# Tensor Decompositions for Multi-aspect Graph Analytics and Beyond

Evangelos (Vagelis) Papalexakis
UC Riverside

MADLab @ UCR
Multi Aspect Data

*One World Signal Processing Seminar Series*
*(remotely from Riverside, CA), Nov. 19 2020*

# Roadmap

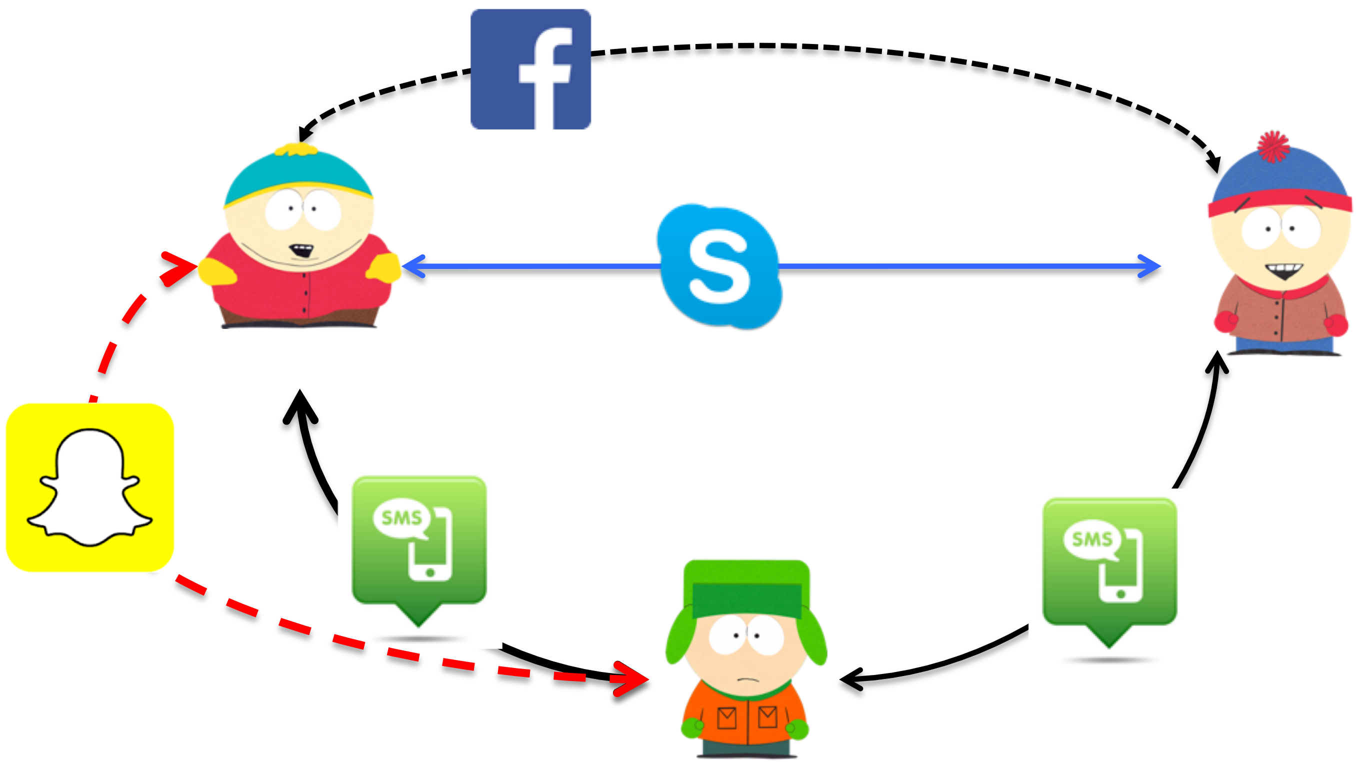

Time-evolving Tensors
& Concept Drift

Conclusion

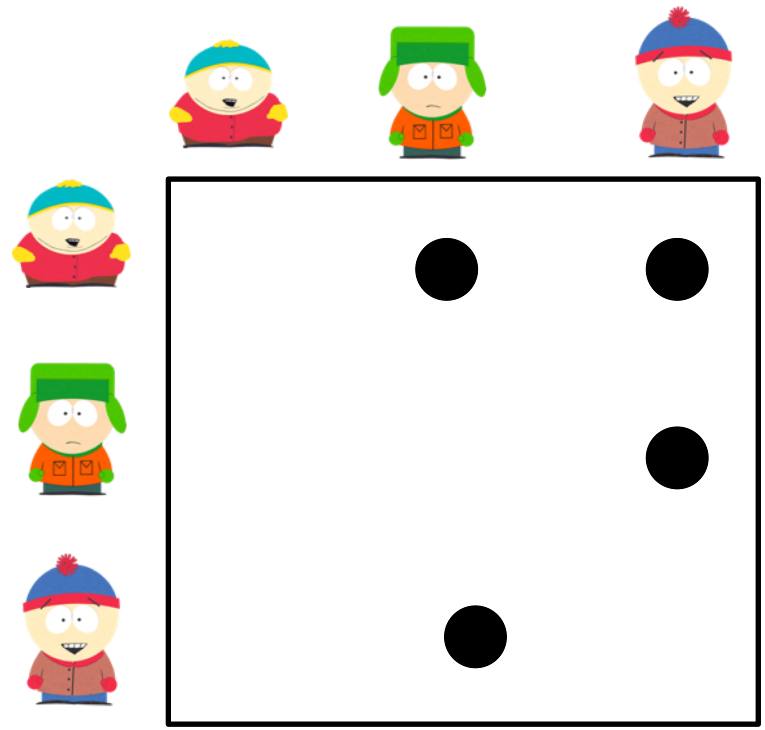
Introduction to
Tensors
& Graph Mining

Adversarial
Machine
Learning

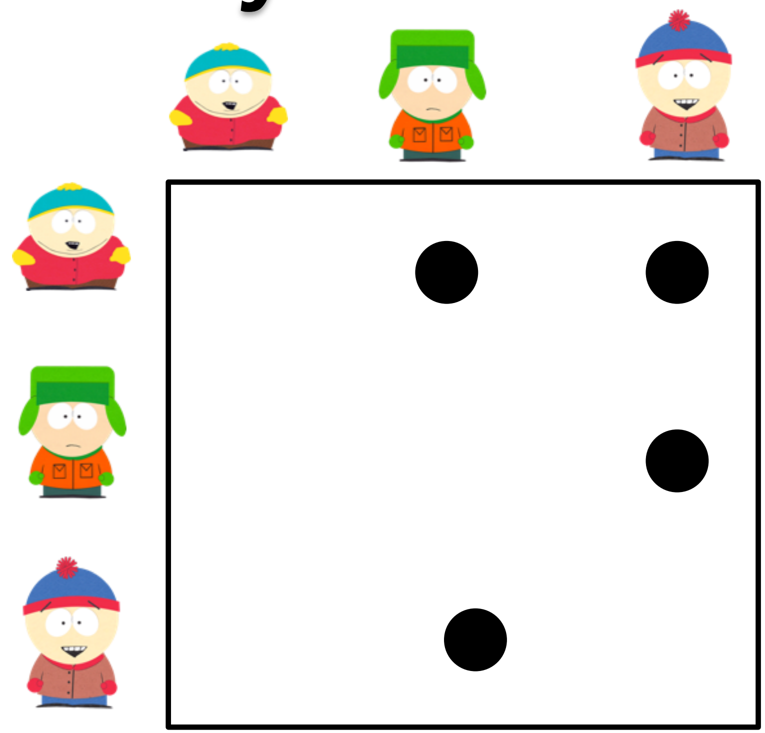
# Multi-View Social Networks

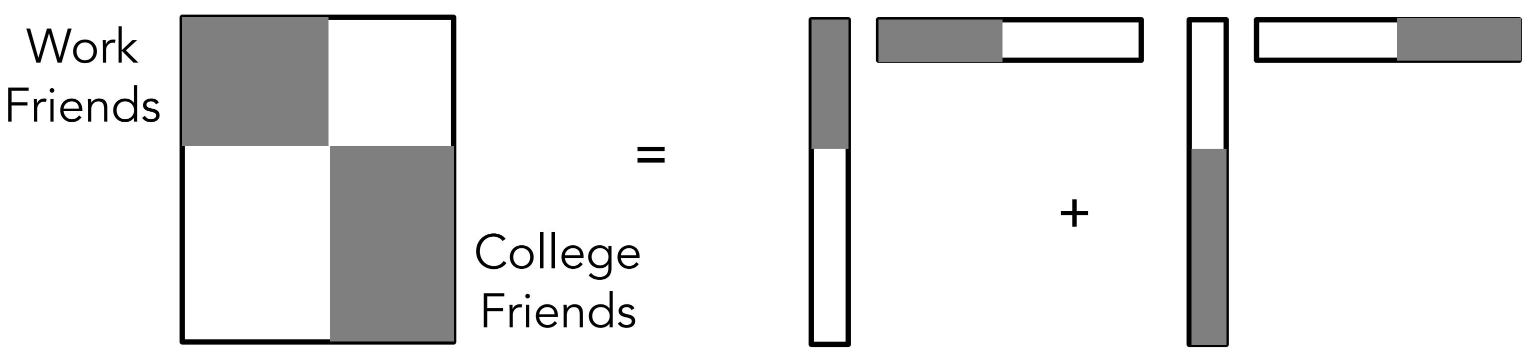


# Social Network Matrix

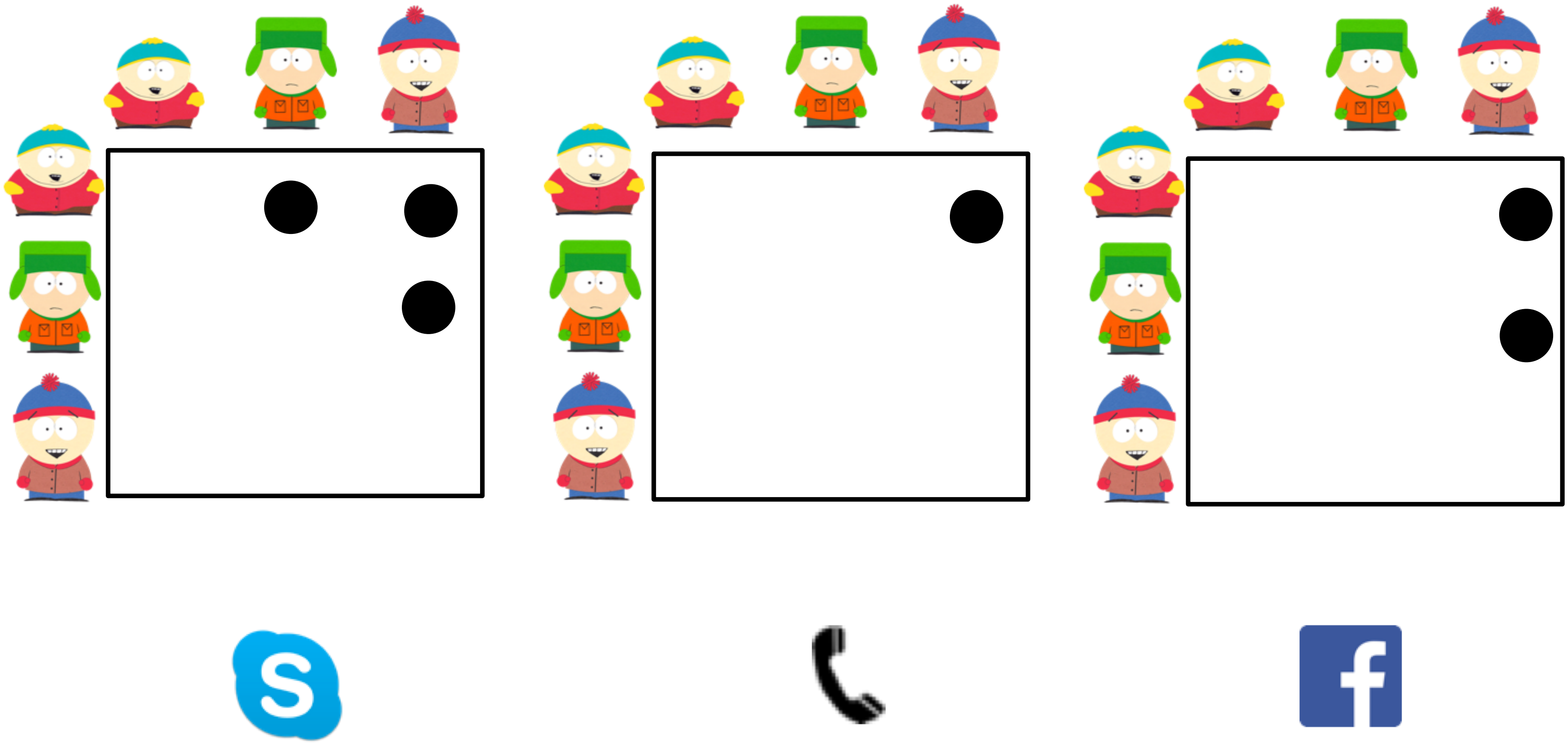


# Adjacency Matrix Analysis





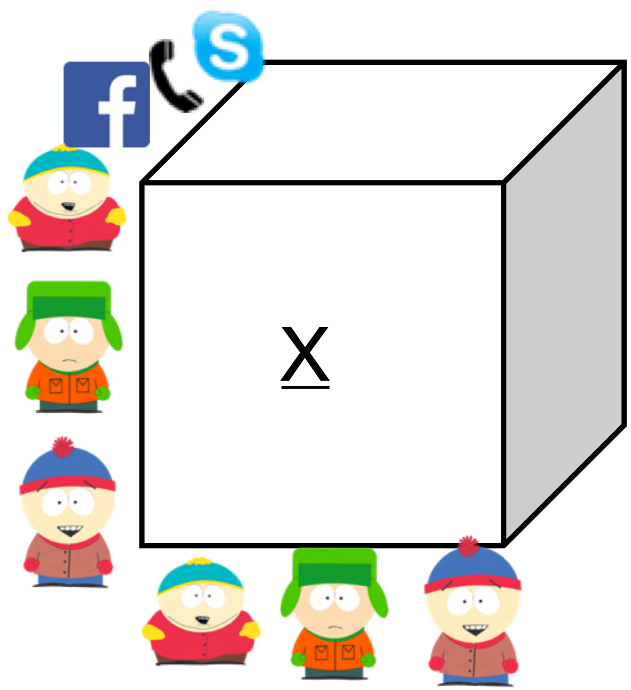
# What about the rest of the views??



If we aggregate, we ignore important structure!!

# Tensors

- Multi-dimensional matrices
- Can naturally model **multi-aspect datasets**
- Long list of applications: Psychometrics, Chemometrics, Signal Processing, Machine Learning, Data Mining



# Survey Papers

IEEE TRANSACTIONS ON SIGNAL PROCESSING, VOL. 65, NO. 13, JULY 1, 2017 3551

## Tensor Decomposition for Signal Processing and Machine Learning

Nicholas D. Sidiropoulos, *Fellow, IEEE*, Lieven De Lathauwer, *Fellow, IEEE*, Xiao Fu, *Member, IEEE*, Kejun Huang, *Member, IEEE*, Evangelos E. Papalexakis, and Christos Faloutsos

*Overview Article*

***Abstract*—Tensors or *multiway arrays* are functions of three or more indices $(i, j, k, \dots)$—similar to matrices (two-way arrays), which are functions of two indices $(r, c)$ for (row, column). Tensors have a rich history, stretching over almost a century, and touching upon numerous disciplines; but they have only recently become ubiquitous in signal and data analytics at the confluence of signal processing, statistics, data mining, and machine learning. This overview article aims to provide a good starting point for researchers and practitioners interested in learning about and working with tensors. As such, it focuses on fundamentals and motivation (using various application examples), aiming to strike an appropriate balance of breadth *and depth* that will enable someone having taken first graduate courses in matrix algebra and probability to get started doing research and/or developing tensor algorithms and software. Some background in applied optimization is useful but not strictly required. The material covered includes tensor rank and rank decomposition; basic tensor factorization models and their relationships and properties (including fairly good coverage of identifiability); broad coverage of algorithms ranging from alternating optimization to stochastic gradient; statistical performance analysis; and applications ranging from source separation to collaborative filtering, mixture and topic modeling, classification, and multilinear subspace learning.**

***Index Terms*—Tensor decomposition, tensor factorization, rank, canonical polyadic decomposition (CPD), parallel factor analysis (PARAFAC), Tucker model, higher-order singular value decomposition (HOSVD), multilinear singular value decomposition (MLSVD), uniqueness, NP-hard problems, alternating optimization, alternating direction method of multipliers, gradient descent, Gauss–Newton, stochastic gradient, Cramér–Rao bound, communications, source separation, harmonic retrieval, speech separation, collaborative filtering, mixture modeling, topic modeling, classification, subspace learning.**

### I. Introduction

TENSORS[1] (of order higher than two) are arrays indexed by three or more indices, say $(i, j, k, \dots)$ – a generalization of matrices, which are indexed by two indices, say $(r, c)$ for (row, column). Matrices are two-way arrays, and there are three- and higher-way arrays (or *higher-order*) tensors.

Tensor algebra has many similarities but also many striking differences with matrix algebra – e.g., low-rank tensor factorization is essentially unique under mild conditions; determining tensor rank is NP-hard, on the other hand, and the best low-rank approximation of a higher rank tensor may not even exist.

Geared towards theoretical & algorithmic understanding

16

## Tensors for Data Mining and Data Fusion: Models, Applications, and Scalable Algorithms

EVANGELOS E. PAPALEXAKIS, University of California Riverside
CHRISTOS FALOUTSOS, Carnegie Mellon University
NICHOLAS D. SIDIROPOULOS, University of Minnesota

Tensors and tensor decompositions are very powerful and versatile tools that can model a wide variety of heterogeneous, multiaspect data. As a result, tensor decompositions, which extract useful latent information out of multiaspect data tensors, have witnessed increasing popularity and adoption by the data mining community. In this survey, we present some of the most widely used tensor decompositions, providing the key insights behind them, and summarizing them from a practitioner's point of view. We then provide an overview of a very broad spectrum of applications where tensors have been instrumental in achieving state-of-the-art performance, ranging from social network analysis to brain data analysis, and from web mining to healthcare. Subsequently, we present recent algorithmic advances in scaling tensor decompositions up to today's big data, outlining the existing systems and summarizing the key ideas behind them. Finally, we conclude with a list of challenges and open problems that outline exciting future research directions.

CCS Concepts: • **General and reference → Document types;** • **Information systems → Information systems applications; Data mining;** • **Computing methodologies → Factorization methods**

Additional Key Words and Phrases: Tensors, tensor decomposition, tensor factorization, multi-aspect data, multi-way analysis

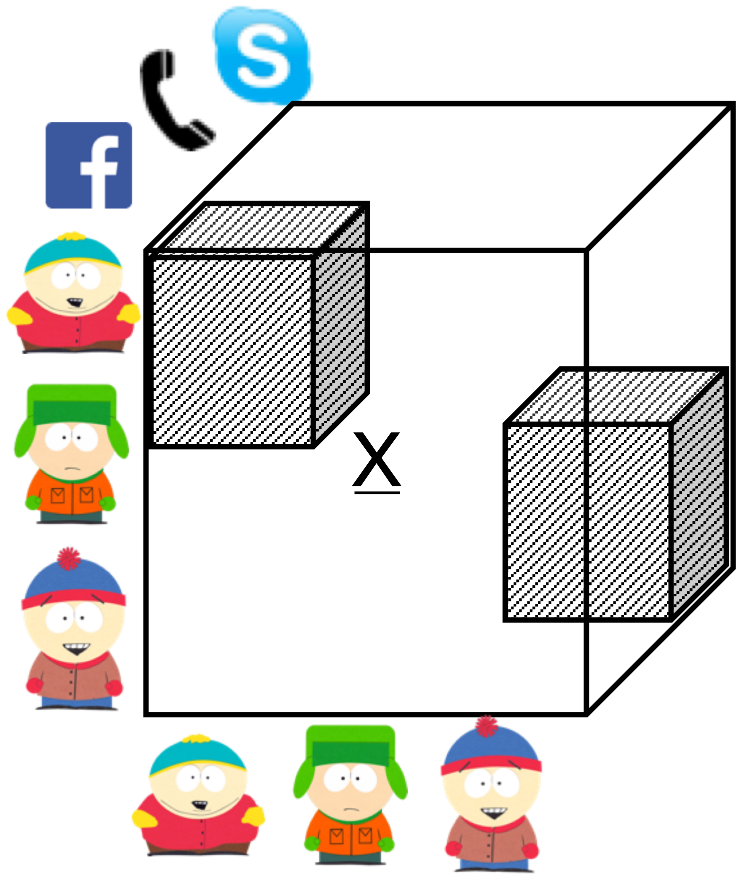
**ACM Reference Format:**
Evangelos E. Papalexakis, Christos Faloutsos, and Nicholas D. Sidiropoulos. 2016. Tensors for data mining and data fusion: Models, applications, and scalable algorithms. ACM Trans. Intell. Syst. Technol. 8, 2, Article 16 (October 2016), 44 pages.
DOI: http://dx.doi.org/10.1145/2915921

### 1. INTRODUCTION

Tensors are multidimensional extensions of matrices. Because of their ability to express multimodal or multiaspect data, they are very powerful tools in applications that inherently create such data. For instance, in online social networks, people tend to interact with each other in a variety of ways: they message each other, they post on each other's pages, and so on. All these different means of interaction are different

Geared towards applications & practitioners

# What are we looking for?

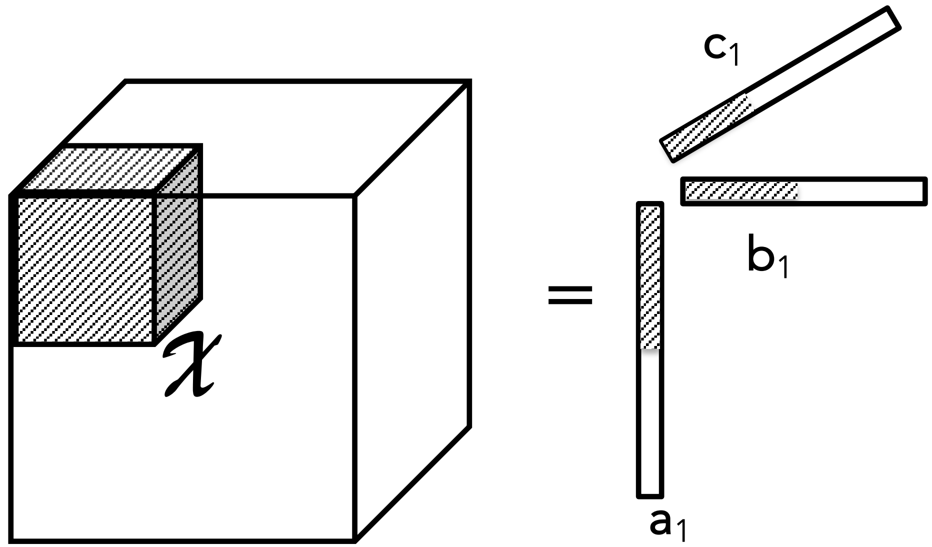



**Blocks** within the data

Subsets / co-clusters of:

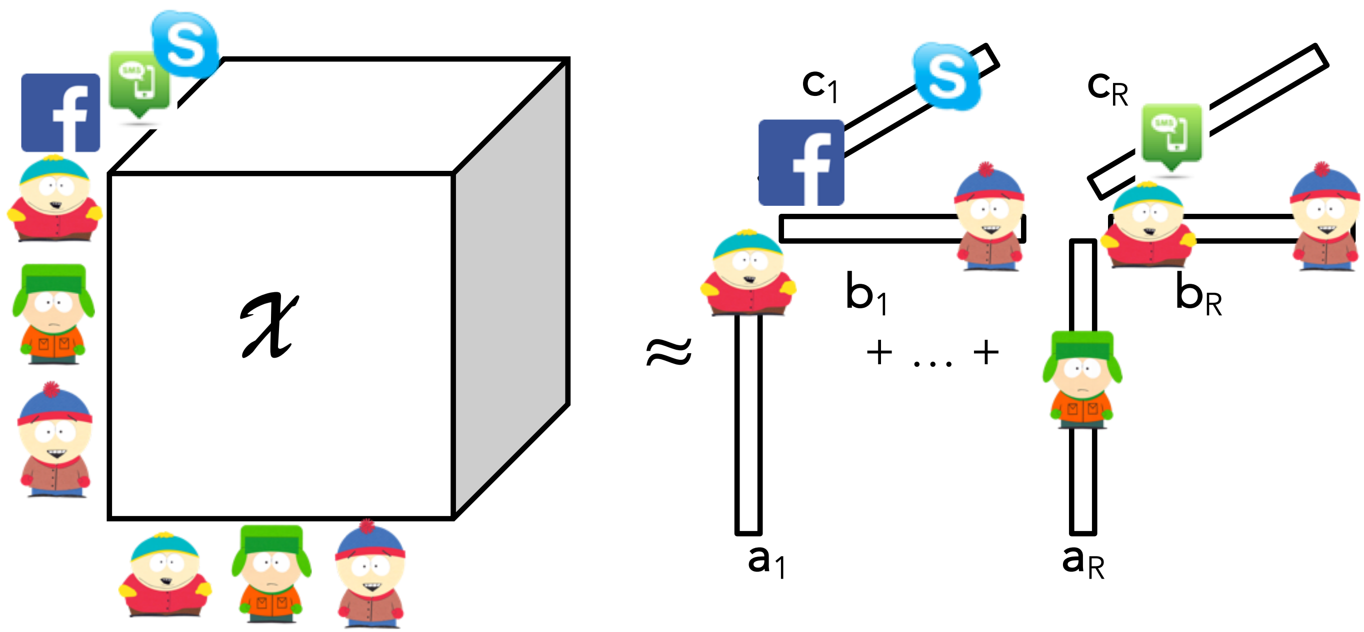
1) Users ("senders")
2) Users ("receivers")
3) Means of communication

# Blocks are rank-one tensors




Direct extension of matrix case!

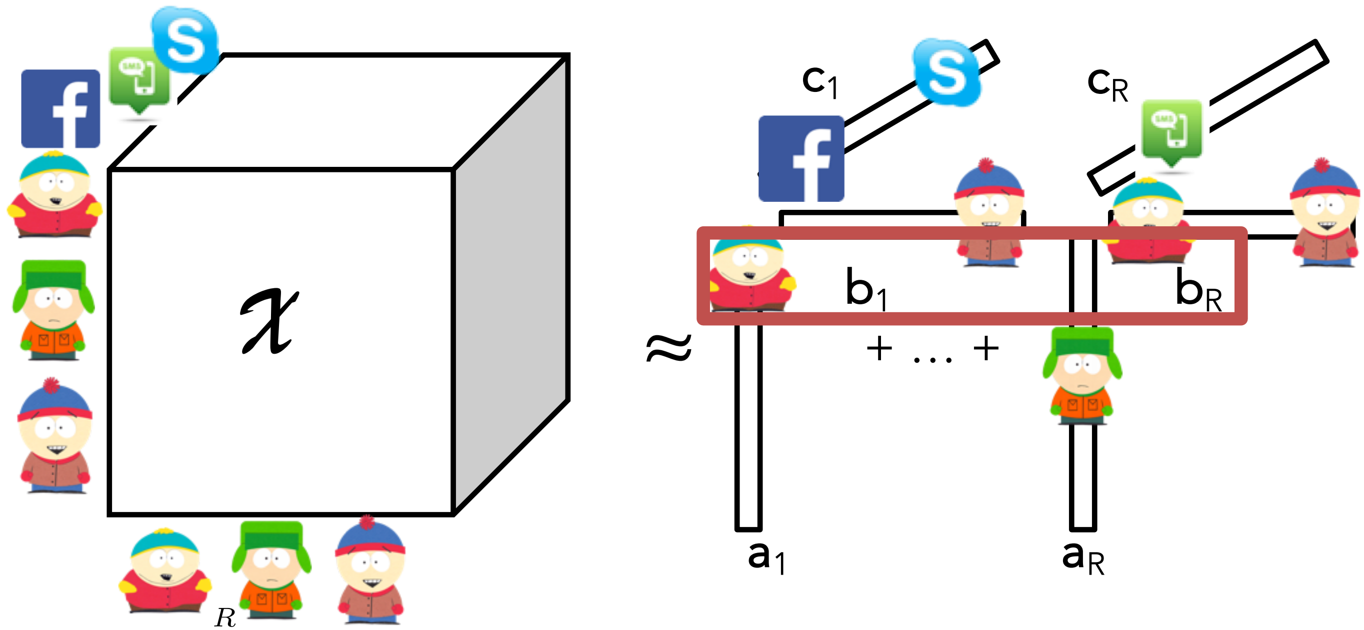
# CP/PARAFAC Decomposition




$$\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{L}} + \boldsymbol{\mathcal{E}}$$

$$\boldsymbol{\mathcal{L}} = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r = [\mathbf{A}, \mathbf{B}, \mathbf{C}]$$

# CP/PARAFAC Decomposition

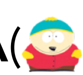
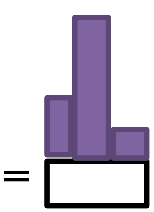



$$\mathcal{X} = \mathcal{L} + \mathcal{E} \qquad \mathcal{L} = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r = [\mathbf{A}, \mathbf{B}, \mathbf{C}]$$

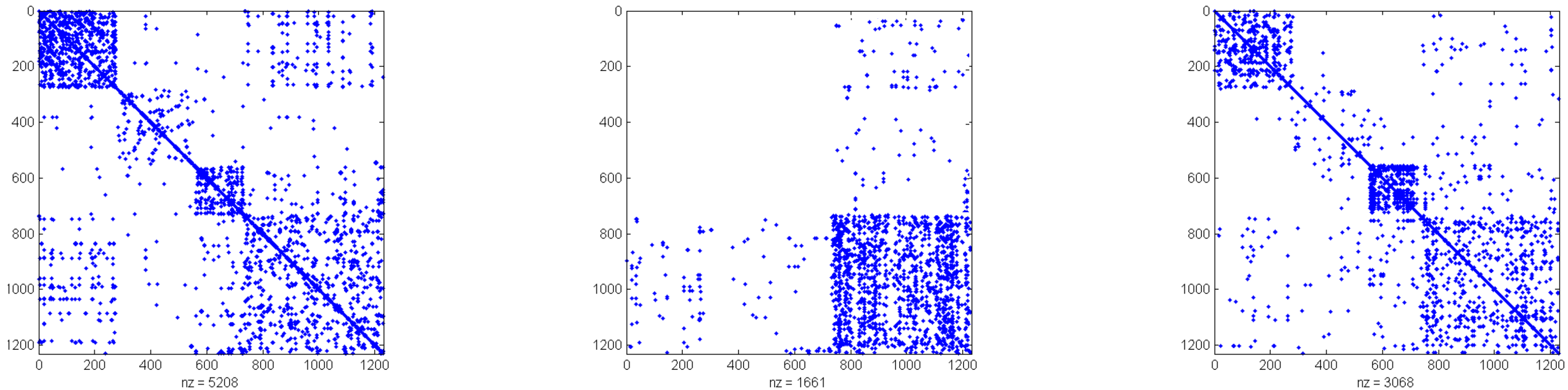
## Intuitive interpretation

1) Each triplet of vectors co-clusters:
(users, users, means of communication)

2) Rows of matrix **A** (or **B**) matrix are ***embeddings*** to "community space"

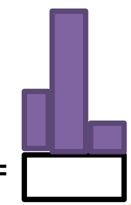
**A**(,:)= 

# DBLP Multi-View Graph




(a) citation (b) co-auth. (c) co-term

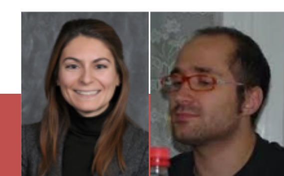
- Take the arg max of the r-dim embedding per node for community assignment
- Baselines
  - Spectral clustering on sum of matrices / views
  - Linked Matrix Factorization
  - [Tang et al. ICDM 2009]
- Outperforms "2D"/matrix baselines wrt NMI (Normalized Mutual Information)



**A**( ,:)=

**Community**( ) =
arg max **A**( ,:)

Info on **B** is *usually* same

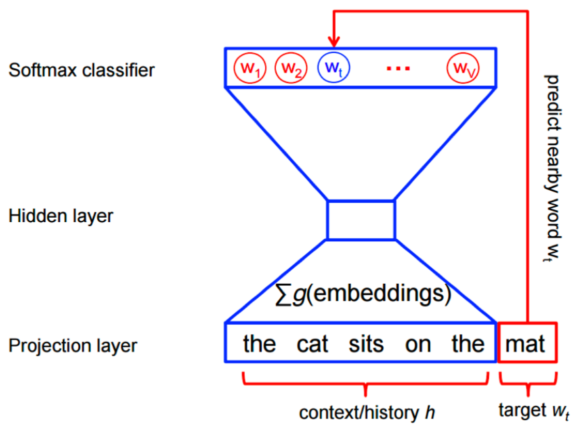


[Papalexakis, Akoglu, Ienco Fusion 2013]

DETOUR

# Word2Vec: Word Embeddings [1]

**Task**: *Given* a target word *predict* which words are more likely context
**Fortuitous by-product**: The learned NN weights provide a vector representation for each word aka **word embedding**



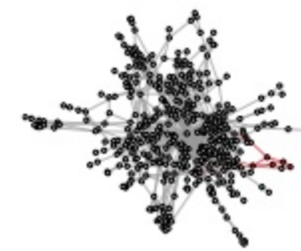

img: https://www.tensorflow.org/versions/r0.11/tutorials/word2vec/index.html

**Cool fact**: Instead of a skipgram, we can factorize a matrix that holds Pointwise Mutual Information [2]

- Embedding space respects contextual relationships
- Can add and subtract the vector embeddings for different words
- Some interesting results:
  - King – Man + Woman = Queen
  - Human – Animal = Ethics
  - Library – Books = Hall
  - President – Power = Prime Minister

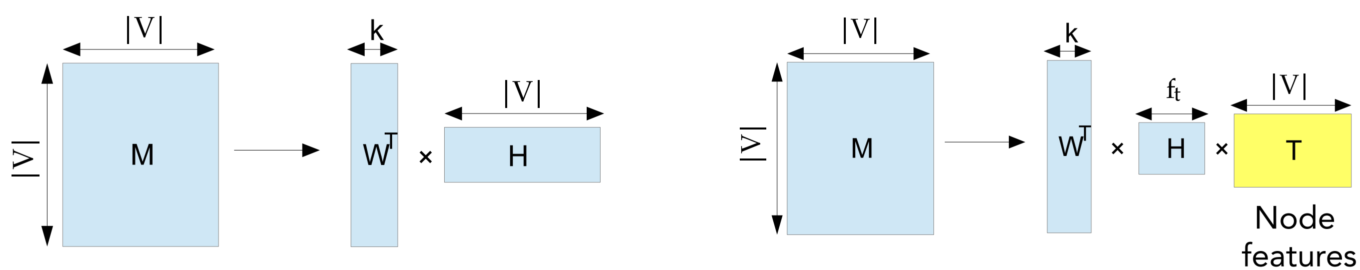

[1] Mikolov et al. "Distributed Representations of Words and Phrases and their Compositionality", NeurIPS'13
[2] Levy et al. "Neural Word Embeddings as Implicit Matrix Factorization", NeurIPS 2014

# Node Embeddings



(a) Random walk generation.

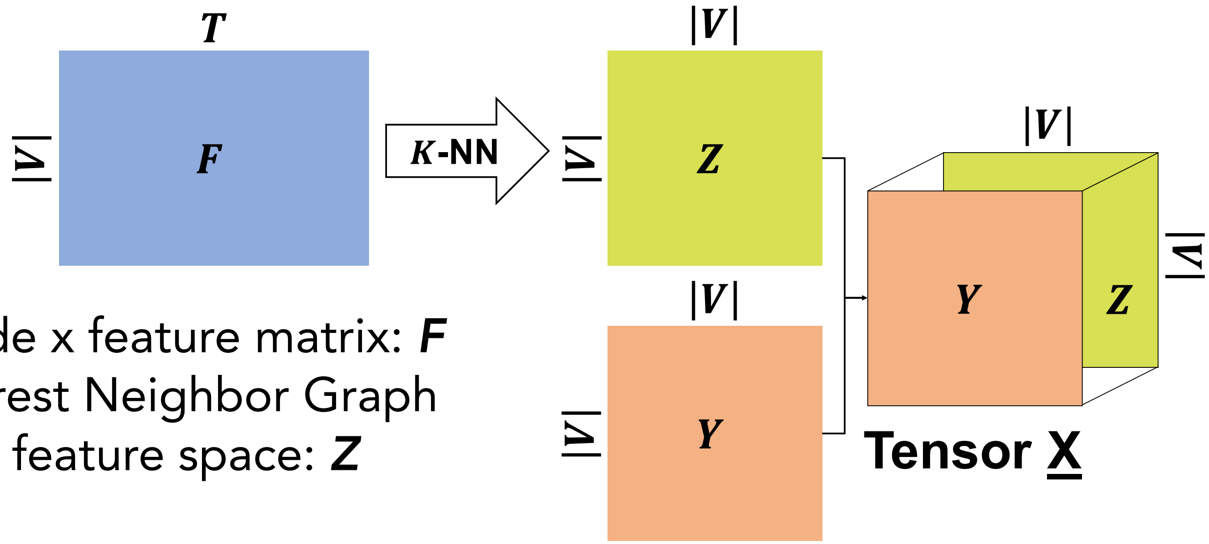
- Inspired by Word Embeddings
- Identifies the *context* by random walks
- Uses skipgram to learn node representation
  - Variants: DeepWalk [KDD15], node2vec [KDD16]
- What if we also have node features?




(a) DeepWalk (b) TADW

Yang et al. Network Representation Learning with Rich Text Information, IJCAI'15

# Tensor-based Context-Aware Node Embeddings

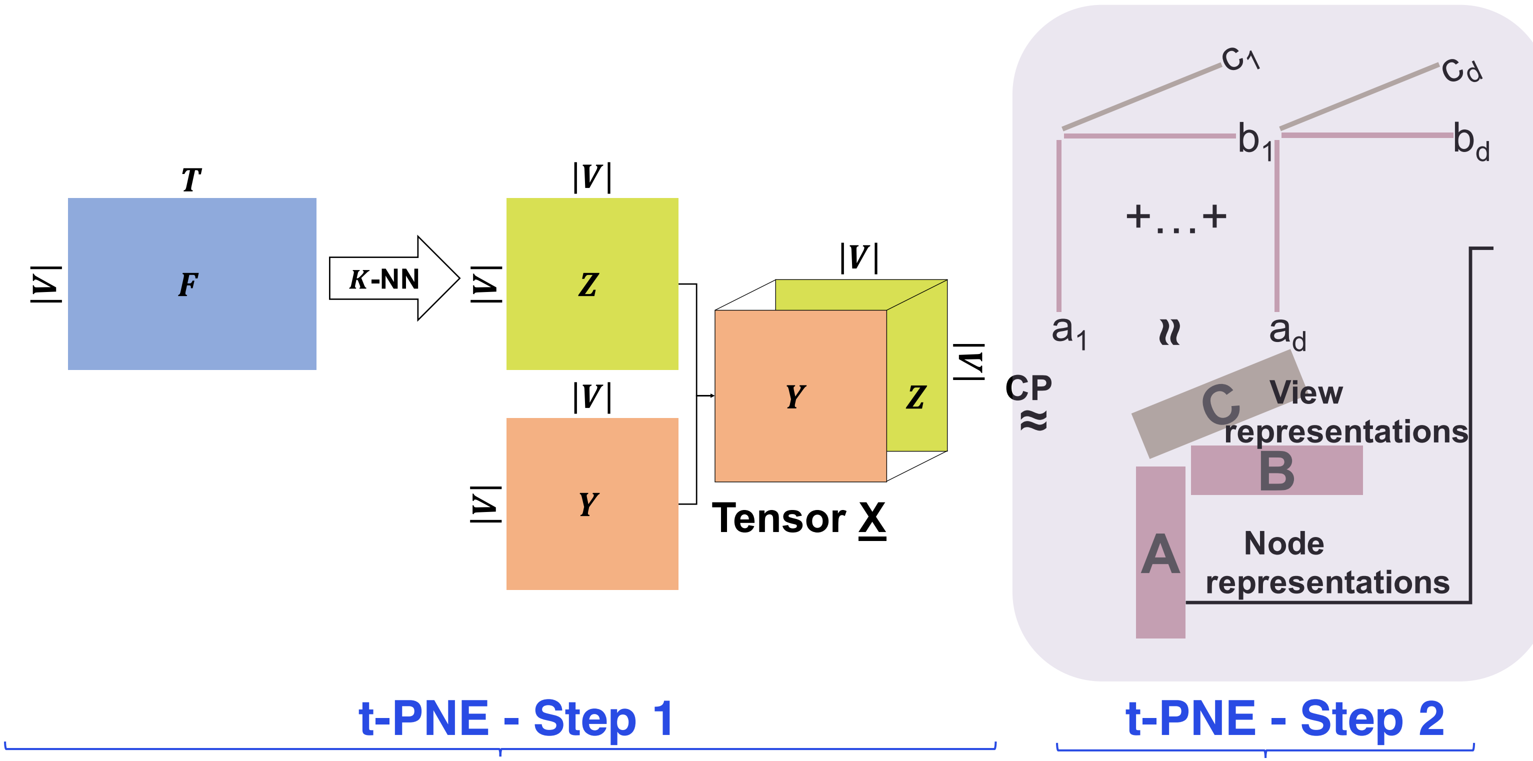



Start from a node x feature matrix: ***F***
Create a K-Nearest Neighbor Graph
for nodes in the feature space: ***Z***

**t-PNE - Step 1**

ASONAM 2018 w/ Saba Al Sayouri, Ekta Gujral, Danai Koutra, Sarah Lam

# Tensor-based Context-Aware Node Embeddings




ASONAM 2018 w/ Saba Al Sayouri, Ekta Gujral, Danai Koutra, Sarah Lam

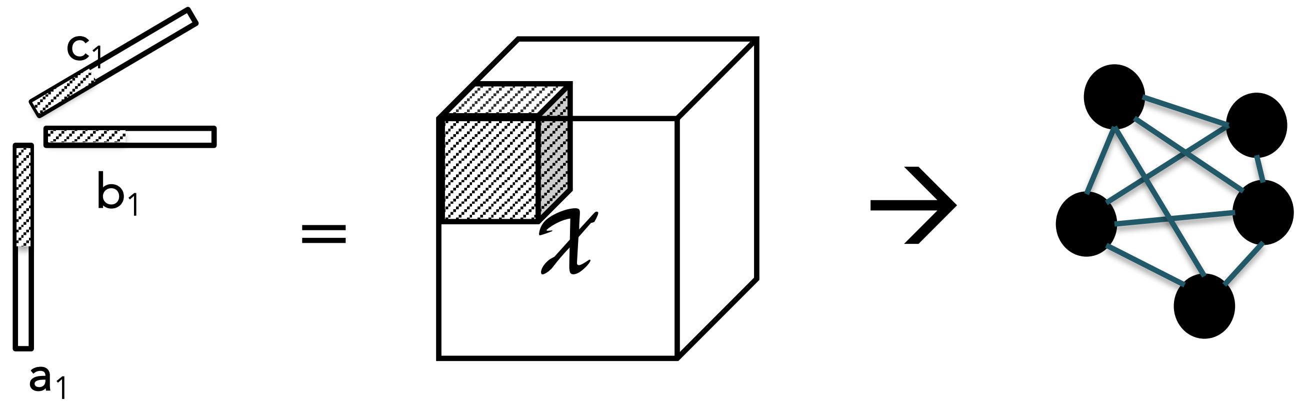
# Tensor-based Context-Aware Node Embeddings

| Algorithm | Wikipedia ($K = 8$) | | | WebKB ($K = 40$) | | | CiteSeer ($K = 15$) | | |
|---|---|---|---|---|---|---|---|---|---|
| | 10% | 50% | 90% | 10% | 50% | 90% | 10% | 50% | 90% |
| DeepWalk | 59.04 | **68.25** | 69.75 | 42.82 | 45.49 | 45.57 | 54.22 | 61.91 | 0.62.11 |
| node2vec | 58.73 | 66.98 | 70.12 | 43.20 | 44.87 | 44.43 | 52.66 | 60.22 | 60.87 |
| Walklets | 58.17 | 65.61 | 66.68 | 42.16 | 46.83 | 49.09 | 52.57 | 59.25 | 60.96 |
| TADW | 19.25 | 32.69 | 46.27 | 48.10 | 49.25 | 48.98 | 25.52 | 56.51 | 67.92 |
| **t-PNE** * | **61.64** | 66.16 | **74.00** | **73.53** | **82.95** | **85.73** | **66.00** | **70.00** | **75.00** |
| Gain over DeepWalk | 4.4 | – | 6.1 | 71.7 | 82.4 | 88.1 | 21.7 | 13.1 | 20.8 |
| Gain over node2vec | 4.9 | – | 5.5 | 70.2 | 84.9 | 92.9 | 25.3 | 16.2 | 23.2 |
| Gain over Walklets | 6.0 | 0.8 | 11.0 | 74.4 | 77.1 | 74.6 | 25.5 | 18.1 | 23.0 |
| Gain over TADW | 220.2 | 102.4 | 59.9 | 52.9 | 68.4 | 75.0 | 158.6 | 23.9 | 10.4 |

Outperforms "traditional" node embeddings

ASONAM 2018 w/ Saba Al Sayouri, Ekta Gujral, Danai Koutra, Sarah Lam

# Rank-1 can be restrictive




Can only "see" full clique

# What if we have richer structure?

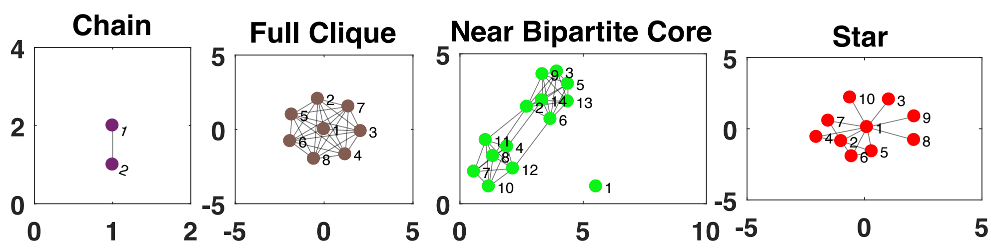
## Knowledge Graph:

Relations of the sort:

*<Trump, is-president, USA>*
*<Merkel, is-chancellor, Germany>*
*<Mitsotakis, is-primeminister, Greece>*

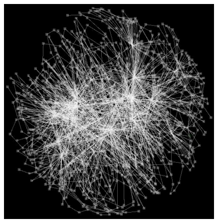
## Social Graph:

*Social media "influencer"*
*Telemarketer/spammer*
*Near cliques/bipartite cores*
…

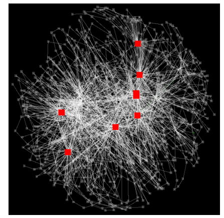



The Web Conference (WWW) 2020 w/ Ekta Gujral

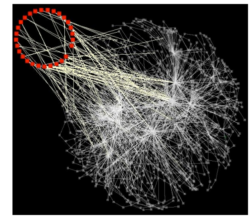
# Related work



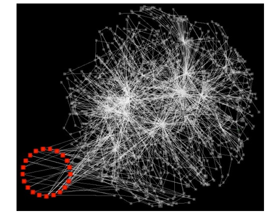
(a) Original Wikipedia `Controversy` graph (with 'spring embedded' layout [15]). No structure stands out.



(b) VoG: 8 out of the 10 most informative structures are stars (their centers in red - Wikipedia editors, heavy contributors etc.).



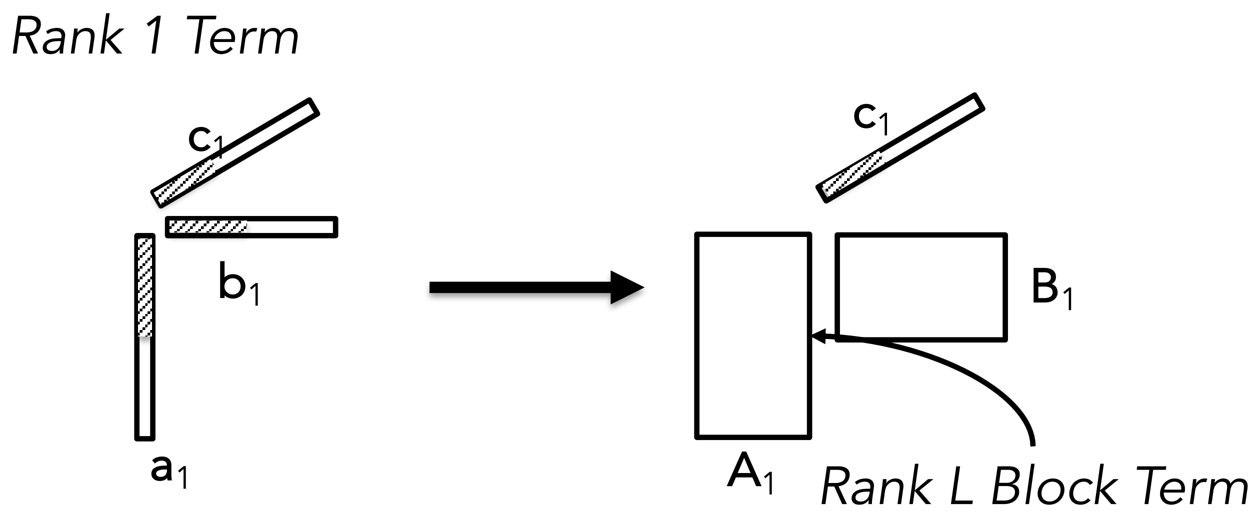
(c) VoG: The most informative bipartite graph - 'edit war' - warring factions (one of them, in the top-left red circle), changing each-other's edits.



(d) VoG: the second most informative bipartite graph - another 'edit war', between vandals (bottom left circle of red points) vs responsible editors (in white).

- Koutra et al. "VoG: Summarizing and Understanding Large Graphs", SDM'14
- Uses a vocabulary of graph structures and tries to compress the graph by using it
- Follow-up work by Shah et al. "TimeCrunch", KDD'15 stitches graph snapshots over time
- Can we automatically extract that rich structure?

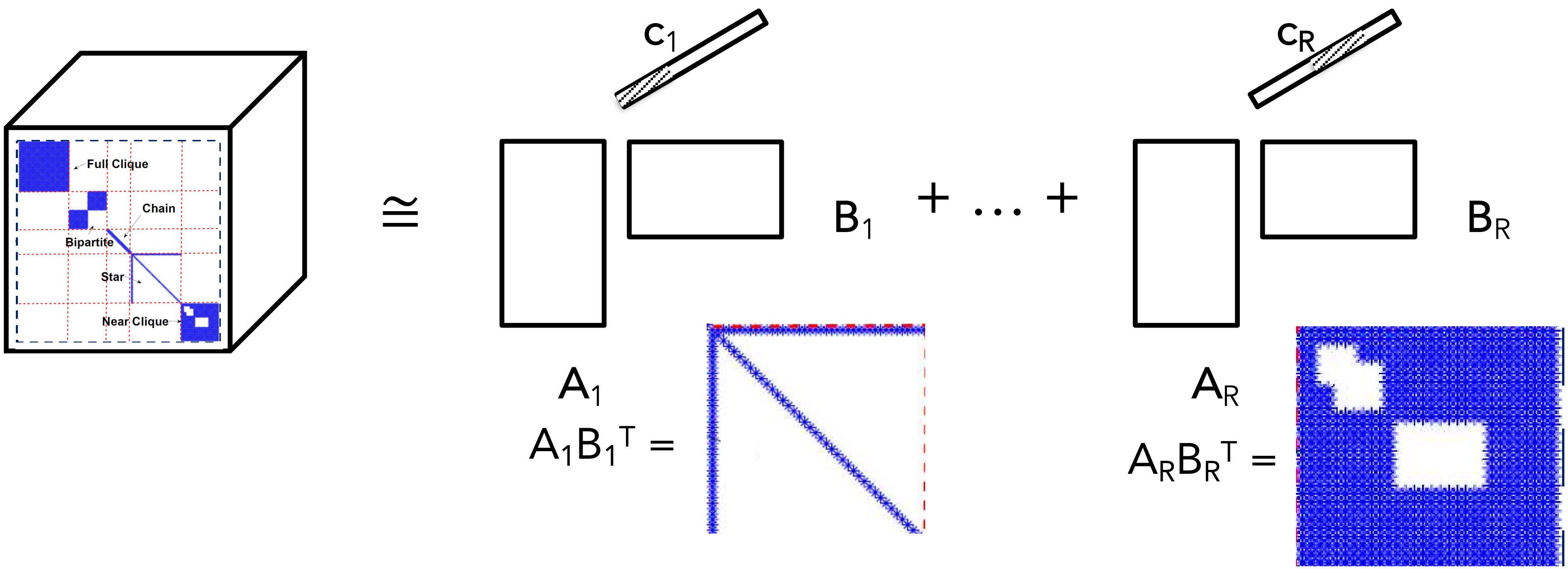
# We need higher-rank blocks!




Can express richer structure
Still has the nice interpretability of CP/PARAFAC

The Web Conference (WWW) 2020 w/ Ekta Gujral

# Block-Term Decomposition

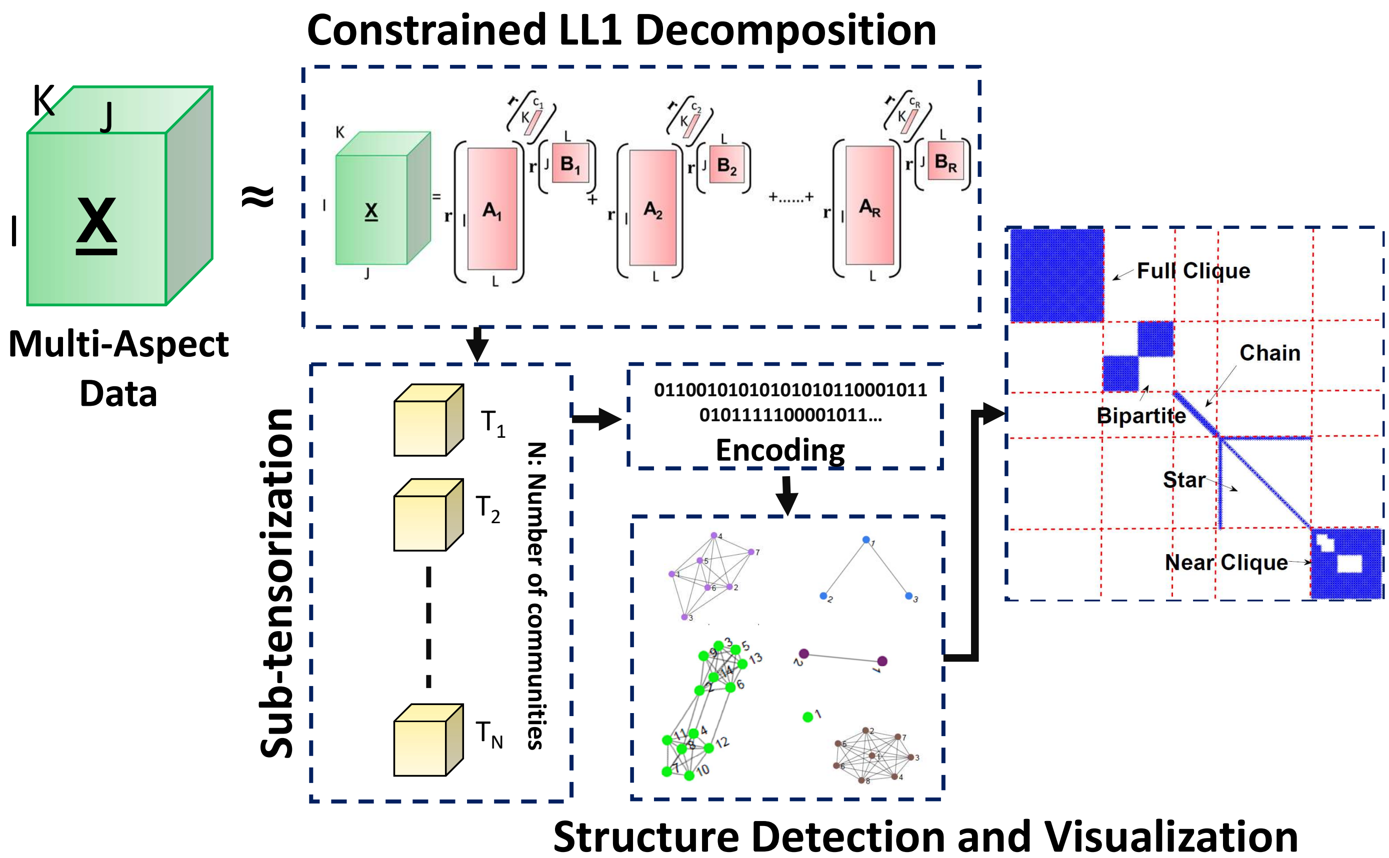



Star

Near Clique

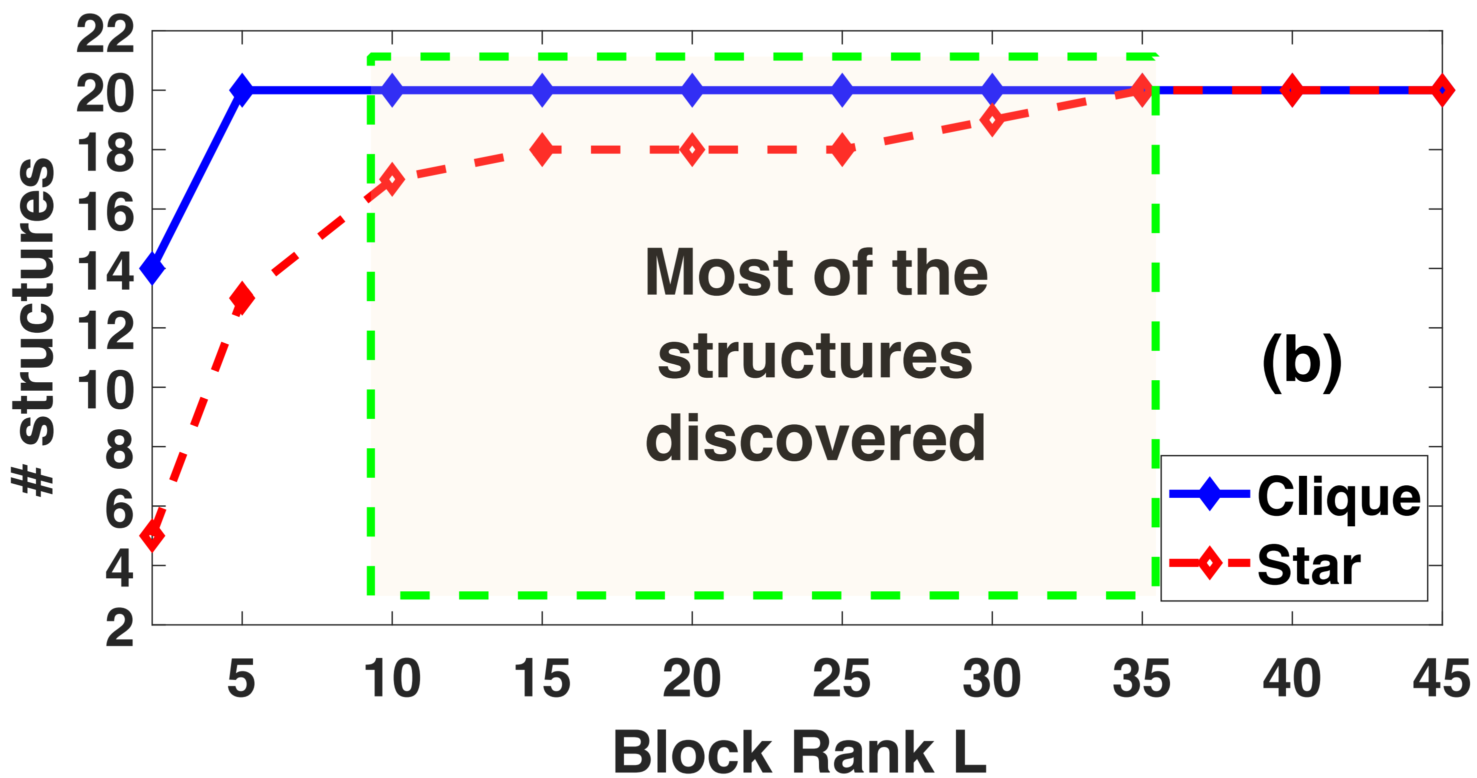
The Web Conference (WWW) 2020 w/ Ekta Gujral

# Beyond Rank-1: Discovering Rich Structure in Multi-Aspect Graphs




The Web Conference (WWW) 2020 w/ Ekta Gujral

# Beyond Rank-1: Discovering Rich Structure in Multi-Aspect Graphs




The Web Conference (WWW) 2020 w/ Ekta Gujral

# Roadmap

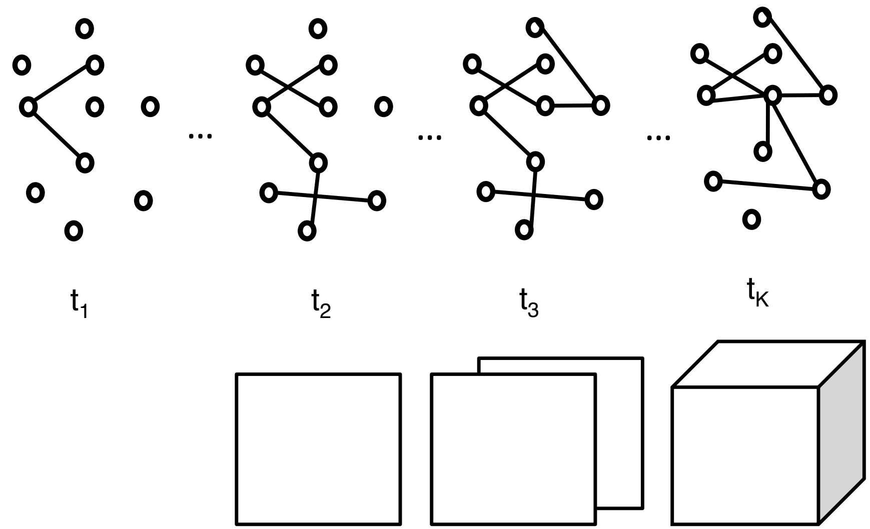
Introduction to Tensors & Graph Mining

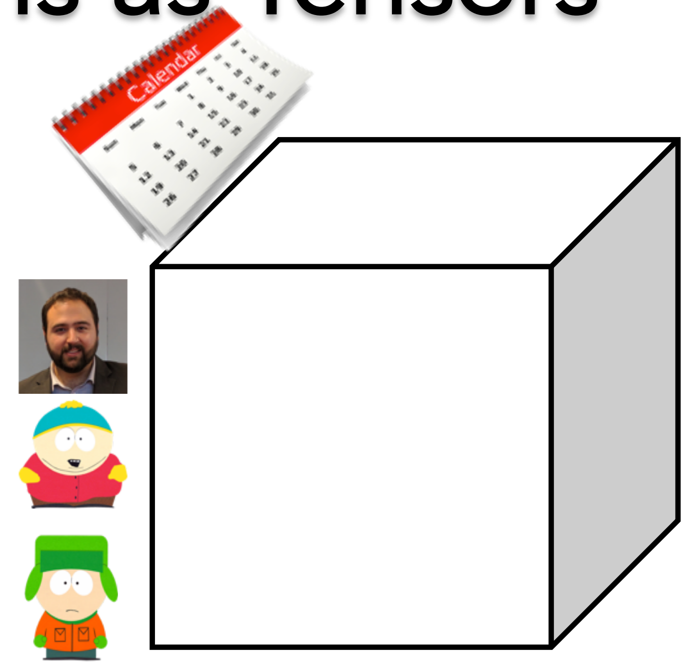
Time-evolving Tensors & Concept Drift

Adversarial Machine Learning

Conclusion

# Time-Evolving Graphs as Tensors



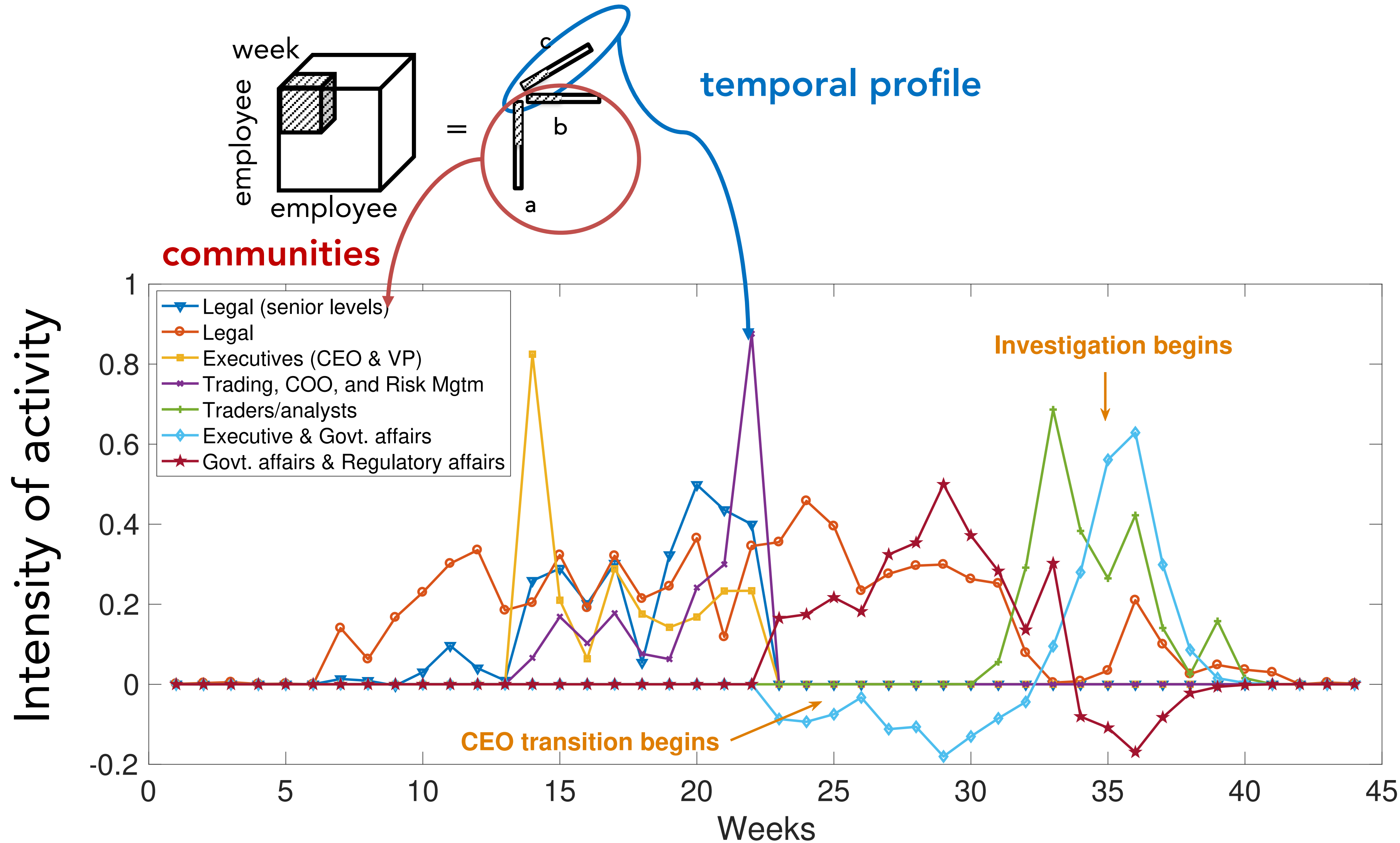

Detect anomalies / real-life events


Papalexakis, Evangelos E., Nicholas D. Sidiropoulos, and Rasmus Bro.
"From k-means to higher-way co-clustering: Multilinear decomposition with sparse latent factors."
*IEEE transactions on signal processing* 61, no. 2 (2012): 493-506.

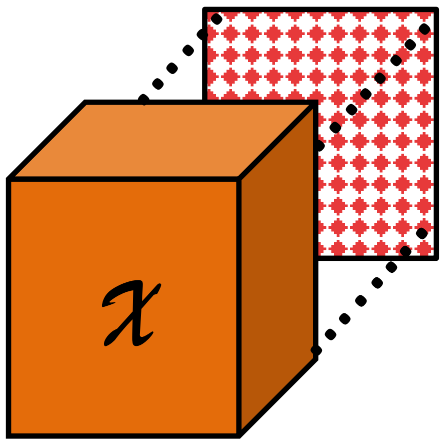
Gorovits, Alexander, Ekta Gujral, Evangelos E. Papalexakis, and Petko Bogdanov.
"LARC: Learning activity-regularized overlapping communities across time." KDD 2018

Shen, Yanning, Brian Baingana, and Georgios B. Giannakis.
"Tensor decompositions for identifying directed graph topologies and tracking dynamic networks."
*IEEE Transactions on Signal Processing 2017*

# Time-evolving communities



# Incremental Decomposition of Streaming Tensors




- Tensor updated in streaming fashion
- New slices arrive
  - New snapshots on a temporal graph
  - In general, new slices (or batches) over time

How can we ***incrementally*** update the decomposition?

- Nion & Sidiropoulos, Adaptive Algorithms to Track the PARAFAC Decomposition of a Third-Order Tensor, IEEE TSP 2009
- Mardani, Morteza, Gonzalo Mateos, and Georgios B. Giannakis. "Subspace learning and imputation for streaming big data matrices and tensors." IEEE Transactions on Signal Processing, 2015
- Baskaran et al, Accelerated Low-Rank Updates to Tensor Decompositions, IEEE HPEC 2016
- Zhou et al, Accelerating Online CP Decompositions for Higher Order Tensors, ACM KDD 2016
- Sun et al., Beyond Streams and Graphs: Dynamic Tensor Analysis, ACM KDD 2006

# A Tale of Two Sketches

SamBaTen: Sampling-based Batch Incremental Tensor Decomposition


Ekta Gujral, UC Riverside, egujr001@ucr.edu
Ravdeep Pasricha, UC Riverside, rpasr001@ucr.edu
Evangelos E. Papalexakis, UC Riverside, epapalex@cs.ucr.edu



**Abstract**

Tensor decompositions are invaluable tools in analyzing multimodal datasets. In many real-world scenarios, such datasets are far from being static, to the contrary they tend to grow over time. For instance, in an online social network setting, as we observe new interactions over time, our dataset gets updated in its "time" mode. How can we maintain a valid and accurate tensor decomposition of such a dynamically evolving multimodal dataset, without having to re-compute the entire decomposition after every single update? In this paper we introduce SamBaTen, a *Sam*pling-based *Ba*tch Incremental *Ten*sor Decomposition algorithm, which incrementally maintains the decomposition given new updates to the tensor dataset. SamBaTen is able to scale to datasets that the state-of-the-art in incremental tensor decomposition is unable to operate on, due to its ability to effectively summarize the existing tensor and the incoming updates, and perform all computations in the reduced summary space. We extensively evaluate SamBaTen using synthetic and real datasets. Indicatively, SamBaTen achieves comparable accuracy to state-of-the-art incremental and non-incremental techniques, while being up to *25-30 times faster*. Furthermore, SamBaTen scales to very large sparse and dense dynamically evolving tensors of dimensions up to $100K \times 100K \times 100K$ where state-of-the-art incremental approaches were not able to operate.

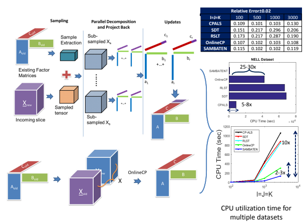




**Figure 1: SamBaTen outperforms state-of-the-art baselines while maintaining competitive accuracy.**

In a wide array of modern real-world applications, data are far from being static. To the contrary, data get updated dynamically. For instance, in an online social network, new interactions occur every second and new friendships are formed at a similar pace. In the tensor realm, we may view a large proportion of these dynamic updates as an introduction of new "slices" in the tensor: in the social network example, new interactions that happen as time evolves imply the introduction of new snapshots of the network, which grow the tensor in the "time" mode. A tensor decomposition in that tensor can discover *communities* and their evolution over

**OCTen: ONLINE COMPRESSION-BASED TENSOR DECOMPOSITION**


*Ekta Gujral, Ravdeep Pasricha, Tianxiong Yang, Evangelos E. Papalexakis*

Department of Computer Science, UC Riverside, California, USA
Email: (egujr001, rpasr001, tyang022)@ucr.edu, epapalex@cs.ucr.edu



**ABSTRACT**

Tensor decompositions are powerful tools for large data analytics, as they jointly model multiple aspects of data into one framework and enable the discovery of the latent structures and higher-order correlations within the data. One of the most widely studied and used decompositions, especially in data mining and machine learning, is the Canonical Polyadic or PARAFAC decomposition. However, today's datasets are not static and often grow and change over time. To operate on such large dynamic data, we present OCTen, the first ever compression-based online parallel implementation for the CP/PARAFAC decomposition. We conduct an extensive empirical analysis of the algorithms in terms of fitness, memory used and CPU time and in order to demonstrate the compression and scalability of the method, we apply OCTen to big tensor data. Indicatively, OCTen performs on-par or better than state-of-the-art online and offline methods in terms of decomposition accuracy and efficiency, while achieving memory savings ranging in *40-200%*.

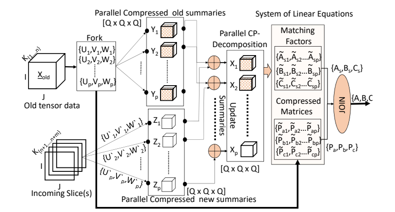




**Fig. 1: Framework. Compressed tensor summaries $\underline{\mathbf{Y}}_p$ and $\underline{\mathbf{Z}}_p$ are obtained by applying randomly generated compression matrices $(\mathbf{U}_p, \mathbf{V}_p, \mathbf{W}_p)$ and $(\mathbf{U}'_p, \mathbf{V}'_p, \mathbf{W}'_p)$ to $\underline{\mathbf{X}}_{old}$ and $\underline{\mathbf{X}}_{new}$ respectively. The updated summaries are computed by $\underline{\mathbf{X}}_p = \underline{\mathbf{Y}}_p + \underline{\mathbf{Z}}_p$. Each $\underline{\mathbf{X}}_p$ is independently decomposed in parallel. The update step anchors all compression and factor matrices to a single reference i.e. $(\mathbf{P}_a, \mathbf{P}_b, \mathbf{P}_c)$ and $(\mathbf{A}_s, \mathbf{B}_s, \mathbf{C}_s)$, and solves a linear equation for the overall A, B, and C.**

tensors. In this paper, we fill that gap. Our contributions are summarized as follows:

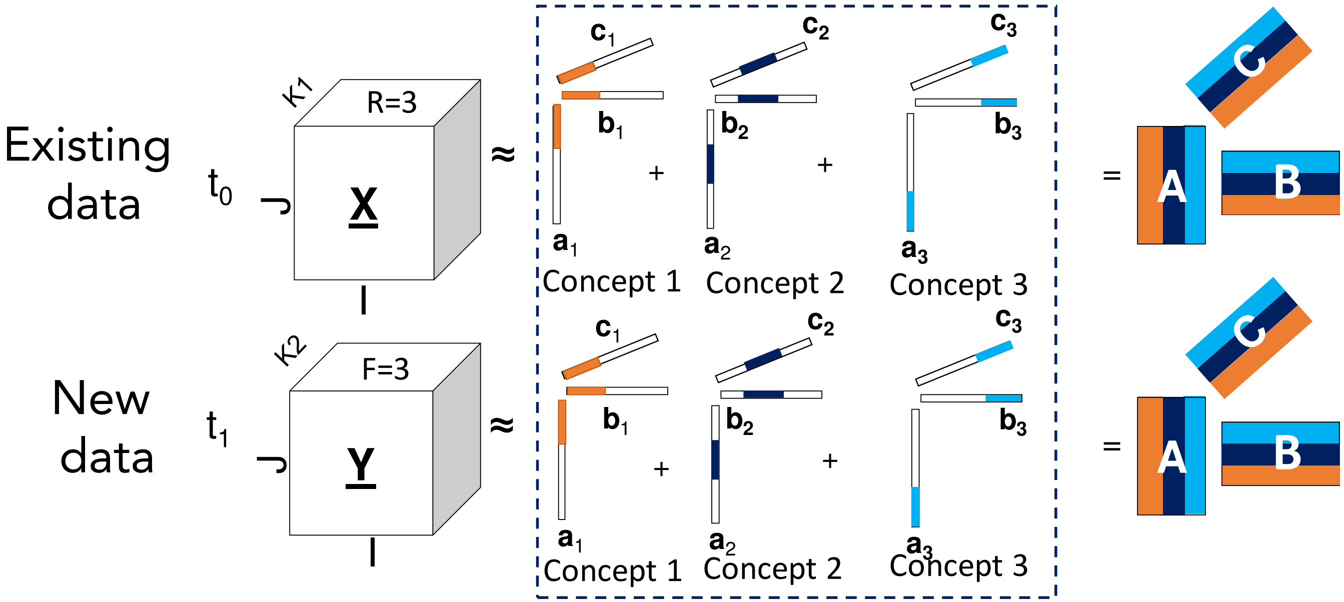
- **Novel Parallel Algorithm** We introduce OCTen, a

Gujral et al, SIAM SDM 2018
Randomized index sampling

Gujral et al, IEEE CAMSAP 2019
Randomized compression

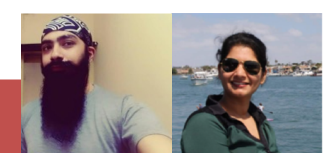
# Central limiting assumption

- Most streaming work **(incl. our work ☺ )** assumes:

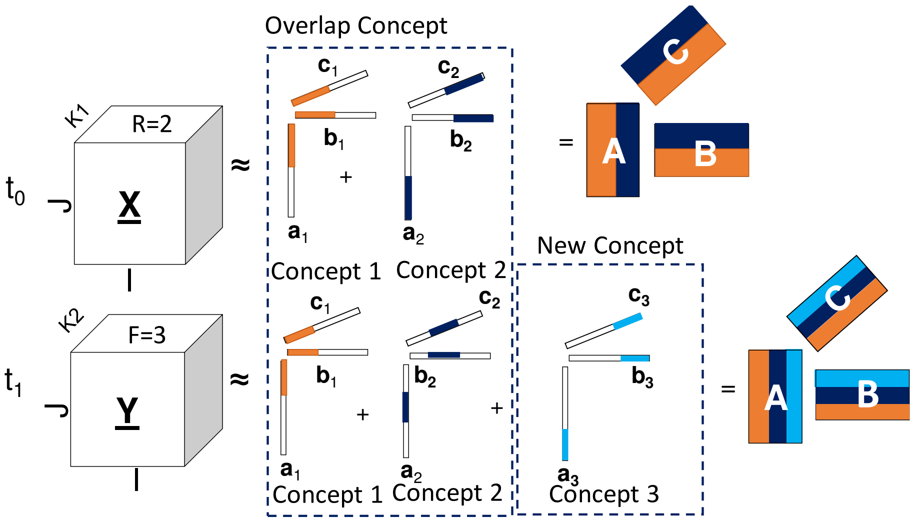



$$\boxed{\text{Rank(existing data)} = \text{Rank(new data)}}$$

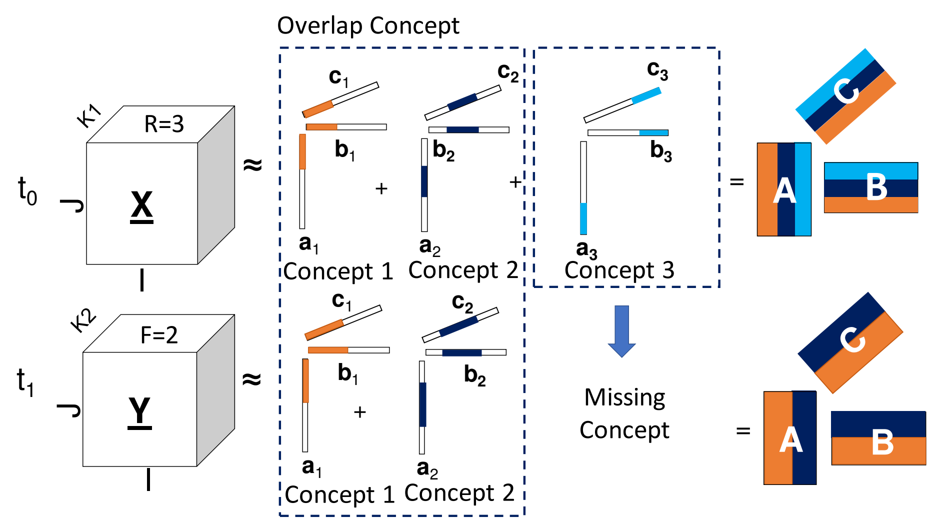
ECML-PKDD 2018 w/ Ravdeep Pasricha &Ekta Gujral



# What if this doesn't hold?



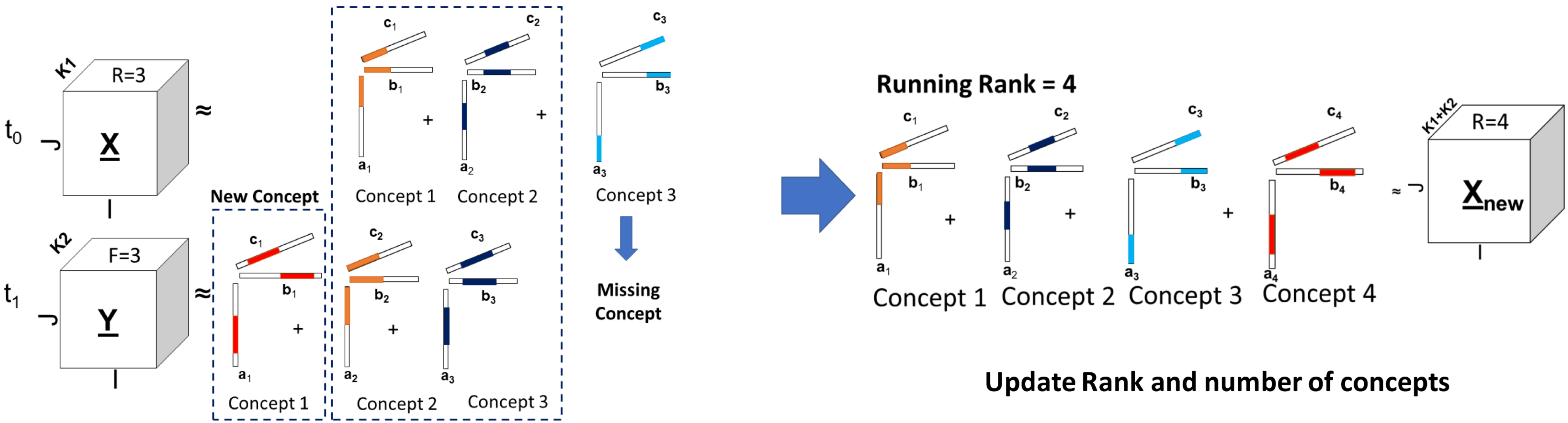

New concept appears




Concept is missing

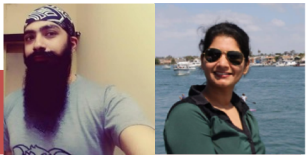
ECML-PKDD 2018 w/ Ravdeep Pasricha &Ekta Gujral

# Identifying and Alleviating Concept Drift in Streaming Tensor Decomposition




- Algorithm for detecting and alleviating concept drift: SeekAndDestroy

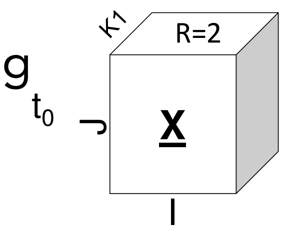
ECML-PKDD 2018 w/ Ravdeep Pasricha &Ekta Gujral



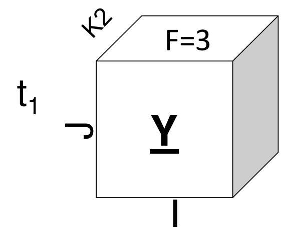
# SeekAndDestroy in a nutshell

- At every step we have

Existing data

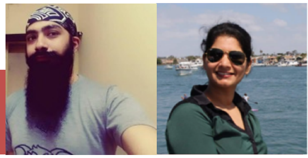



Incoming batch

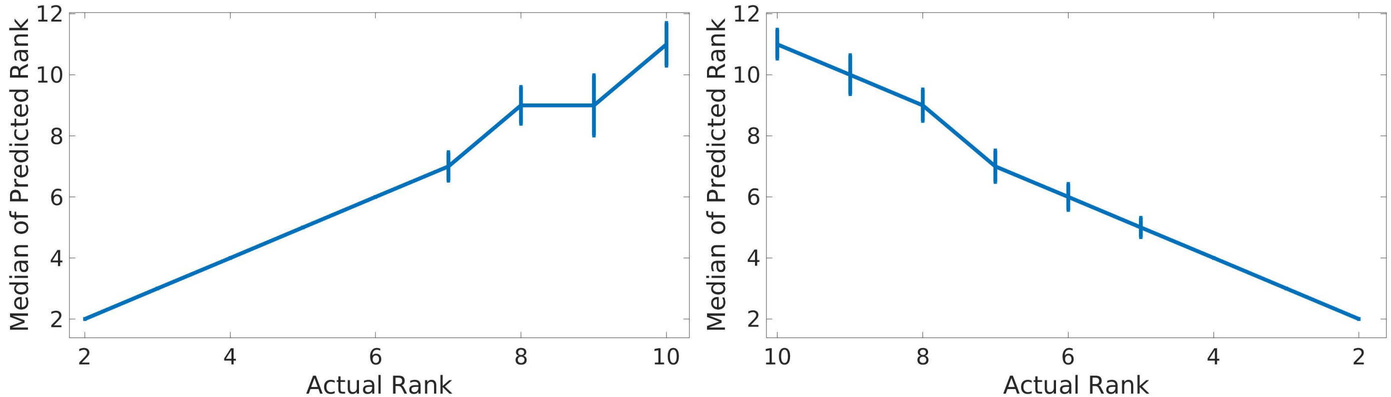



- To determine drift:
  - ✧ We compute rank(Y) and compare with rank(X)
  - ✧ Even if rank(Y) = rank(X), we may have new components
  - ✧ We compute matching of components
  - ✧ If similarity>threshold, same component
  - ✧ Else this is a new component

ECML-PKDD 2018 w/ Ravdeep Pasricha &Ekta Gujral



# Detection of Concept Drift




(a) Increasing rank (b) Decreasing rank

**Fig. 4:** *SeekAndDestroy* is able to successfully detect concept drift, which is manifested as changes in the rank throughout the stream.

| DataSet | Dimension | Initial Rank | Full Rank |
|---|---|---|---|
| SDS1<br>SDS2 | 100 x 100 x 100 | 2 | 5<br>10 |
| SDS3<br>SDS4 | 300 x 300 x 300 | 2 | 5<br>10 |
| SDS5<br>SDS6 | 500 x 500 x 500 | 2 | 5<br>10 |

*Synthetic data with simulated drift*

ECML-PKDD 2018 w/ Ravdeep Pasricha &Ekta Gujral

# Concept Drift Effects in Reconstruction

*Reconstruction error*

| DataSet | OnlineCP (Initial Rank) | OnlineCP (Full Rank) | SamBaTen (Initial Rank) | SamBaTen (Full Rank) | *SeekAndDestroy* |
|---|---|---|---|---|---|
| SDS1 | 0.2782±0.0221 | 0.197±0.086 | **0.261±0.048** | 0.317±0.058 | 0.283±0.075 |
| SDS2 | 0.2537±0.0125 | 0.168±0.507 | **0.244±0.028** | 0.480±0.051 | 0.253±0.0412 |
| SDS3 | 0.2731±0.0207 | 0.205±0.164 | 0.385±0.021 | 0.445±0.164 | **0.266±0.081** |
| SDS4 | 0.245±0.013 | 0.171±0.537 | 0.299±0.045 | 0.402±0.049 | **0.221±0.0423** |
| SDS5 | 0.2719±0.0198 | 0.206±0.022 | 0.559±0.046 | 0.519±0.0219 | **0.256±0.105** |
| SDS6 | 0.238±0.013 | 0.171±0.374 | 0.510±0.036 | 0.547±0.0276 | **0.208±0.0433** |

| DataSet | Dimension | Initial Rank | Full Rank |
|---|---|---|---|
| SDS1<br>SDS2 | 100 x 100 x 100 | 2 | 5<br>10 |
| SDS3<br>SDS4 | 300 x 300 x 300 | 2 | 5<br>10 |
| SDS5<br>SDS6 | 500 x 500 x 500 | 2 | 5<br>10 |

*Synthetic data with simulated drift*

**If final/full rank is unknown**:
SeekAndDestroy can detect drift and have lower error than SOTA

**If final rank is known (*unrealistic*)**:
SOTA performs on par or better

ECML-PKDD 2018 w/ Ravdeep Pasricha &Ekta Gujral

# Evidence of drift in real data

| Running Rank | Predicted Full Rank | Batch Size | Approximation Error | | |
|---|---|---|---|---|---|
| | | | *SeekAndDestroy* | SambaTen | OnlineCP |
| $7 \pm 0.88$ | $4 \pm 0.57$ | 22 | $\mathbf{0.68 \pm 0.002}$ | $0.759 \pm 0.059$ | $0.941 \pm 0.001$ |

When streaming Enron, we encounter a number of drifting communities that other methods miss

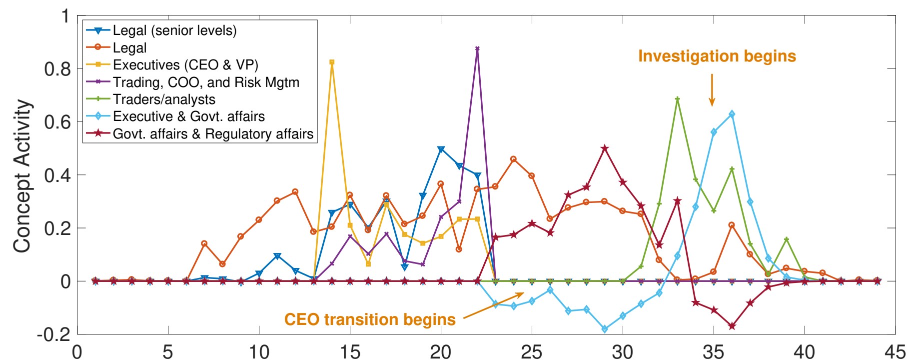


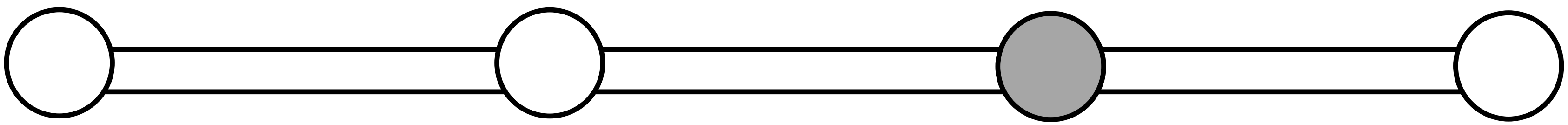

**Fig. 5:** Timeline of concepts discovered in Enron.

ECML-PKDD 2018 w/ Ravdeep Pasricha &Ekta Gujral

# Roadmap

Time-evolving Tensors
& Concept Drift

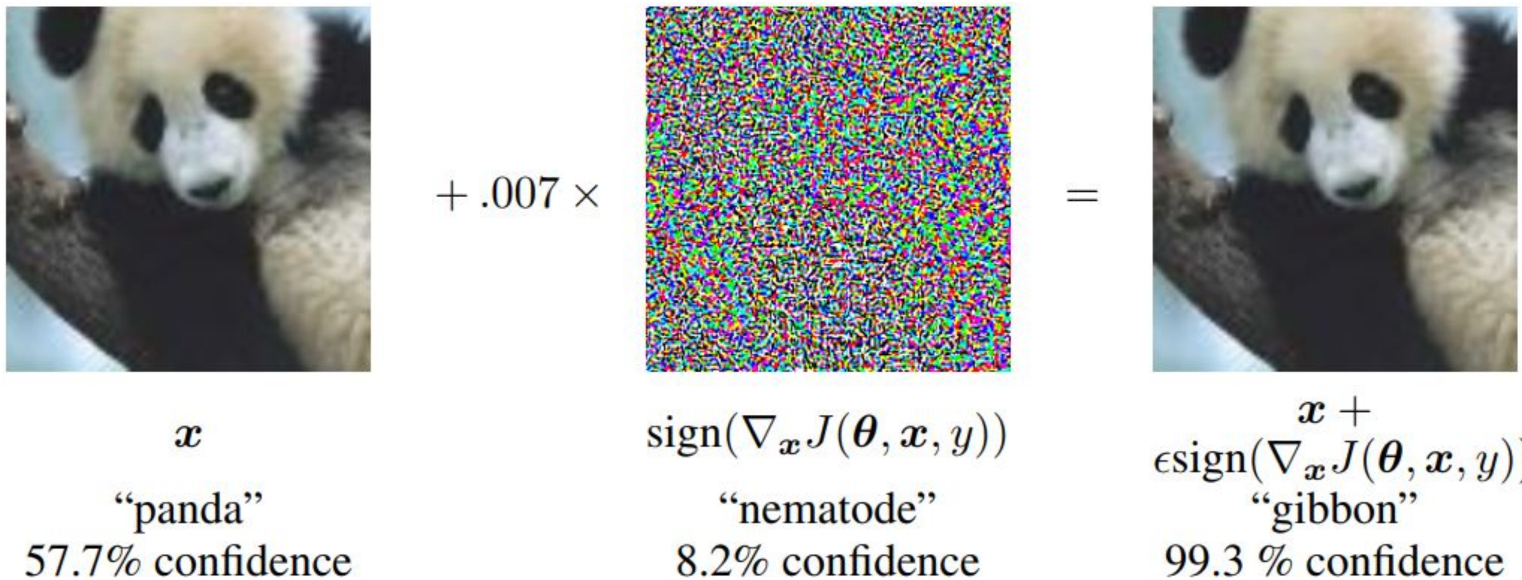
Conclusion



Introduction to
Tensors
& Graph Mining

**Adversarial
Machine
Learning**

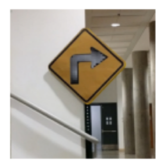
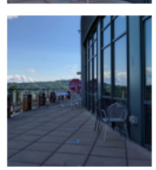
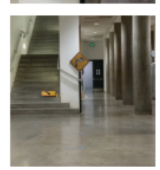
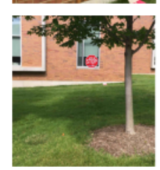
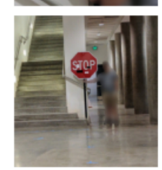
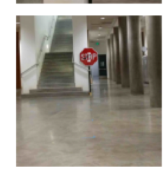
# What’s happening here??




https://arxiv.org/pdf/1412.6572.pdf : *fast gradient sign method*

# What's happening here??

Table 1: Sample of physical adversarial examples against LISA-CNN and GTSRB-CNN.

| Distance/Angle | Subtle Poster | Subtle Poster Right Turn | Camouflage Graffiti | Camouflage Art (LISA-CNN) | Camouflage Art (GTSRB-CNN) |
|---|---|---|---|---|---|
| 5′ 0° |  |  |  |  |  |
| 5′ 15° |  |  |  |  |  |
| 10′ 0° |  |  |  |  |  |
| 10′ 30° |  |  |  |  |  |
| 40′ 0° |  |  |  |  |  |
| Targeted-Attack Success | 100% | 73.33% | 66.67% | 100% | 80% |

https://arxiv.org/pdf/1707.08945.pdf

# Not just an "academic curiosity"

MIT Technology Review

Computing / Cybersecurity

## Hackers can trick a Tesla into accelerating by 50 miles per hour

A two inch piece of tape fooled the Tesla's cameras and made the car quickly and mistakenly speed up.

by **Patrick Howell O'Neill** February 19, 2020

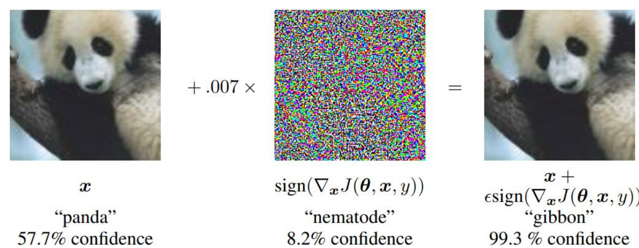
https://www.technologyreview.com/2020/02/19/868188/hackers-can-trick-a-tesla-into-accelerating-by-50-miles-per-hour/

# "Defense" Problem Definition

For a given model $C$:

- $x$: clean instance, $x'$: perturbed instance
- Goal of adversarial attack:

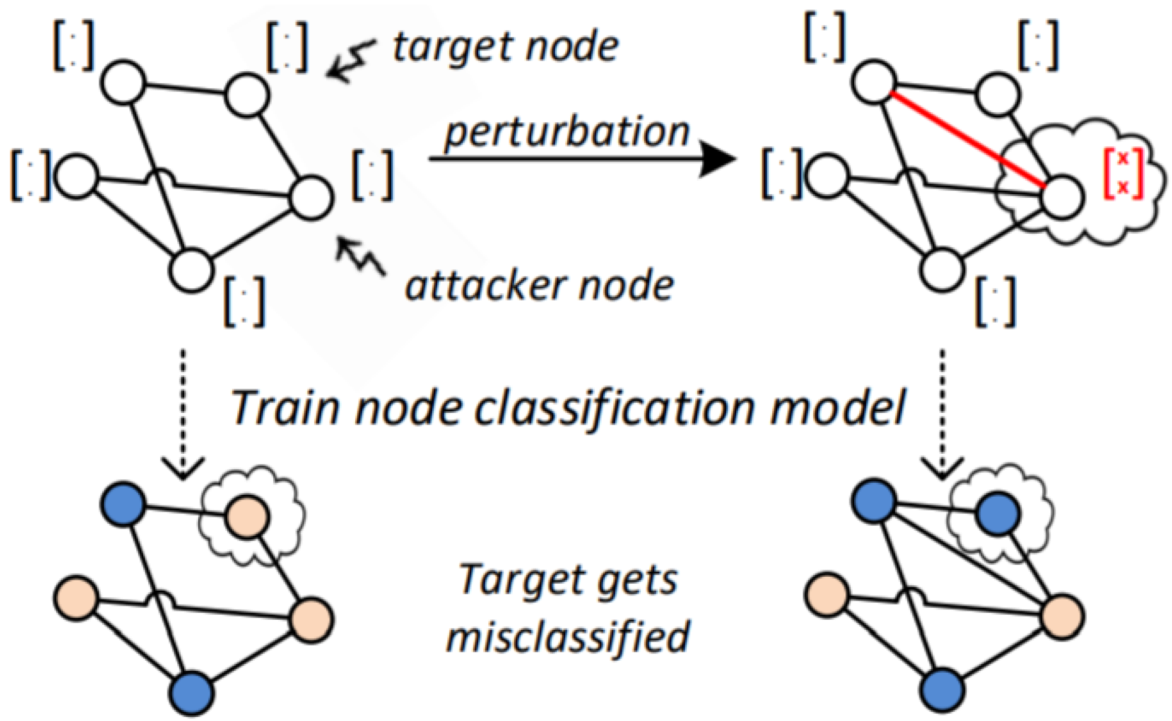
$$\mathrm{x}' = \mathrm{x} + \delta \Rightarrow C(x) \neq C(x') \text{ while } \|x - x'\| < \tau$$




- Goal of defense mechanism:
  apply a preprocessing operation $g(.)$ that brings back $x'$ closer to the clean instance $x$ such that:

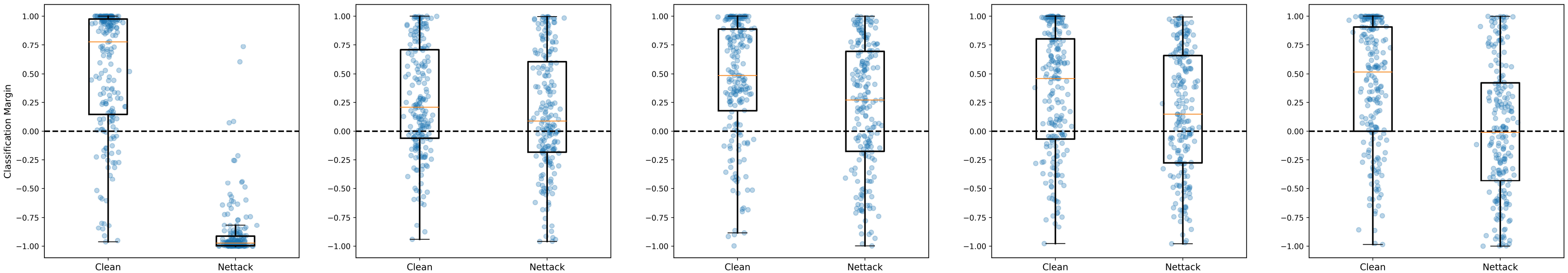
$$C\big(g(x')\big) = C(x)$$

# Attack on Graph Convolutional Networks



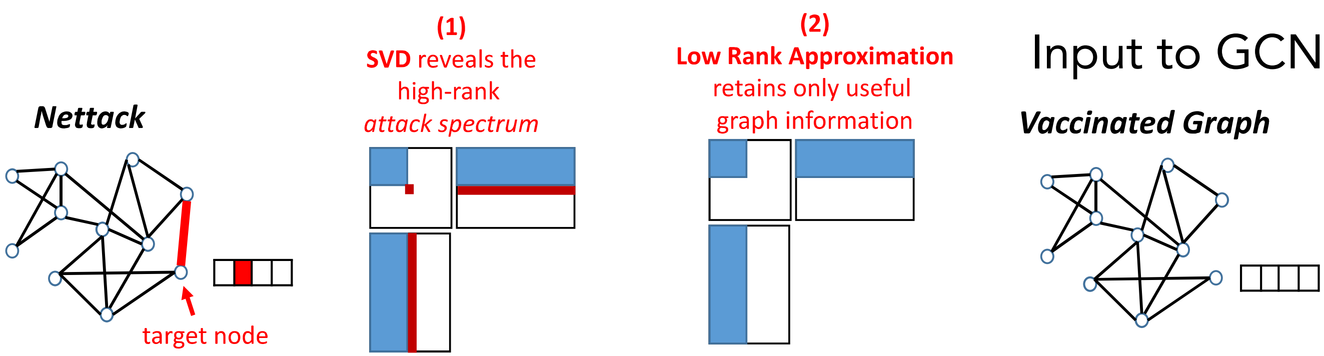

Nettack [Zugner et al. KDD18]

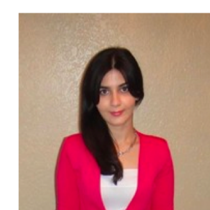
# All you need is low (rank)




(a) Unvaccinated (b) rank-5 approx. (c) rank-10 approx. (d) rank-15 approx. (e) rank-50 approx.

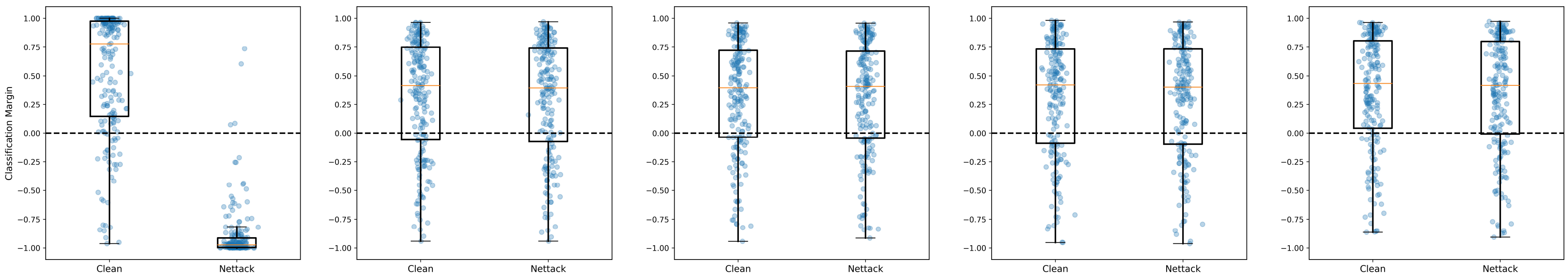
**Figure 6: Vaccinating GCN against NETTACK**



WSDM 2020 w/ Negin Entezari

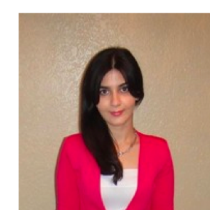
# All you need is low (rank)




(a) GCN (b) t-PINE ($K = 30$) (c) t-PINE ($K = 35$) (d) t-PINE ($K = 40$) (e) t-PINE ($K = 45$)

**Figure 8: Poisoning of t-PINE with NETTACK on CiteSeer. The embedding dimension is 32.**

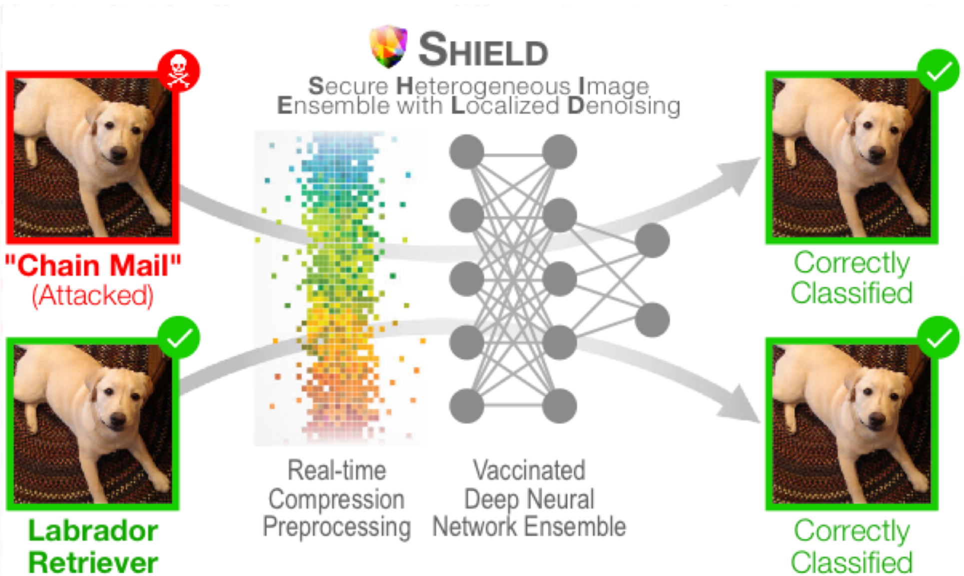
| | Method | CiteSeer | Cora-ML | PoliticalBlogs |
|---|---|---|---|---|
| GCN | Clean | 0.83 | 0.82 | 0.90 |
| | NETTACK | 0.02 | 0.01 | 0.06 |
| t-PINE | Clean | 0.74 | 0.68 | 0.87 |
| | NETTACK | 0.72 | 0.64 | 0.30 |



WSDM 2020 w/ Negin Entezari

# Attack in images is of high-frequency!

SHIELD: Fast, Practical Defense and Vaccination for Deep Learning using JPEG Compression

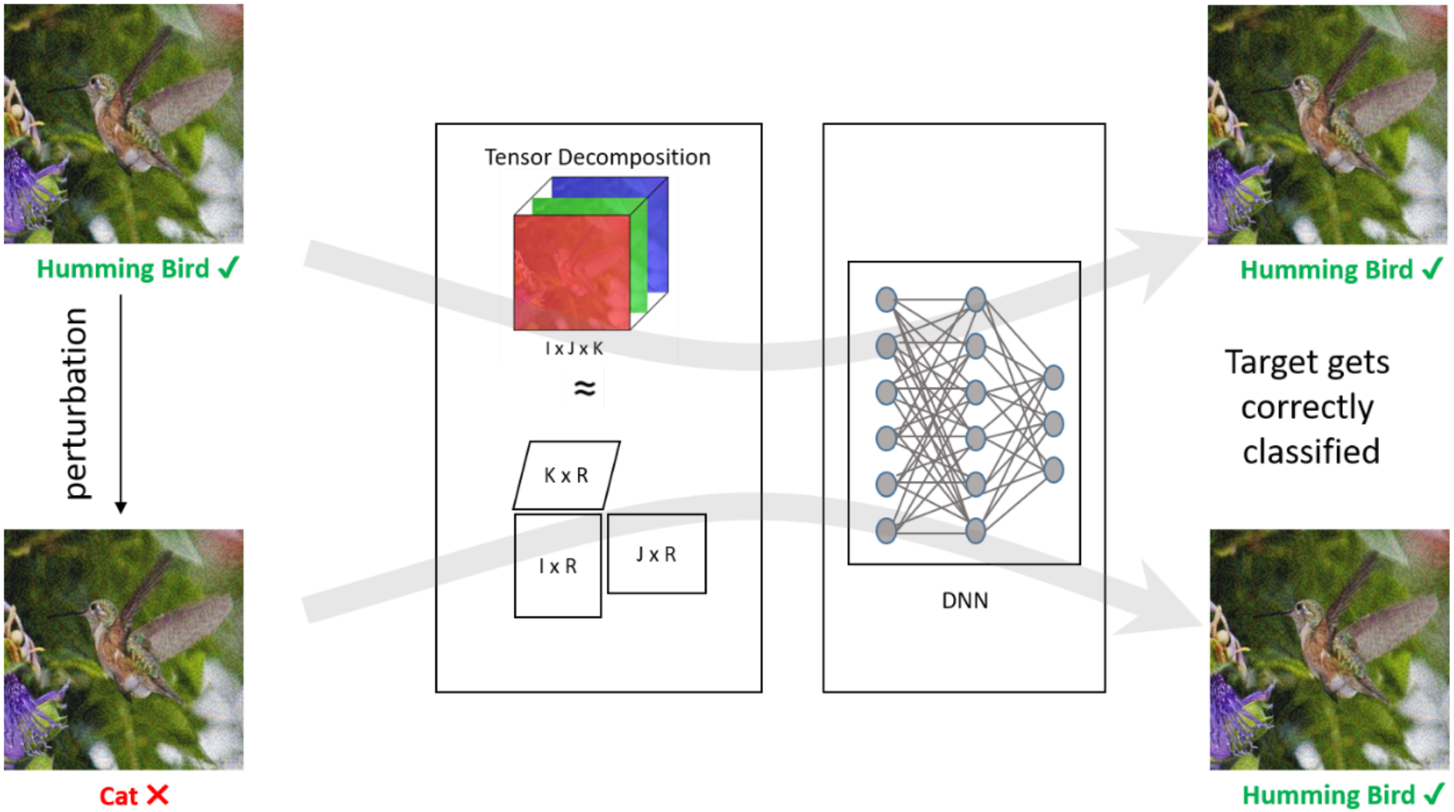


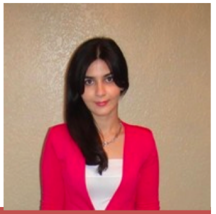


JPEG compression removes adversarial perturbations

https://arxiv.org/pdf/1802.06816.pdf

# Tensor-Based Defense Mechanism

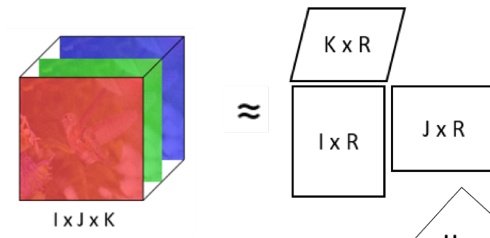


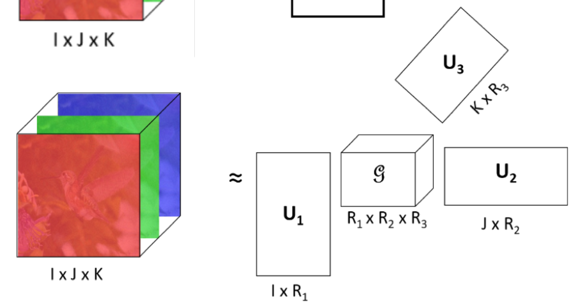
On-going work & arXiv:2002.10252 w/ Negin Entezari

# Choice of Tensor Model

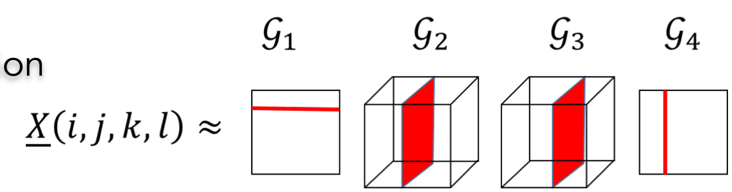
- CP/PARAFAC:
  - Pros: Interpretable latent factors
  - Cons: Slow,
    Restricted to have same ranks for all modes and super-diagonal core makes it not a suitable choice for image decomposition

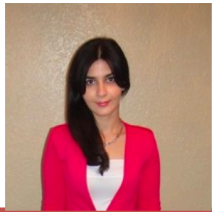



- Tucker:
  - Pros: No constraint on the core tensor
    Each mode can have a different rank
  - Cons: Slow
    latent factors are not easily interpretable




- Tensor-Train:
  - Pros: No constraint on ranks of different modes
    Linearly scalable with respect to tensor dimension
  - Cons: latent factors are not easily interpretable

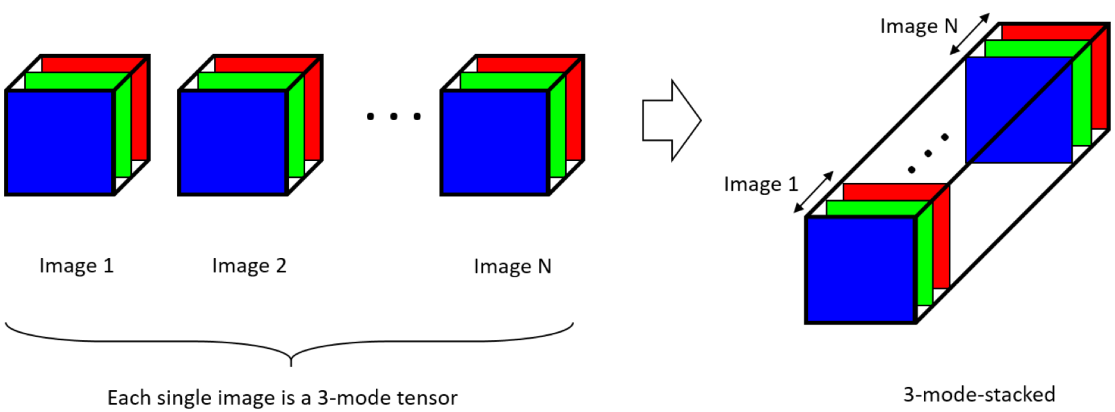


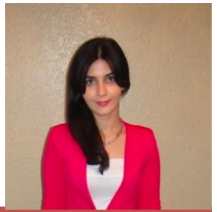



On-going work & arXiv:2002.10252 w/ Negin Entezari

# How to Represent Batch of Images?

- Batching images in either
  - 4-mode tensor
  - Stack all slices into 3-mode tensor
- Through batching we
  - Amortize computational cost
  - Leverage patterns across images

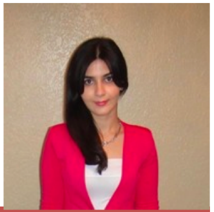


On-going work & arXiv:2002.10252 w/ Negin Entezari

# Experimental Results

| Configurations | PGD ($\epsilon = 4$) | FGSM ($\epsilon = 4$) | i-FGSM ($\epsilon = 4$) | Runtime (seconds) |
|---|---|---|---|---|
| No defense | 11.10 | 18.40 | 7.49 | |
| [Tensor-Train, 4-mode, 5, [5,90,3]] | **51.53** | **43.59** | **50.46** | 675 |
| [Tensor-Train, 4-mode, 10, [10,100,3]] | 51.01 | 43.10 | 49.95 | 605 |
| [Tensor-Train, 3-mode, 1, 40] | 49.75 | 42.32 | 48.52 | **530** |
| [Tucker, 3-mode-stacked, 30, [105,105,90]] | 49.37 | 40.07 | 48.79 | 1050 |
| [Parafac, 3-mode, 1, 60] | 48.11 | 41.38 | 49.75 | 5500 |
| SLQ | 44.60 | 29.40 | 38.60 | **410** |

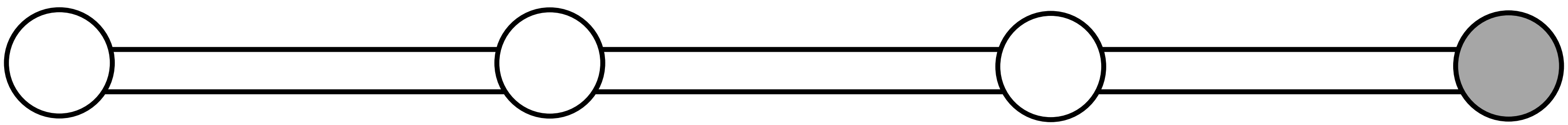


On-going work & arXiv:2002.10252 w/ Negin Entezari

# Roadmap

Time-evolving Tensors
& Concept Drift

Conclusion



Introduction to
Tensors
& Graph Mining

Adversarial
Machine
Learning

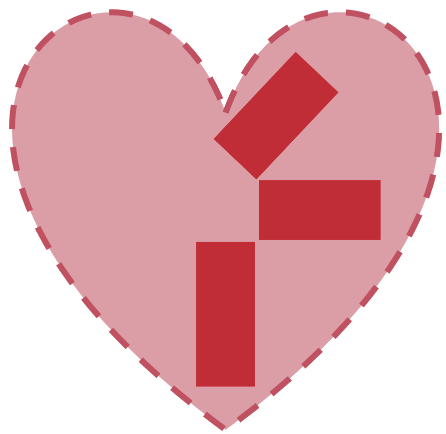
# Tensors Everywhere!

- Unsupervised exploratory analysis
  - ✧ Challenges:
    - Is there structure in the data? What kind?
    - How many useful patterns in the data?
    - Which model should I use?
- Tensors in a Brave New World
  - ✧ Interplay of traditional tensor methods & deep learning
    - E.g., defending against adv. attacks

# Thank you! Questions?

- How to reach me: http://www.cs.ucr.edu/~epapalex/

I



Tensors

Supported by:

NAVSEA NAVAL SEA SYSTEMS COMMAND · NSF · Adobe · GPU Grant NVIDIA · UCR Bourns College of Engineering

M A D Lab @ UCR
Multi Aspect Data