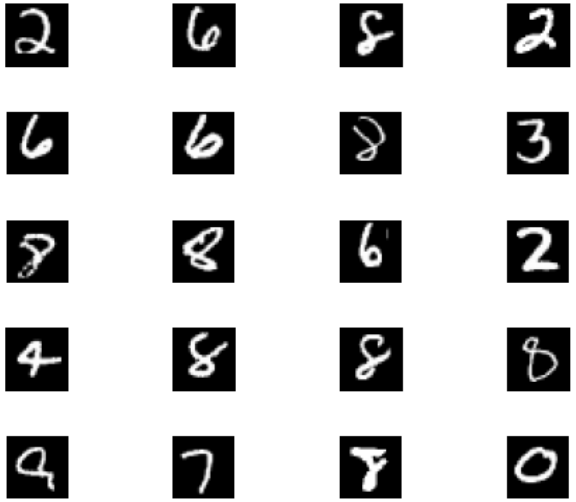
(k-NN, errors = 15/20)



(RBM, errors = 10/20)



(DBN-L2, errors = 12/20)



(MAD-L2p2, errors = 8/20)

AGENDA

Part I: Introduction to deep learning

Part II: Anomaly detection

- Multichannel
- Unusual mixed-data co-occurrence
- Object lifetime model

Part III: Software vulnerabilities

OBJECT LIFETIME MODELS

Objects with a life

- User
- Devices
- Account

Detect unusual behavior at a given time given object's history

- I.e., (Low) conditional probability of next event/action/observation given the history

Two properties:

- Irregular time by internal activities
- Intervention by external agents

DEEPEVOLVE: A MODEL OF EVOLVING BODY

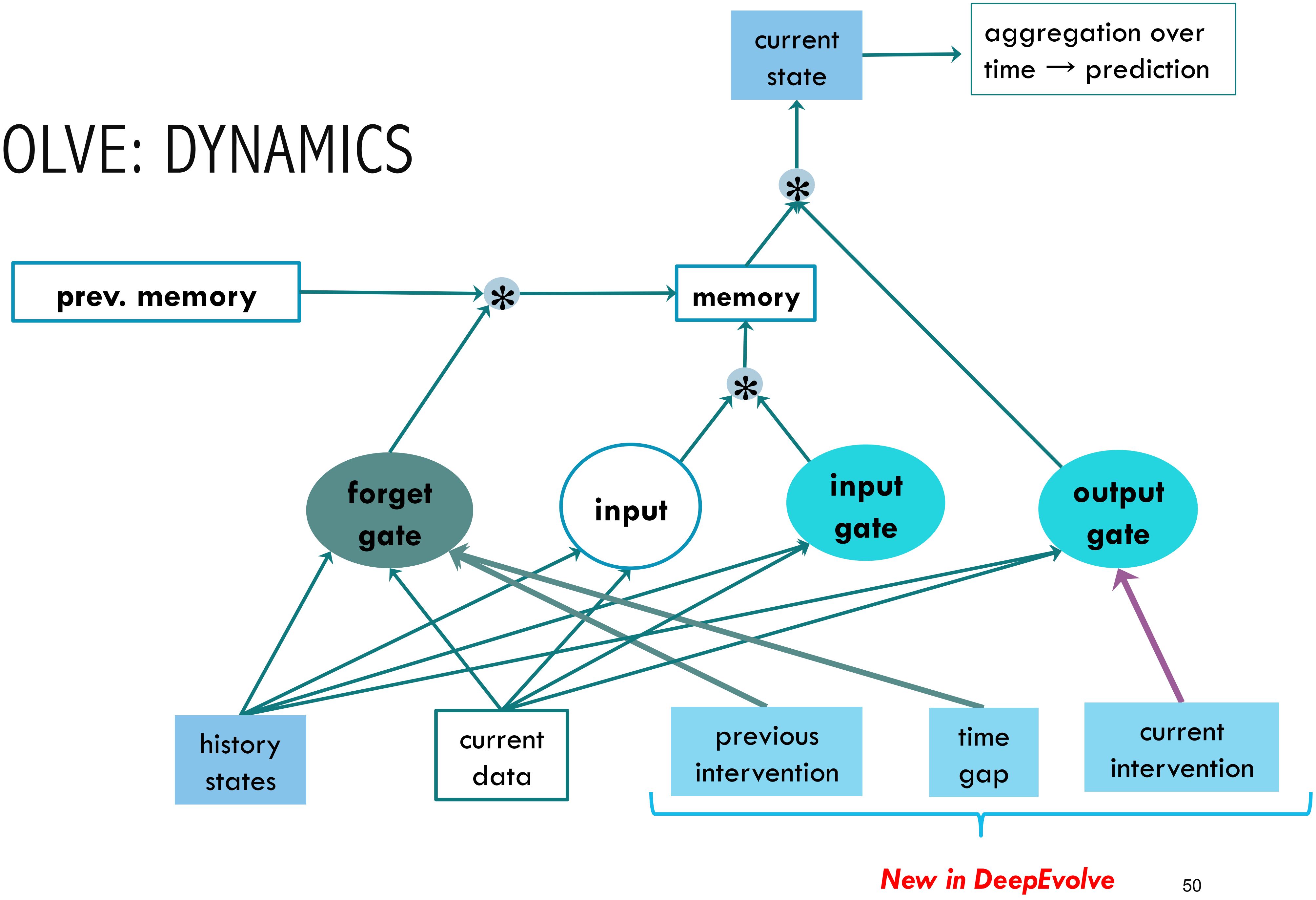
States are a dynamic memory process → LSTM moderated by time and intervention

Discrete observations → vector embedding

Time and previous intervention → "forgetting" of old states

Current intervention → controlling the current states

DEEPEVOLVE: DYNAMICS



END OF PART II

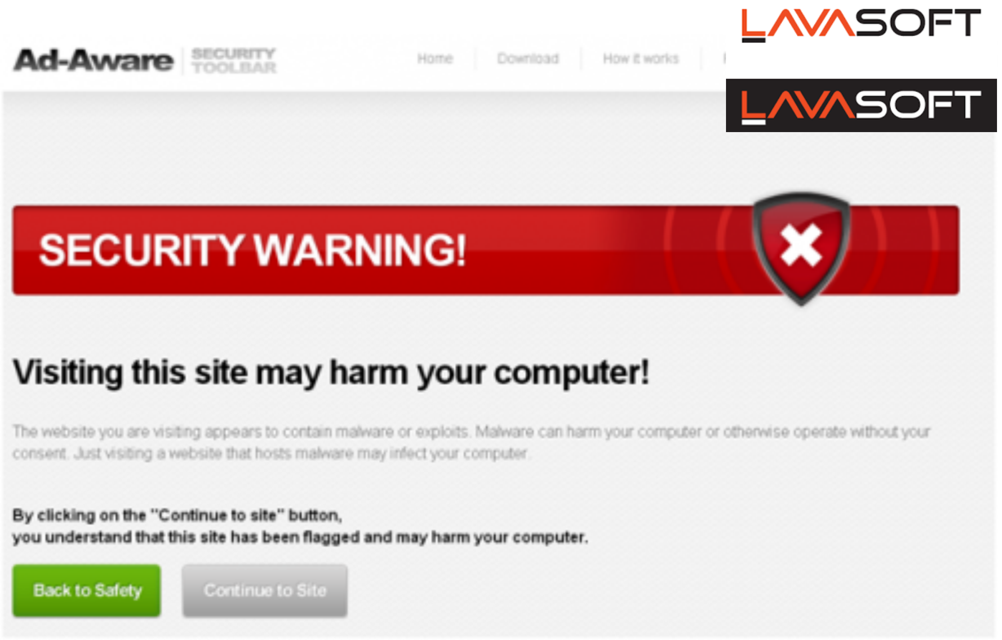
AGENDA

Part I: Introduction to deep learning

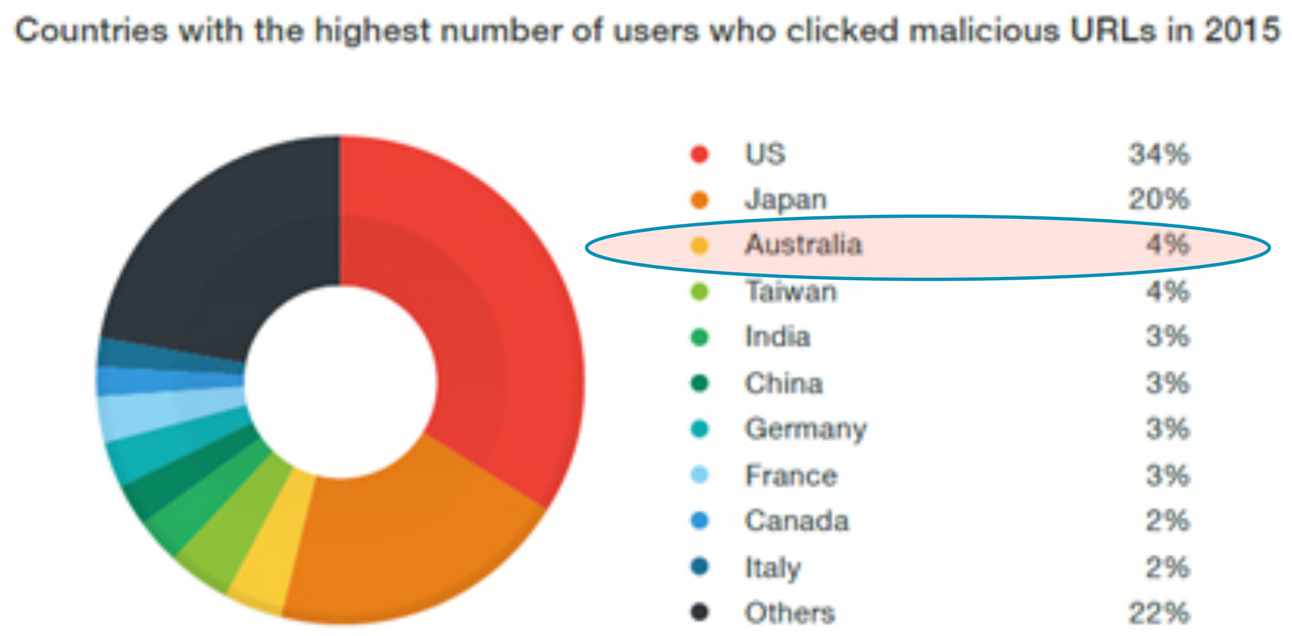
Part II: Anomaly detection

Part III: Software vulnerabilities

- Malicious URL detection
- Unusual source code
- Code vulnerabilities



MALICIOUS URL CLASSIFICATION



http://www.indiainfoline.com/article/news-sector-information-technology/india-ranks-4th-in-highest-users-who-clicked-malicious-urls-in-2015-trend-micro-116052700684_1.html

TRADITIONAL METHOD: FEATURE ENGINEERING + CLASSIFIER

Protocols

Domains, countries

IP-based analysis

Lexical analysis

Query analysis

Handling shortening & dynamically-generated queries

N-grams

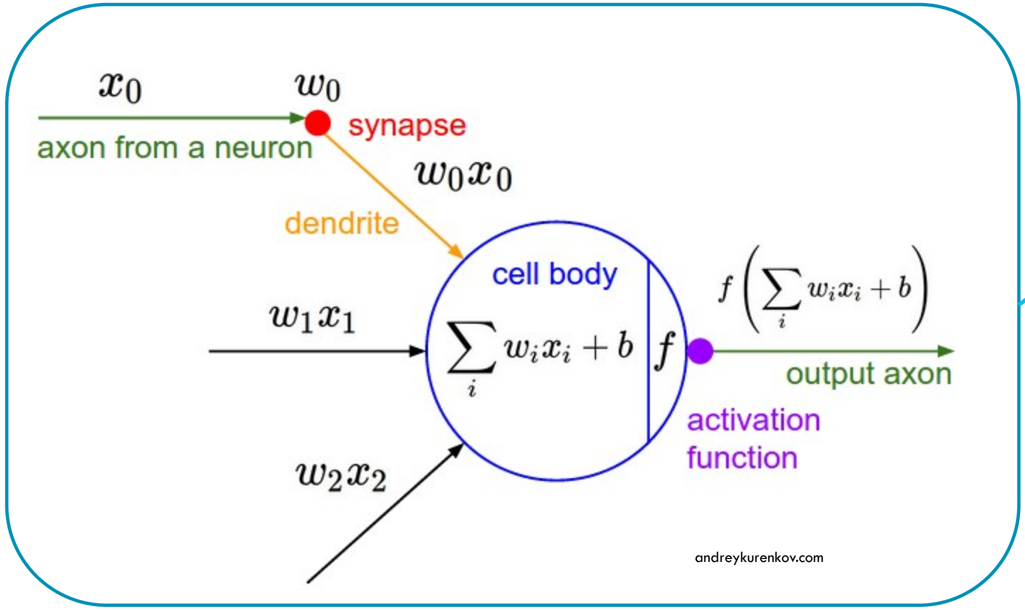
Special characters

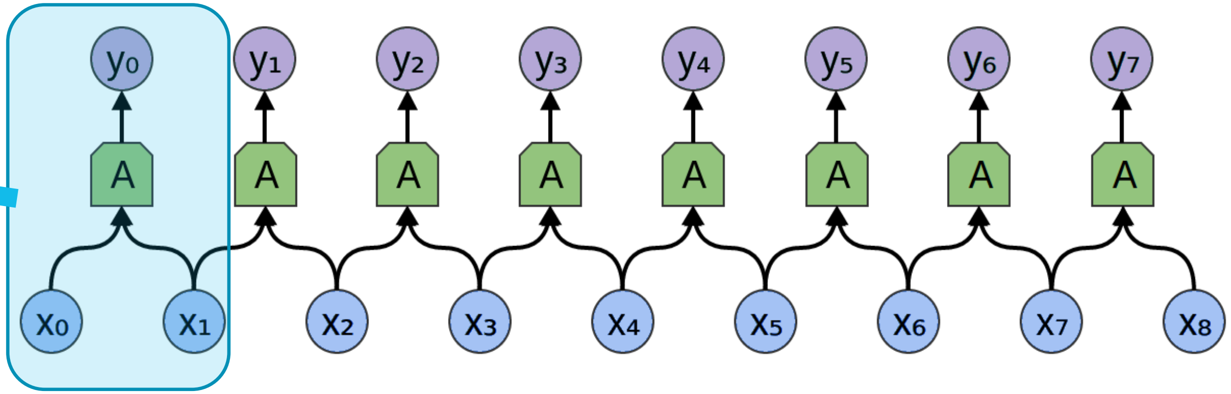
Blacklist

NEW METHOD: LEARNABLE CONVOLUTION AS FEATURE DETECTOR

$$y_i = \sum_c \boxed{K(c)} x_{i+c}$$

Learnable kernels



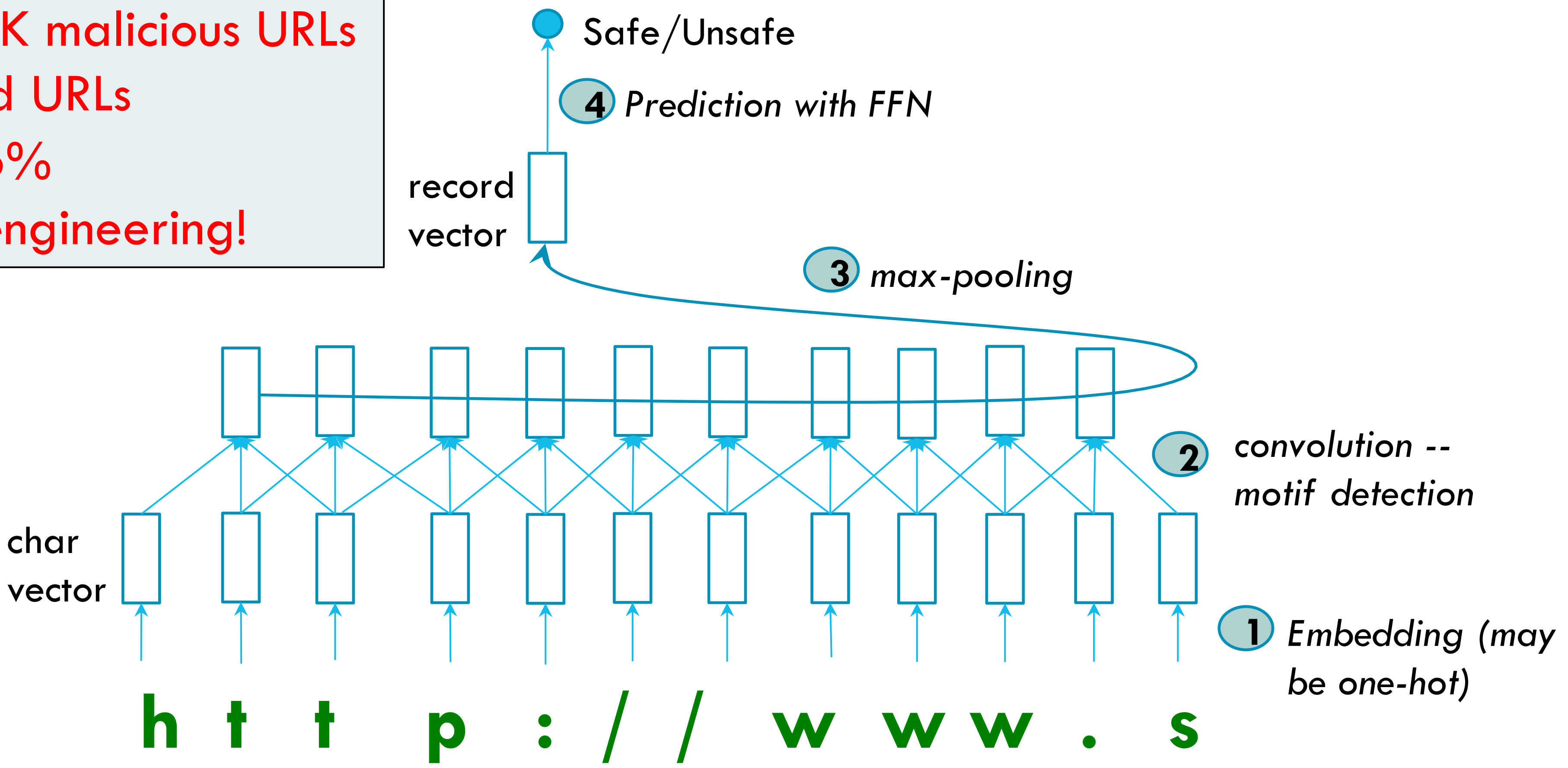


http://colah.github.io/posts/2015-09-NN-Types-FP/

Feature detector, often many

END-TO-END MODEL OF MALICIOUS URLS

Train on 900K malicious URLs
1,000K good URLs
Accuracy: 96%
No feature engineering!



AGENDA

Part I: Introduction to deep learning

Part II: Anomaly detection

Part III: Software vulnerabilities

- Malicious URL detection
- Unusual source code
- Code vulnerabilities

SOFTWARE ANALYTICS

DATA-DRIVEN SOFTWARE ENGINEERING

PROJECT: SAMSUNG GLOBAL REACH

http://www.bentoaktechnologies.com/Images/code_scrn.jpg

MOTIVATIONS

Software is eating the world.

IoT development is exploding. Software security is an extremely critical issue

Vulerable source files: 0.3-5%, depending on code review policy & quality ot code.

General approach: Machine learning instead of human manual effort and programming heuristics

Many software metrics have been found:

- Bugs, code complexity, churn rate, developer network activity metrics, fault history metrics,

Question: can machine learn all of these by itself?

APPROACH: CODE MODELING

Open source code is massive

Bad coding is often the sign of bugs and security holes

Malicious code may be different from the safe code

Ideas:

- A code model assigns probability to a piece of code
- Given the code context, if conditional probability of a code piece per token is low compared to the rest → unusual code → more likely to contain defects or security vulnerability

A DEEP LANGUAGE MODEL FOR SOFTWARE CODE (DAM ET AL, FSE'16 SE+NL)

A good language model for source code would capture the long-term dependencies

The model can be used for various prediction tasks, e.g. defect prediction, code duplication, bug localization, etc.

UNIVERSITY OF WOLLONGONG AUSTRALIA

DEAKIN UNIVERSITY AUSTRALIA

CHARACTERISTICS OF SOFTWARE CODE

Repetitiveness

- E.g. for (int i = 0; i < n; i++)

Localness

- E.g. *for (int size* may appear more often that *for (int i* in some source files.

Rich and explicit structural information

- E.g. nested loops, inheritance hierarchies

Long-term dependencies

- *try* and *catch* (in Java) or file *open* and *close* are not immediately followed each other.

A LANGUAGE MODEL FOR SOFTWARE CODE

Given a code sequence s= <w_1, …, w_k>, a language model estimate the probability distribution P(s):

$$P(s) = P(w_1) \prod_{t=2}^{k} P\left(w_t \mid \boldsymbol{w}_{1:t-1}\right)$$

where $\boldsymbol{w}_{1:t-1} = (w_1, w_2, ..., w_{t-1})$ is the historical *context* used to estimate the probability of the next code token w_t.

TRADITIONAL MODEL: N-GRAMS

Truncates the history length to n-1 words (usually 2 to 5 in practice)

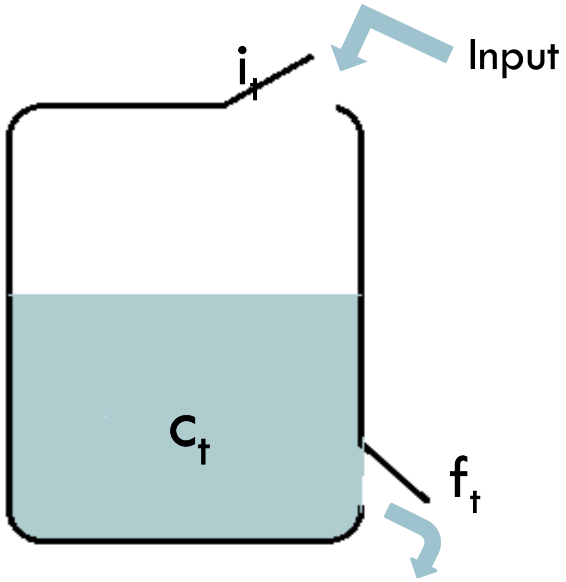
Useful and intuitive in making use of repetitive sequential patterns in code

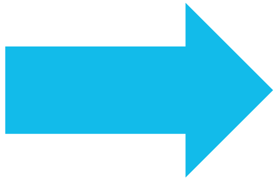
Context limited to a few code elements

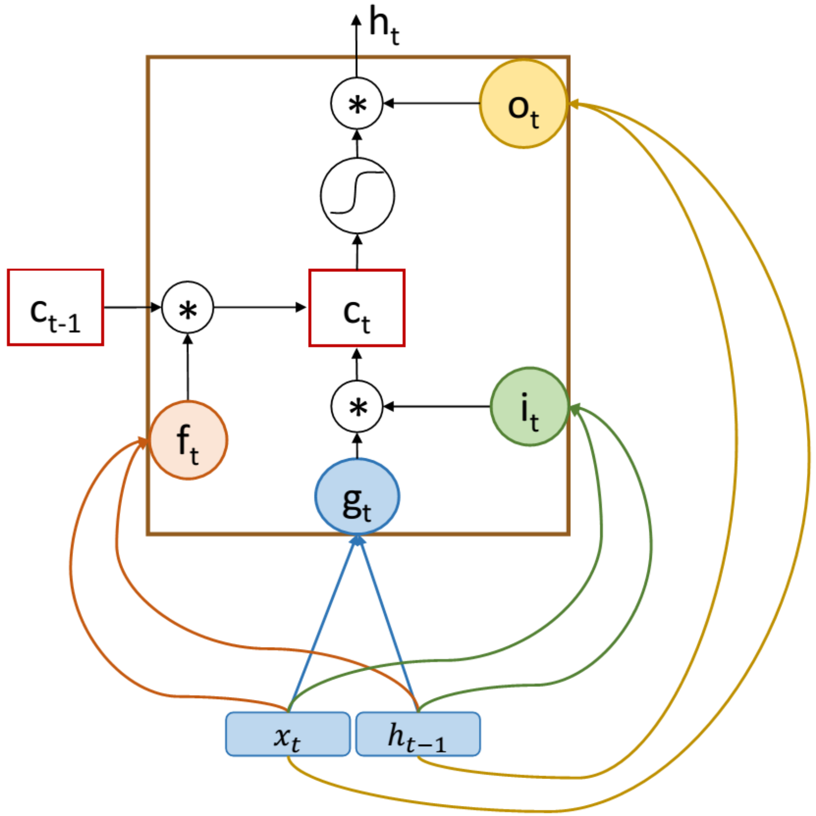
- Not sufficient in complex SE prediction tasks.

As we read a piece of code, we understand each code token based on our understanding of previous code tokens, i.e. the information persists.

NEW METHOD: LONG SHORT-TERM MEMORY (LSTM)

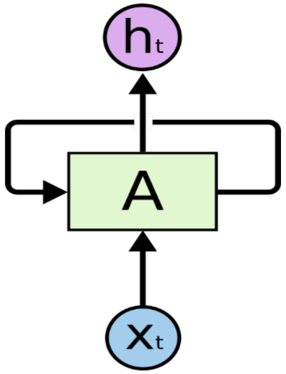






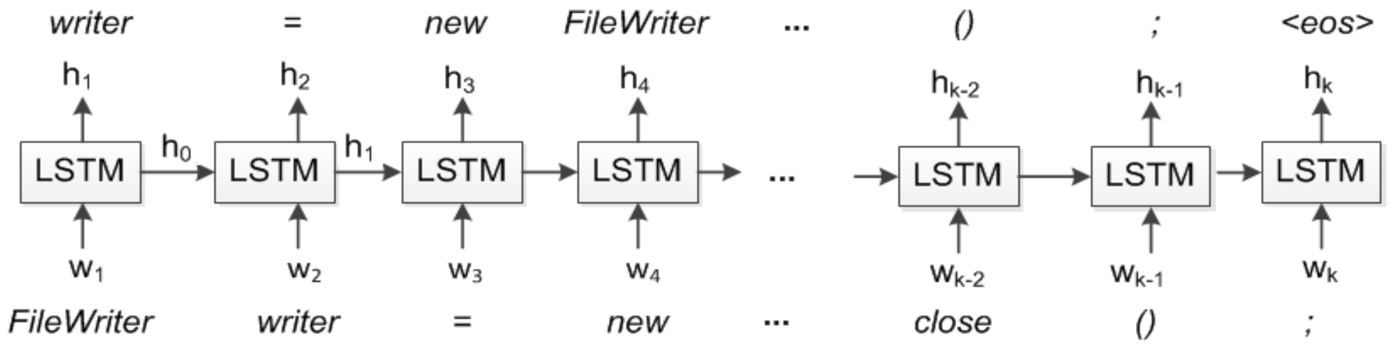
$$i_t = \sigma\left(W_i x_t + U_i h_{t-1} + b_i\right)$$
$$f_t = \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right)$$
$$o_t = \sigma\left(W_o x_t + U_o h_{t-1} + b_o\right)$$
$$c_t = f_t * c_{t-1} + i_t * \tanh\left(W_c x_t + U_c h_{t-1} + b_c\right)$$
$$h_t = o_t * \tanh(c_t)$$

CODE LANGUAGE MODEL




```
FileWriter writer = new FileWriter(file);
writer.write(``This is an example'');
int count = 0;
System.out.prinltln(``Long gap'');
      ......

writer.flush();
writer.close();
```



Previous work has applied RNNs to model software code *(White et al, MSR 2015)*

RNNs however do not capture the long-term dependencies in code

EXPERIMENTS

Built dataset of 10 Java projects: Ant, Batik, Cassandra, Eclipse-E4, Log4J, Lucene, Maven2, Maven3, Xalan-J, and Xerces.

Comments and blank lines removed. Each source code file is tokenized to produce a sequence of code tokens.

- Integers, real numbers, exponential notation, hexadecimal numbers replaced with <num> token, and constant strings replaced with <str> token.
- Replaced less "popular" tokens with <unk>

Code corpus of 6,103,191 code tokens, with a vocabulary of 81,213 unique tokens.

EXPERIMENTS (CONT.)

sent-len	embed-dim	RNN	LSTM	improv %
10	50	13.49	12.86	**4.7**
20		10.38	9.66	**6.9**
50		7.93	6.81	**14.1**
100		7.20	6.40	**11.1**
200		6.64	5.60	**15.7**
500		6.48	4.72	**27.2**
100	20	7.96	7.11	**10.7**
	50	7.20	6.40	**11.1**
	100	7.23	5.72	**20.9**
	200	9.14	5.68	**37.9**

Table 1: Perplexity on test data (the smaller the better).

Both RNN and LSTM improve with more training data (whose size grows with sequence length).

LSTM consistently performs better than RNN: 4.7% improvement to 27.2% (varying sequence length), 10.7% to 37.9% (varying embedding size).

AGENDA

Part I: Introduction to deep learning

Part II: Anomaly detection

Part III: Software vulnerabilities

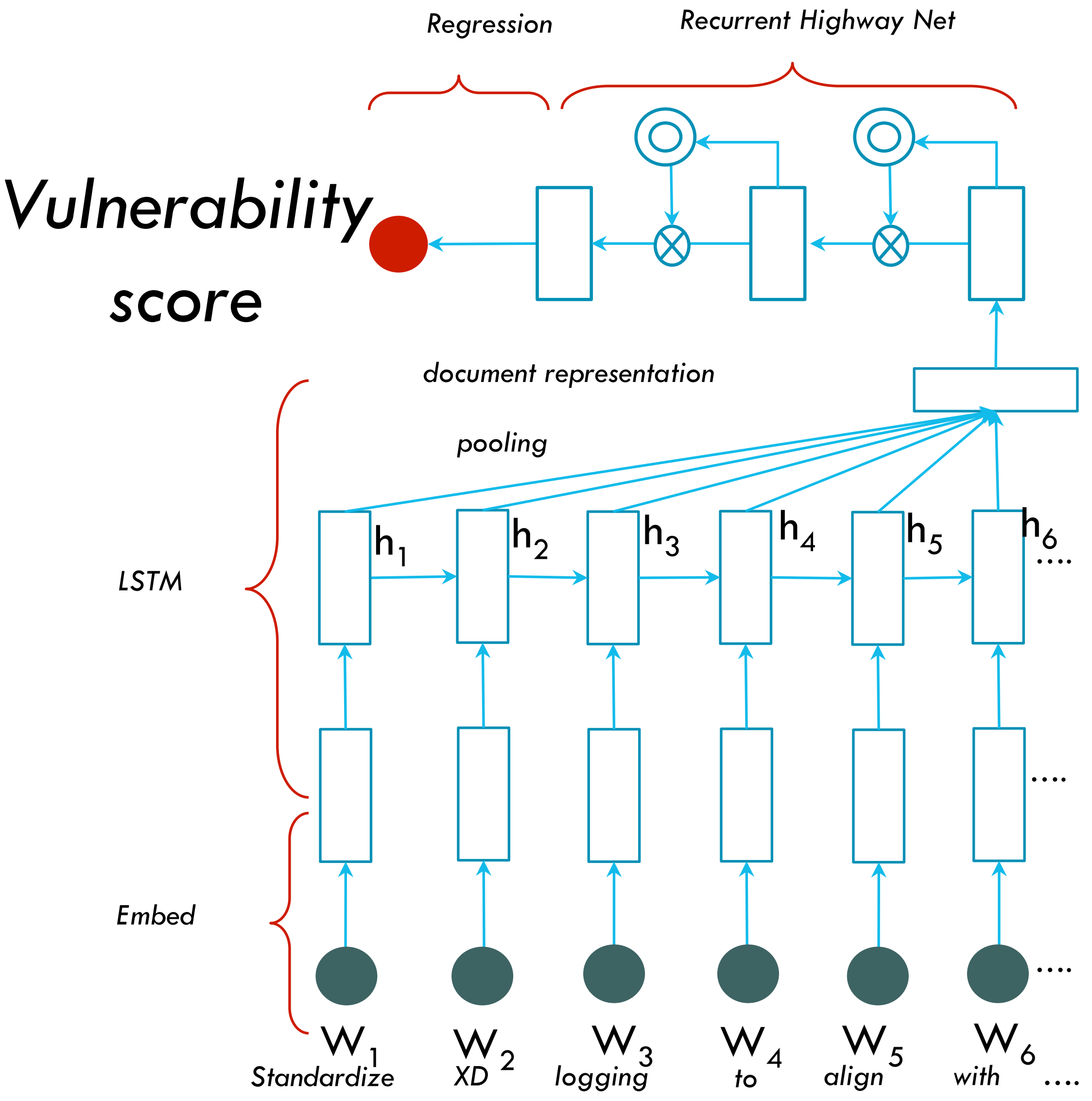
- Malicious URL detection
- Unusual source code
- Code vulnerabilities

METHOD-1: LD-RNN FOR SEQUENCE CLASSIFICATION
(CHOETKIERTIKUL ET AL, WORK IN PROGRESS)

LD = Long Deep

LSTM for document representation

Highway-net with tied parameters for vulnerability score

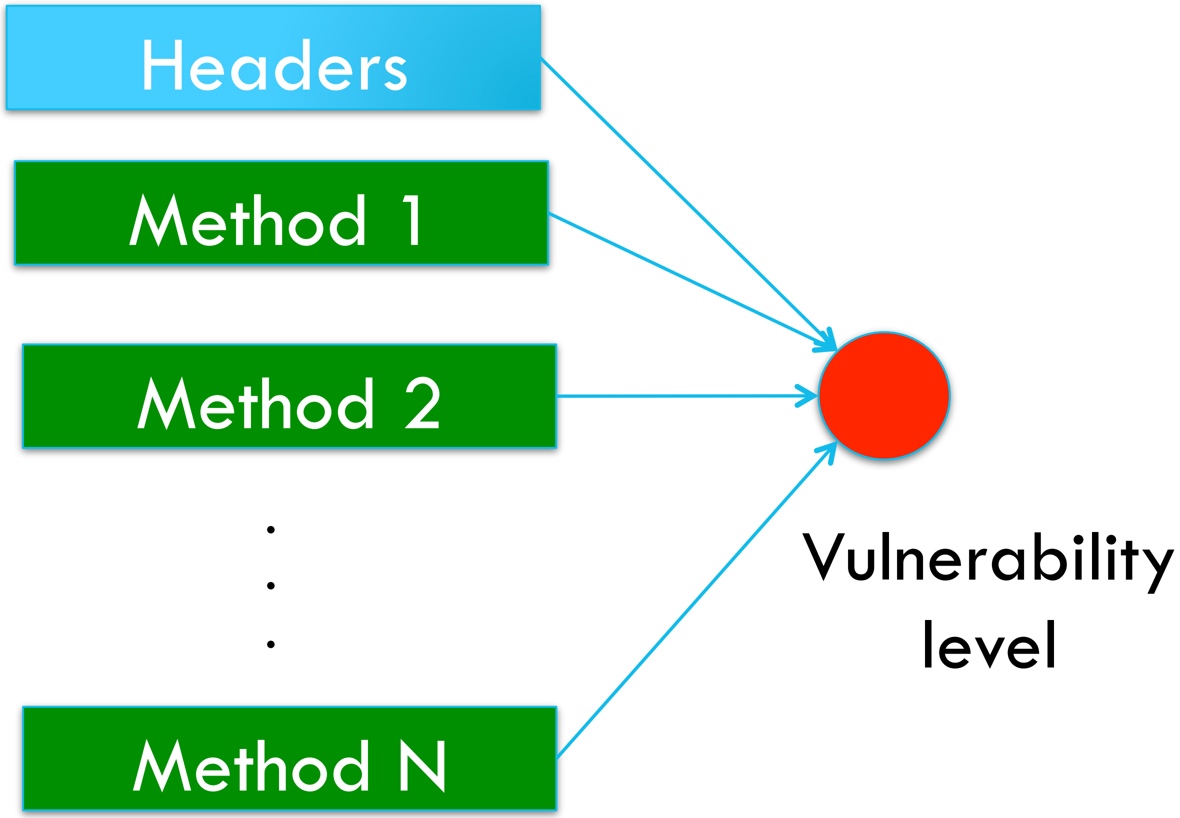


METHOD-2: DEEP SEQUENTIAL MULTI-INSTANCE LEARNING

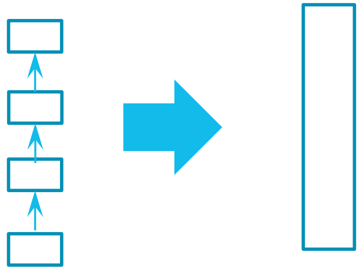
Code file as a bag

Methods as instances

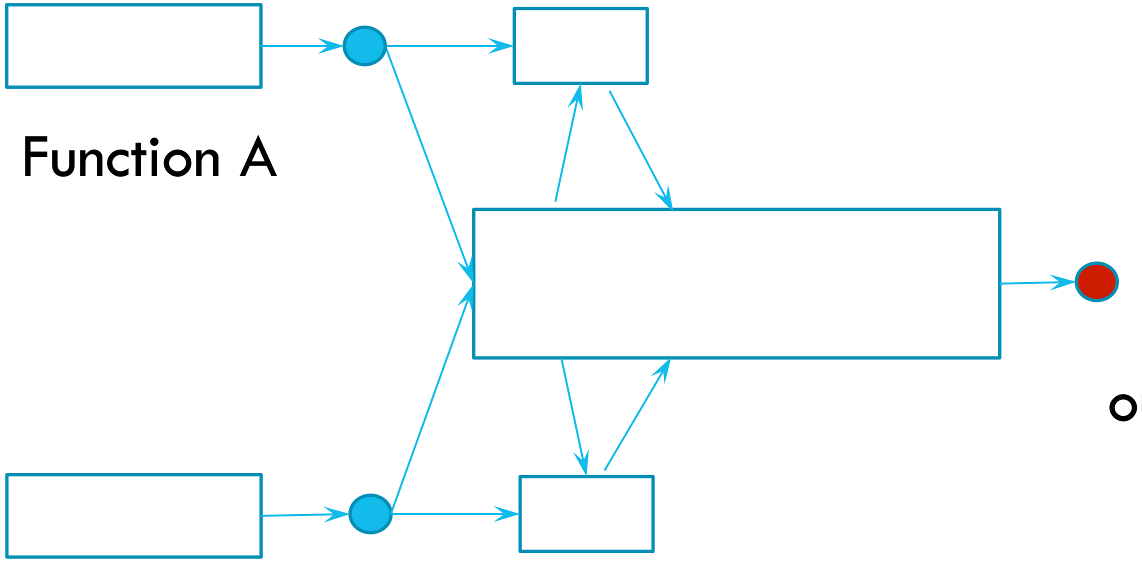
Data are sequential



COLUMN BUNDLE FOR N-TO-1 MAPPING
(PHAM ET AL, WORK IN PROGRESS)

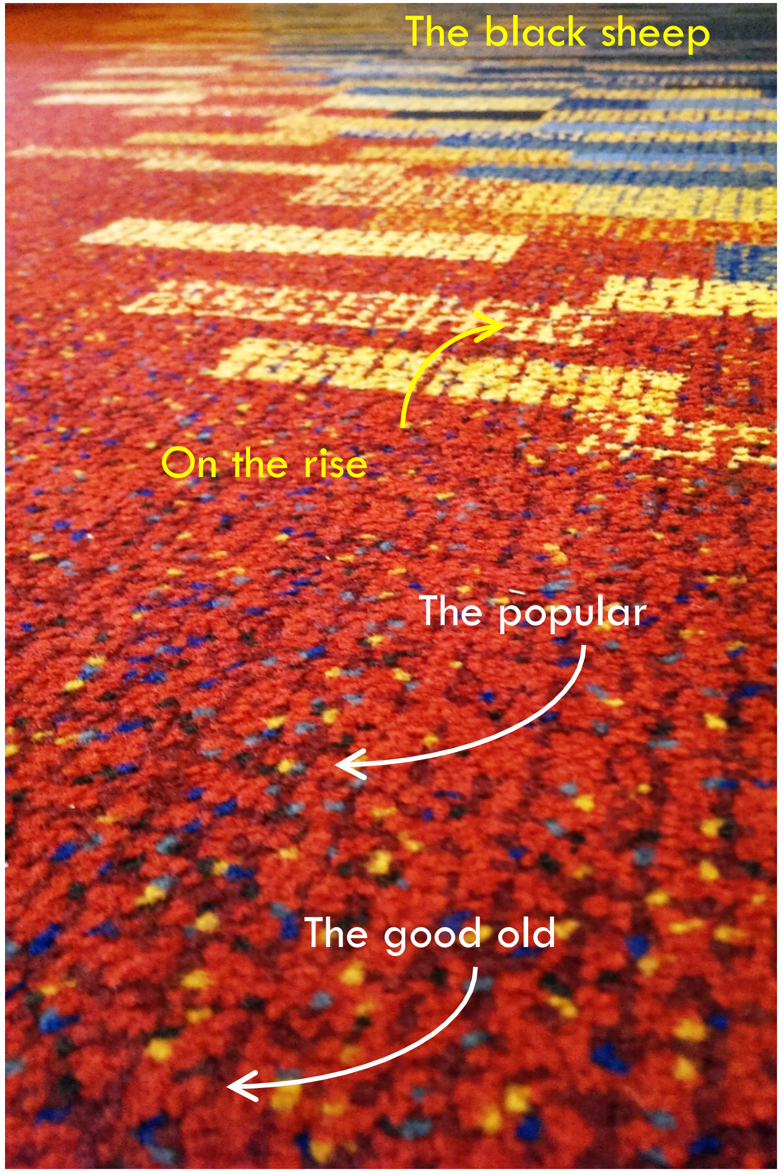


Column representation



Thank you!

http://ahsanqawl.com/2015/10/qa/



- Group theory (Lie algebra, renormalisation group, spin-class)

- Differential Turing machines
- Memory, attention & reasoning
- Reinforcement learning & planning
- Lifelong learning

- Dropouts & batch-norm
- Rectifier linear transforms & skip-connections
- Highway nets, LSTM & CNN

- Representation learning (RBM, DBN, DBM, DDAE)
- Ensemble
- Back-propagation
- Adaptive stochastic gradient

WHY DEEP LEARNING WORKS: PRINCIPLES

Expressiveness

- Can represent the complexity of the world → Feedforward nets are universal function approximator
- Can compute anything computable → Recurrent nets are Turing-complete

Learnability

- Have mechanism to learn from the training signals → Neural nets are highly trainable

Generalizability

- Work on unseen data → Deep nets systems work in the wild (Self-driving cars, Google Translate/Voice, AlphaGo)

WHEN DEEP LEARNING WORKS

Lots of data (e.g., millions)

Strong, clean training signals (e.g., when human can provide correct labels – cognitive domains).

- Andrew Ng of Baidu: When humans do well within sub-second.

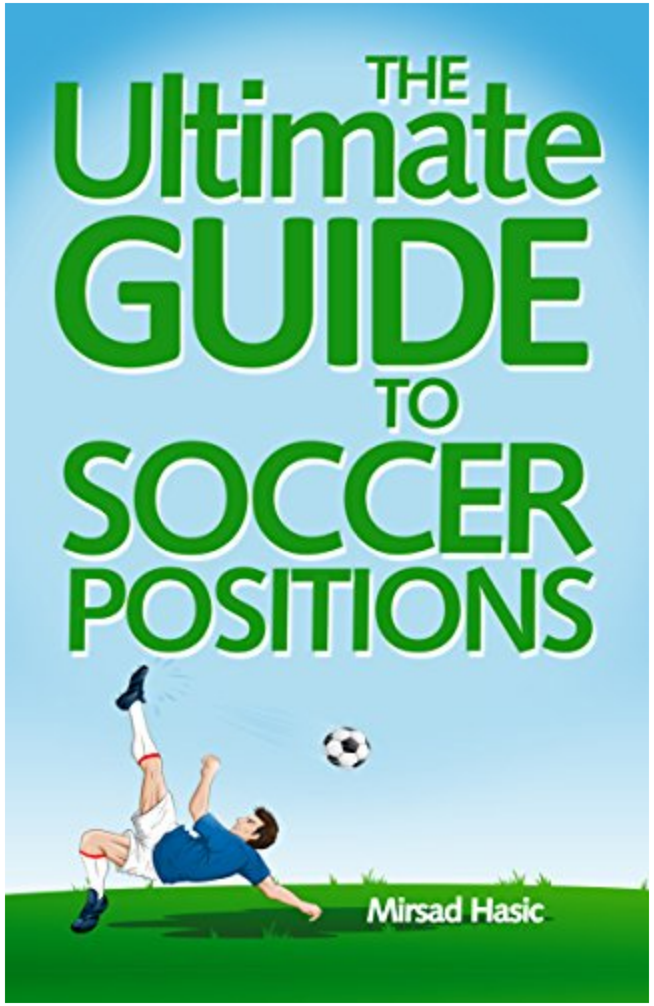
Data structures are well-defined (e.g., image, speech, NLP, video)

Data is compositional (luckily, most data are like this)

The more primitive (raw) the data, the more benefit of using deep learning.

BONUS: HOW TO POSITION

"[…] the dynamics of the game will evolve. In the long run, the right way of playing football is to position yourself intelligently and to wait for the ball to come to you. You'll need to run up and down a bit, either to respond to how the play is evolving or to get out of the way of the scrum when it looks like it might flatten you." (*Neil Lawrence, 7/2015, now with Amazon*)



http://inverseprobability.com/2015/07/12/Thoughts-on-ICML-2015/

THE ROOM IS WIDE OPEN

Architecture engineering

Non-cognitive apps

Unsupervised learning

Graphs

Learning while preserving privacy

Modelling of domain invariance

Better data efficiency

Multimodality

Learning under adversarial stress

Better optimization

Going Bayesian

http://smerity.com/articles/2016/architectures_are_the_new_feature_engineering.html