FROM A MILLION-DOLLAR COMPETITION TO DEEP NEURAL NETWORKS

IMPROVING RECOMMENDER SYSTEM PERFORMANCE CASE STUDY

MICHAŁ FILIPIUK, FREDERIC GRABOWSKI

AGENDA

WHAT IS COLLABORATIVE FILTERING?

NETFLIX PRIZE

HOW TO MEASURE MODEL EFFICIENCY?

DATASETS

AUTOENCODERS

HOW TO SPEED IT UP?

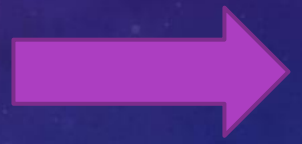
BIBLIOGRAPHY

WHAT IS COLLABORATIVE FILTERING?

COLLABORATIVE FILTERING IS A PROBLEM OF MAKING AUTOMATIC PREDICTIONS ABOUT THE INTERESTS OF A USER BY COLLECTING PREFERENCES OR TASTE INFORMATION FROM MANY USERS.

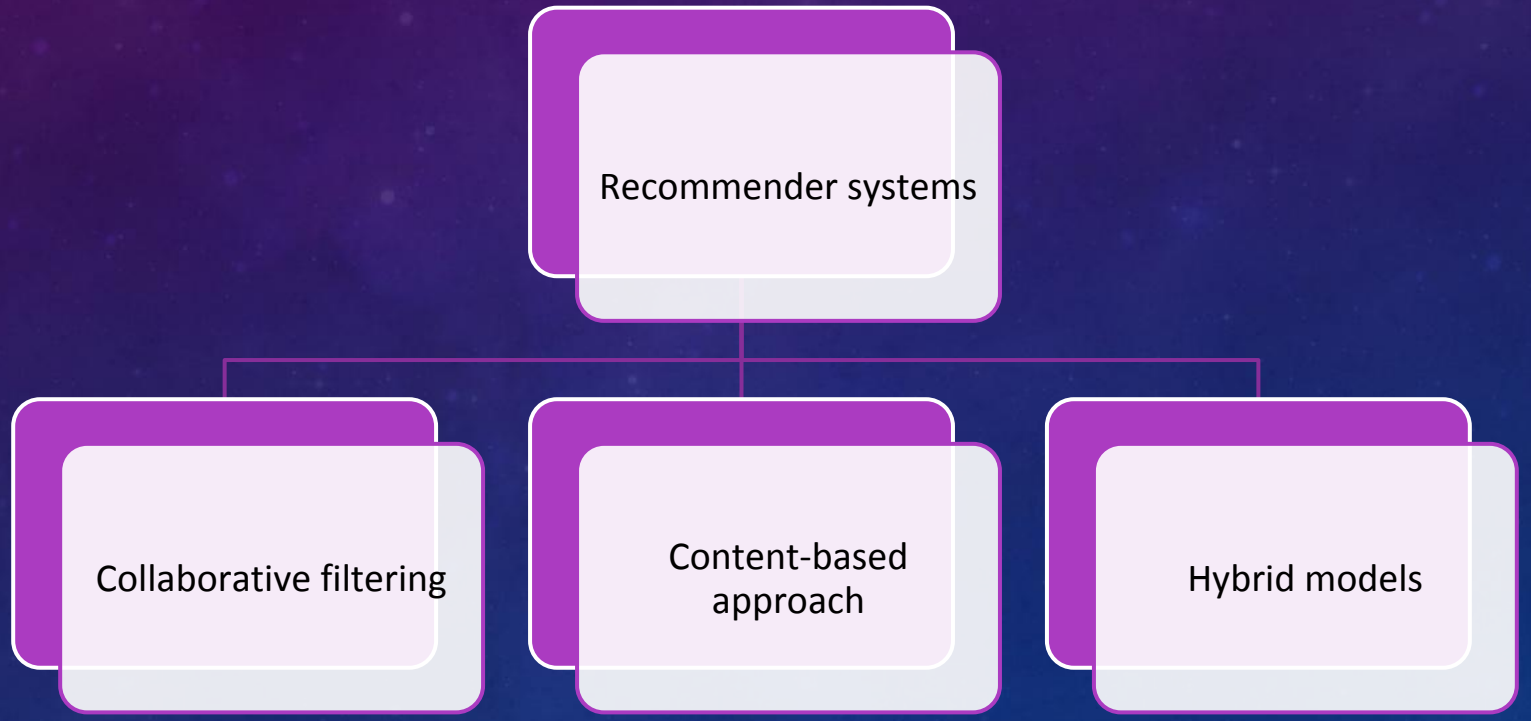
WHAT IS COLLABORATIVE FILTERING?

	P1	P2	P3	P4	P5
U1	3	1	?	3	1
U2	?	2	4	1	3
U3	3	1	?	?	?
U4	4	3	5	?	4



	P1	P2	P3	P4	P5
U1	3	1	1.1	3	1
U2	1.7	2	4	1	3
U3	3	1	1.5	3.4	2.2
U4	4	3	5	4.1	4

WHAT IS COLLABORATIVE FILTERING?



NETFLIX PRIZE

Goal: reduce RSME by 10% comparing to Cinematch, Netflix best model

Timeline:

October 2, 2006 – competition started	October 8, 2006 – Cinematch beaten	September 21, 2009 – BellKor's Pragmatic Chaos team wins the competition

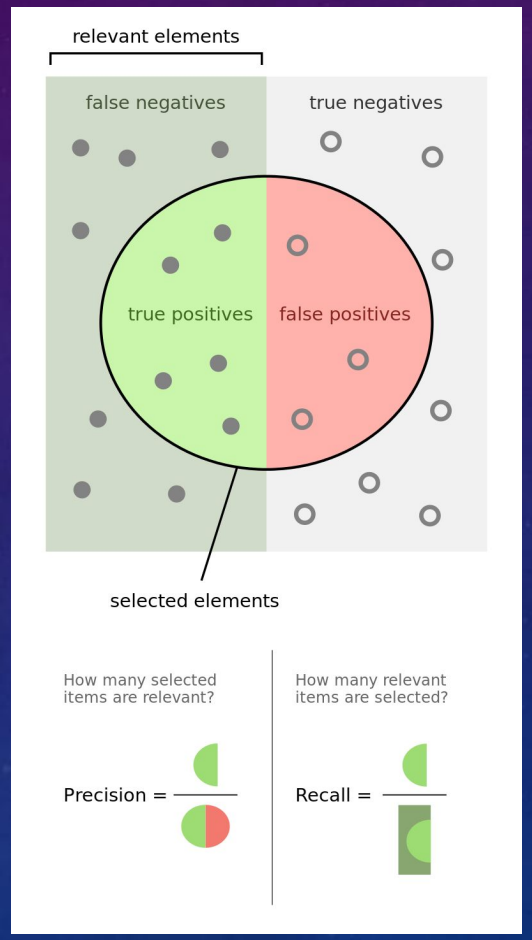
HOW TO MEASURE MODEL EFFICIENCY?

- Recall@K:

$$\text{Recall}(k) = \frac{|\{Relevant\ in\ first\ k\}|}{\min(|\{Relevant\}|, k)}$$

- Average Precision@K:

$$AveP(k) = \frac{\sum_{i=1}^{k} P(k) \cdot rel(k)}{k}$$



- Normalized Discounted Cumulative Gain@K:

$$\text{NDCG}(k) = \frac{\text{DCG}(k)}{\text{IDCG}(k)}$$

$$\text{DCG}(k) = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

$$\text{IDCG}(k) = \sum_{i=1}^{|REL_p|} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

DATASETS

MovieLens-20M

Users' activity from MovieLens, a movie recommendation service run by GroupLens, a research lab at the University of Minnesota. Data was gathered between January 09, 1995 and March 31, 2015.

- 5-star rating with 0.5-star scale
- 20,000,263 ratings
- 27,278 movies
- 138,493 users

movielens

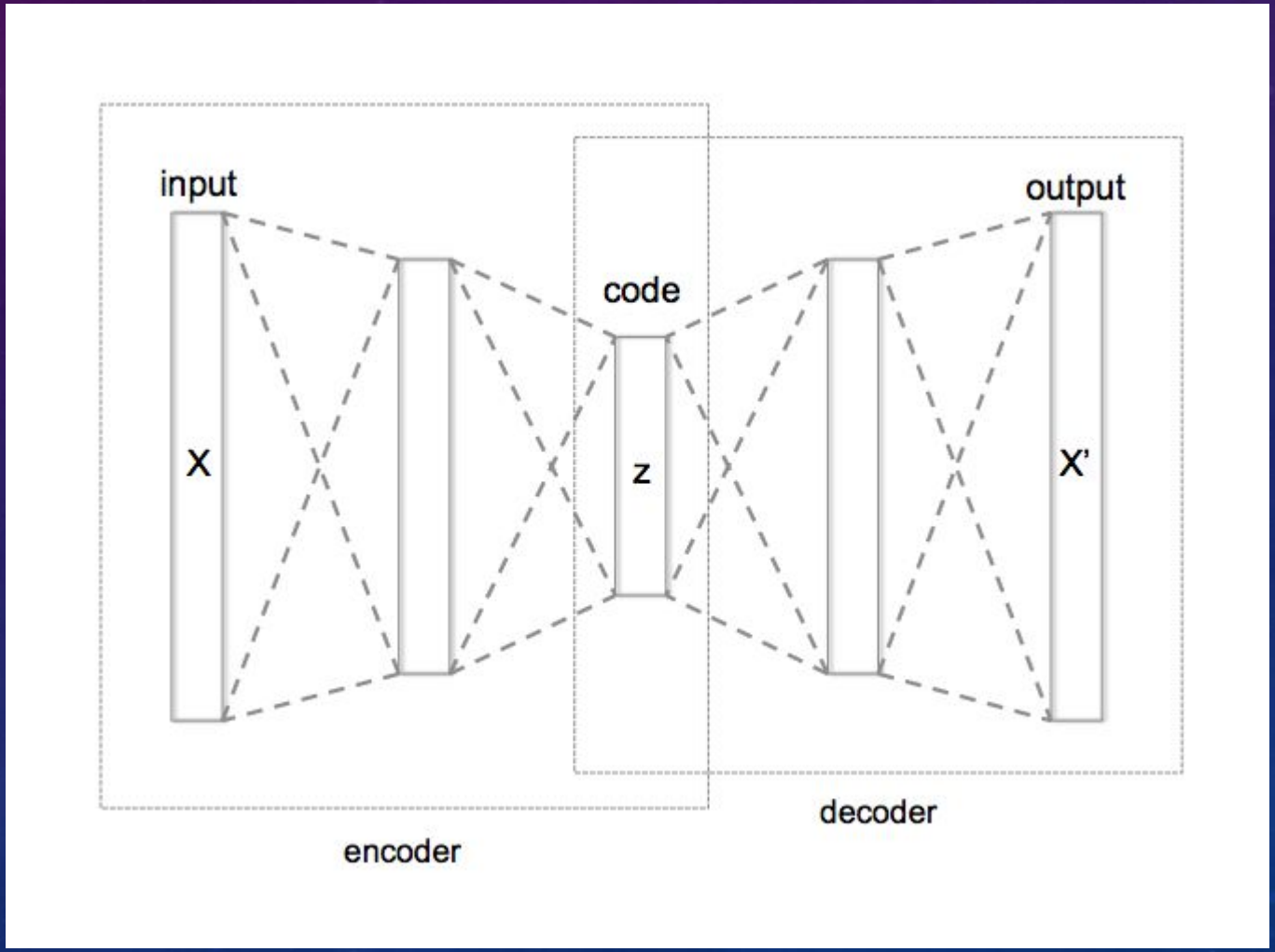
Netflix

Netflix's user movie ratings. The data were collected between October 1998 and December 2005.

- 5-star rating with 1-star scale
- 100,480,507 ratings
- 17,770 movies
- 480,189 users

NETFLIX

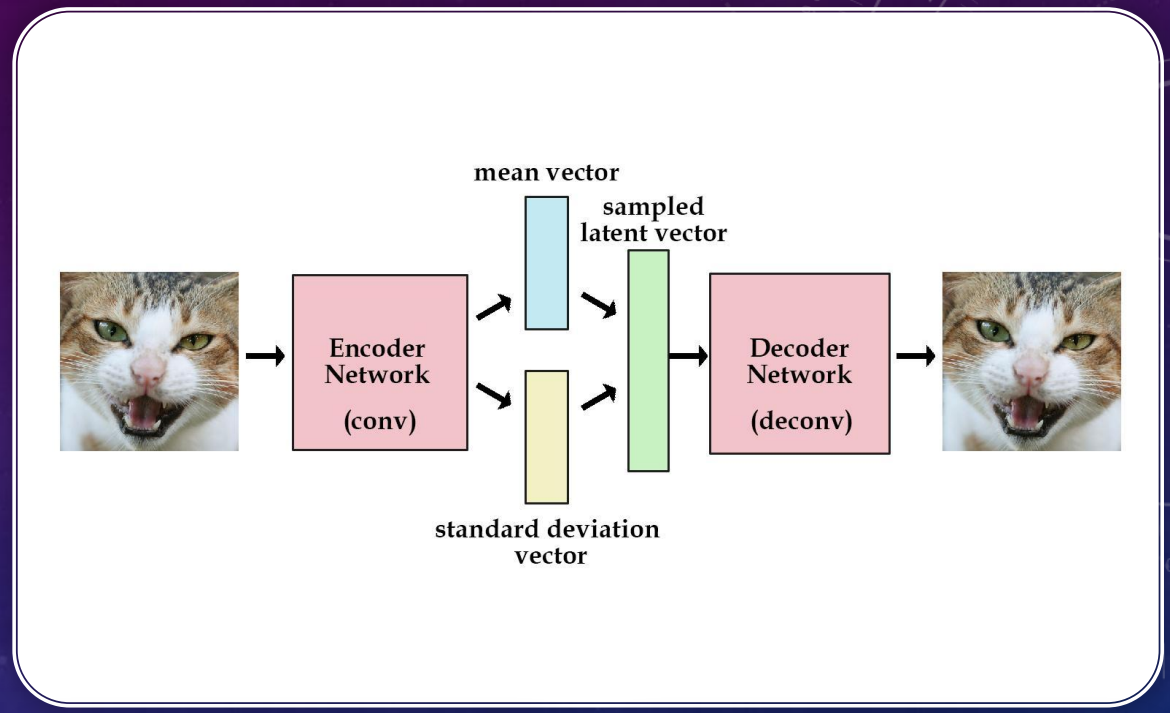
AUTOENCODER

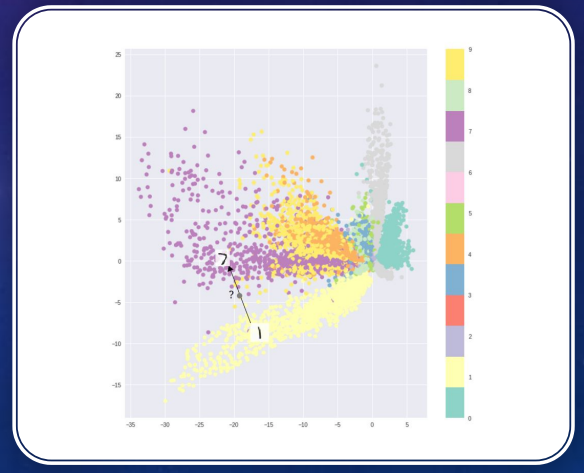


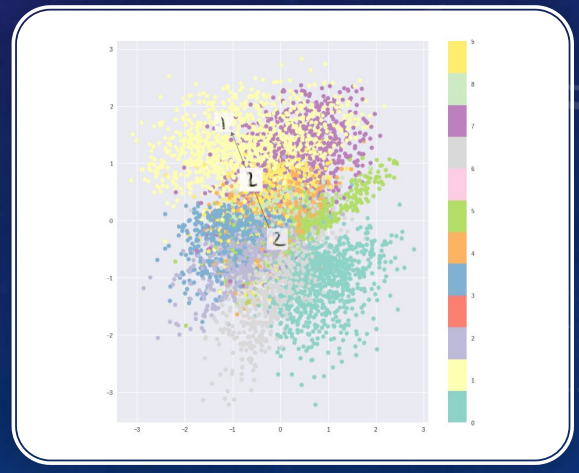
VARIATIONAL AUTOENCODER

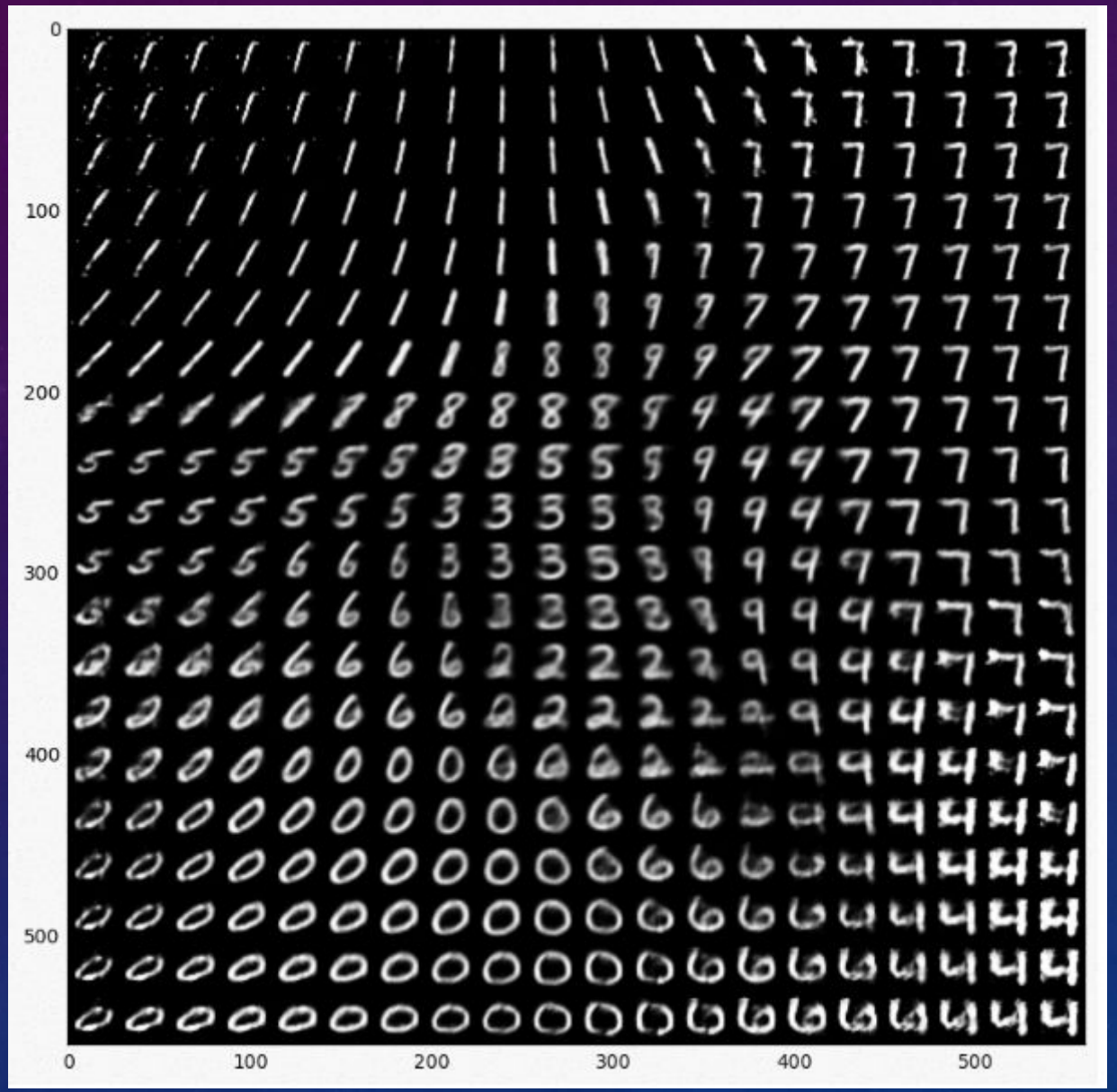
- A plain autoencoder but with sampling and additional loss:

```
generation_loss =
mean(square(generated_image -
real_image))

latent_loss =
KL-Divergence(latent_variable,
unit_gaussian)

loss = generation_loss +
latent_loss
```








0
100
200
300
400
500
0
100
200
300
400
500

VARIATIONAL AUTOENCODER - ABSTRACTLY

1. We are trying to model the distribution of images P(X)
2. It is easier to first decide what to draw, and then how -> use latent variable
3. One can model complex distributions by mapping a normal distribution -> assume z normal

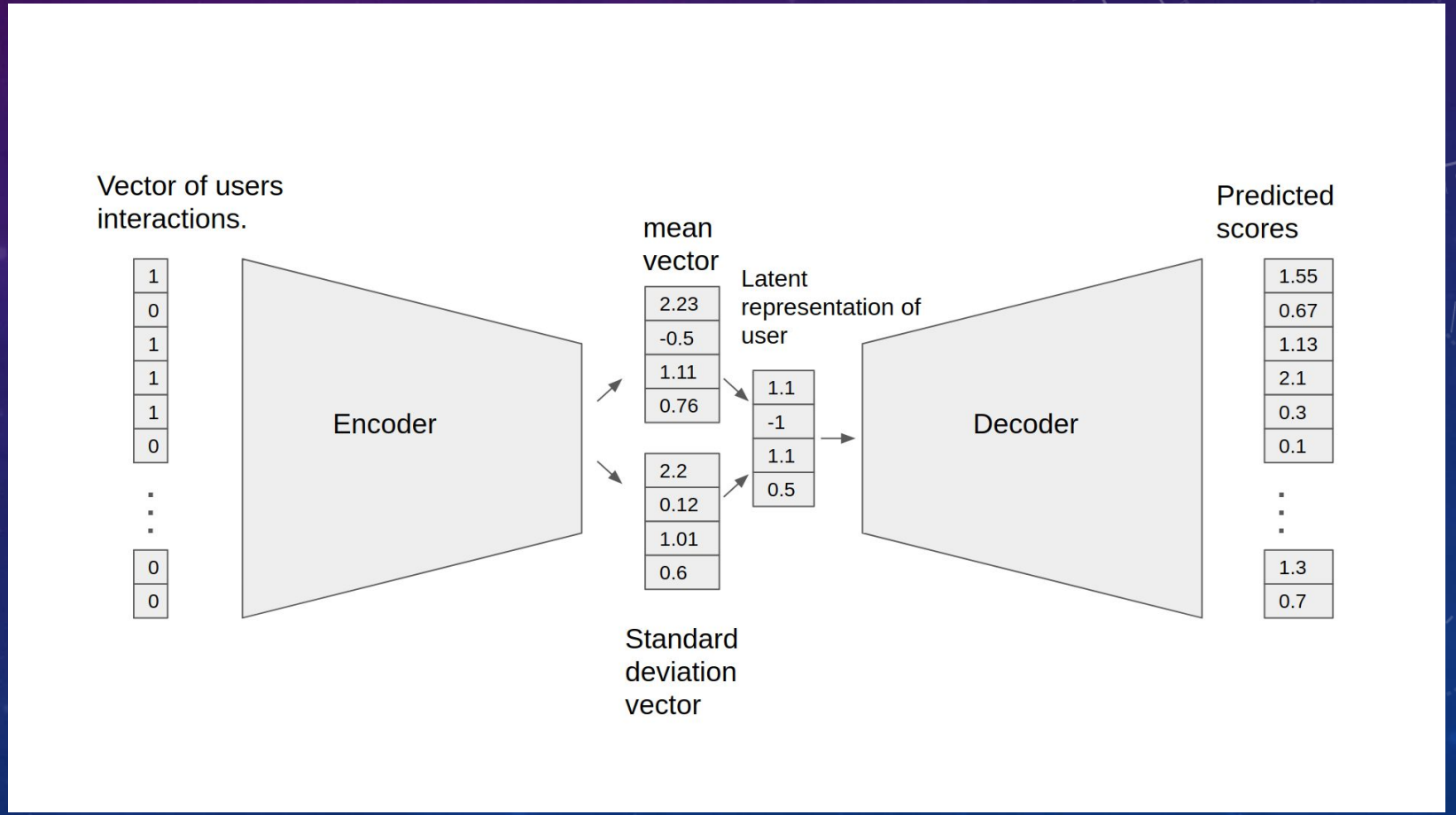
$$P(X) = \int P(X|z;\theta)P(z)dz.$$

```
X - data we're modeling   (like images)
z - latent variable       (like which number to draw)
θ - model parameters      (network weights)
```

VARIATIONAL AUTOENCODER FOR COLLABORATIVE FILTERING

- VAE for CF introduces multinomial log-likelihood as generation loss

$$\sum_i x_{ui} \log \pi_i(z_u)$$

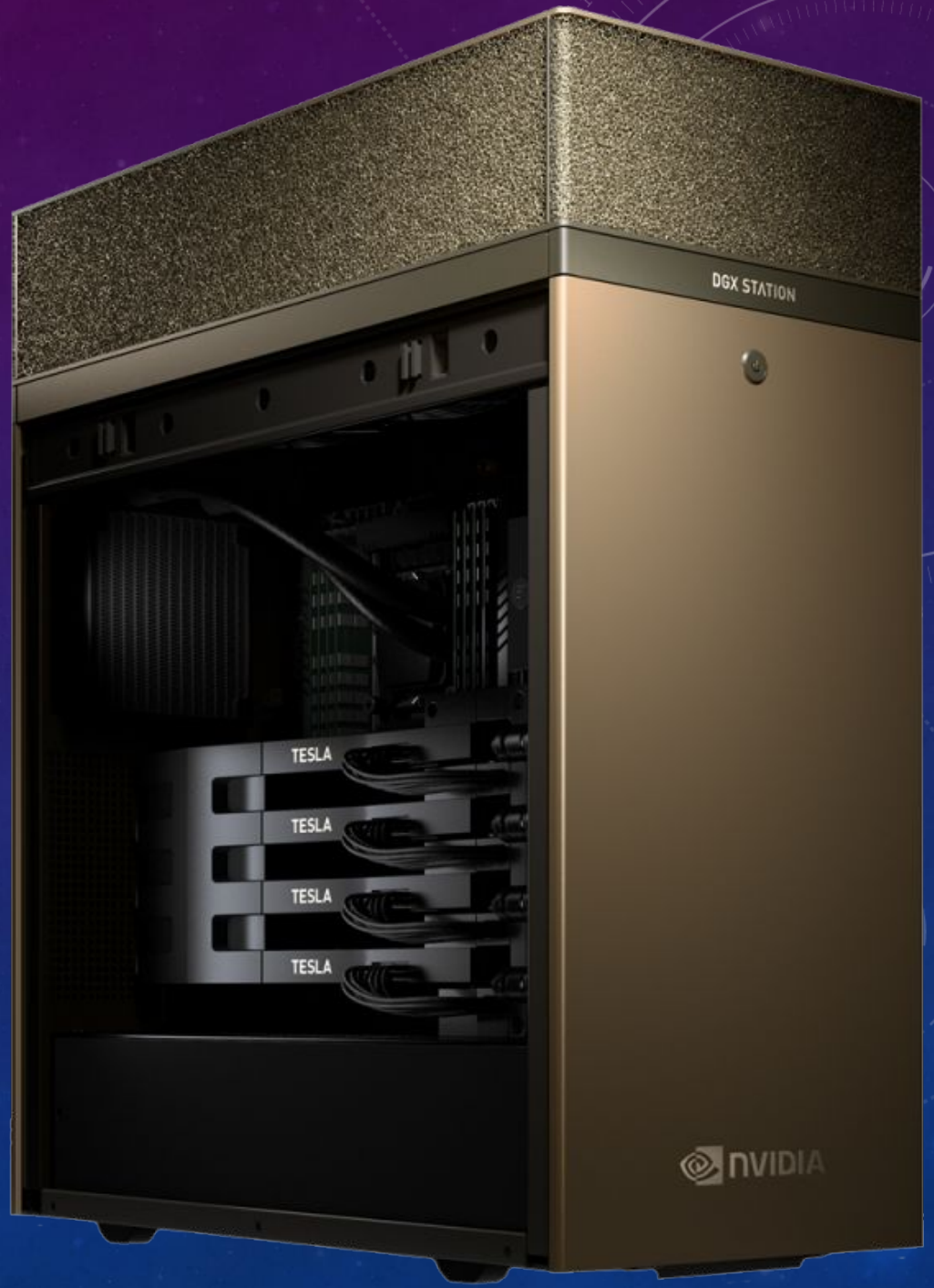


CASE STUDY

HOW TO SPEED IT UP?

BUT FIRST, HOW IT ALL STARTED?

- Bachelor’s thesis with DL expert from NVIDIA
- Original code written by paper’s author posted on GitHub
- DGX Station workstation from NVIDIA (4x Tesla V100 32 GB GPU – worth 100k$)
- Goal: show how great NVIDIA hardware is by speeding up the model



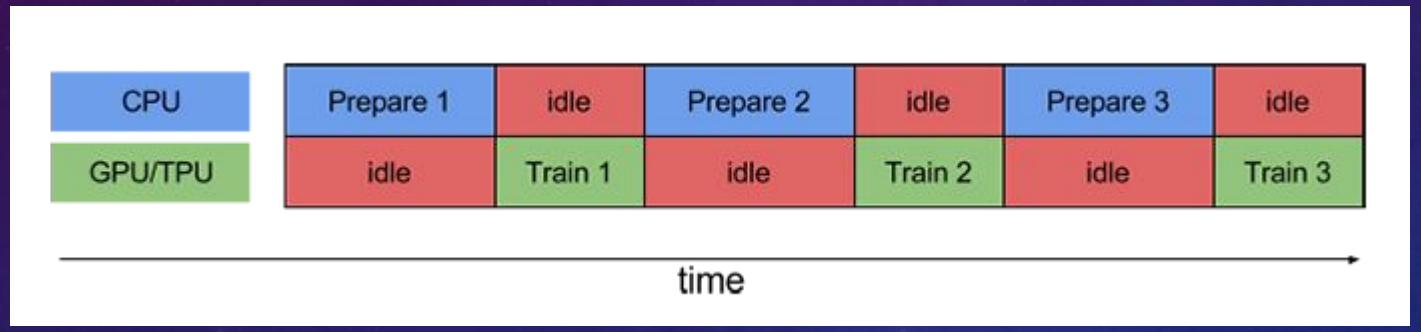
USE RIGHT DATA STRUCTURES FOR YOUR DATA

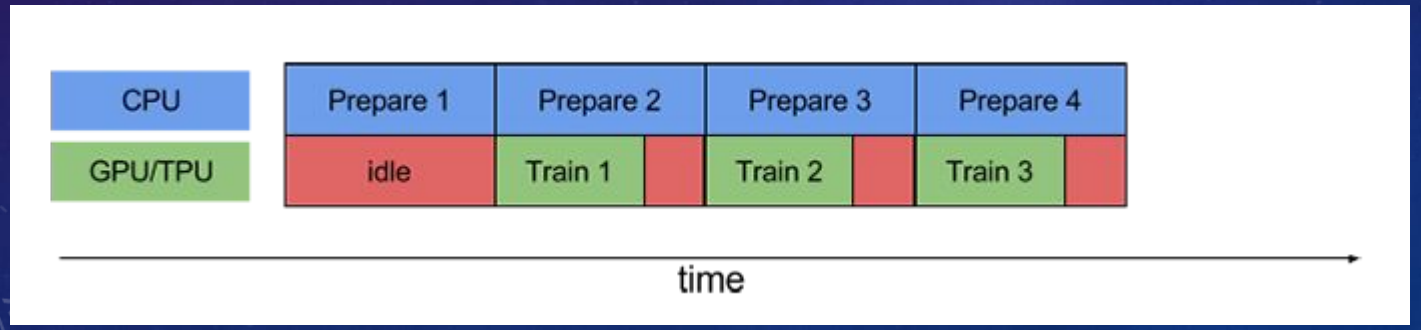
- Data in collaborative filtering problem is very sparse (around 1% of nonzero values)
- Dense matrices weight:
  - ML20M: **14GB**
  - Netflix: **32GB**
- Sparse matrices weight:
  - ML20M: **228 MB**
  - Netflix: **1.1 GB**
- Lighter matrices = faster data transfers and more data fitting in GPU memory!
- This change allowed us to increase a batch size almost 100x times (the limit here became not GPU memory but TF nonzero values indexing in sparse matrices)

Speed-up: 12.5x

KNOW YOUR FRAMEWORK – YOU USE IT FOR A REASON

- New implemented data pipelining in TensorFlow allows asynchronous calculation on CPU and GPU and loading more than one batch of data to GPU to maximize its utilisation



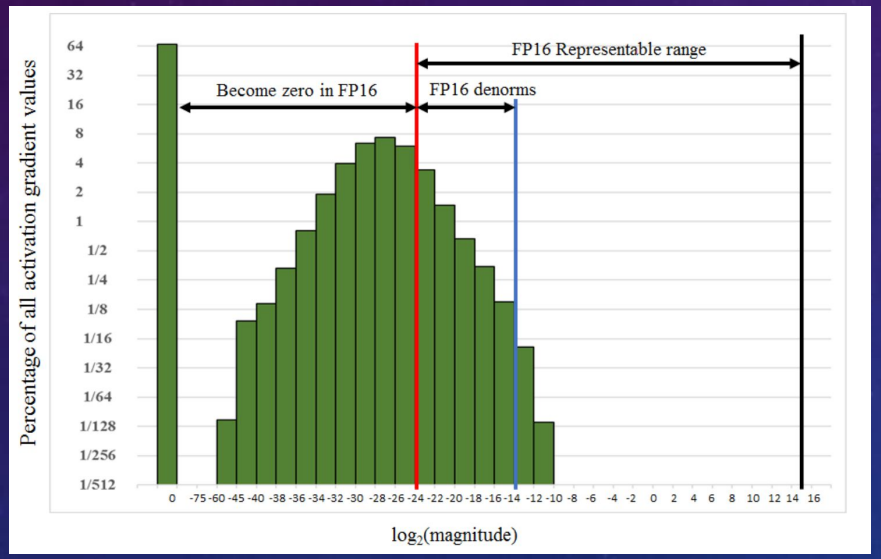


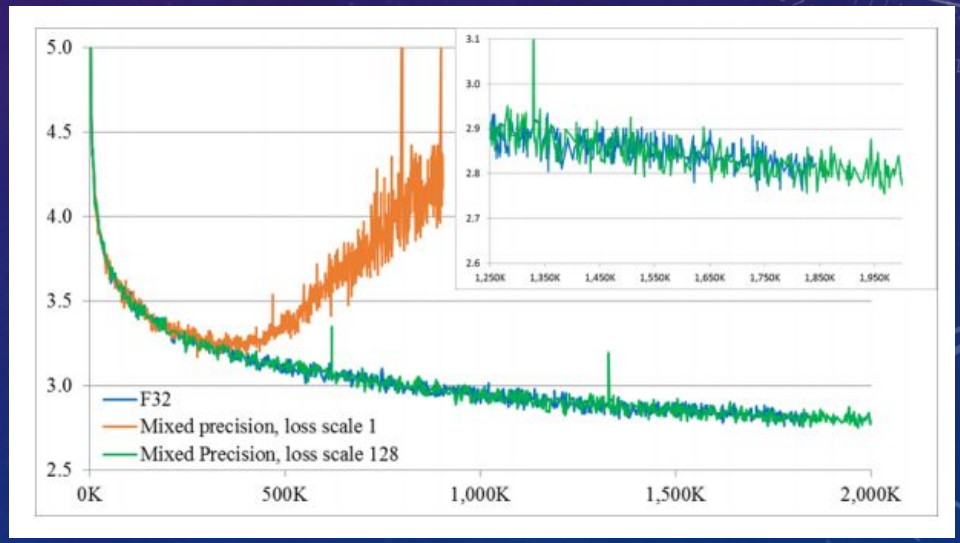
Speed-up: 1.8x

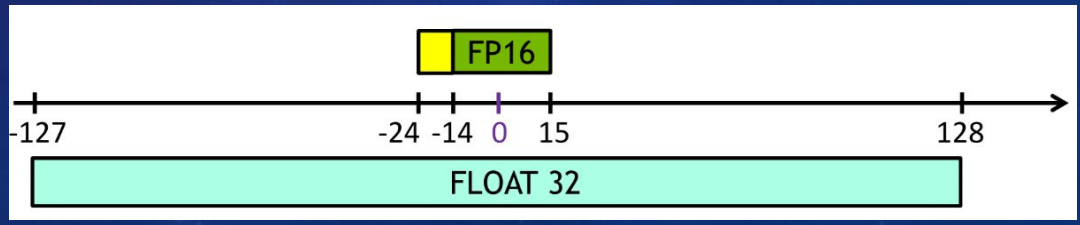
Overall: 22.6x

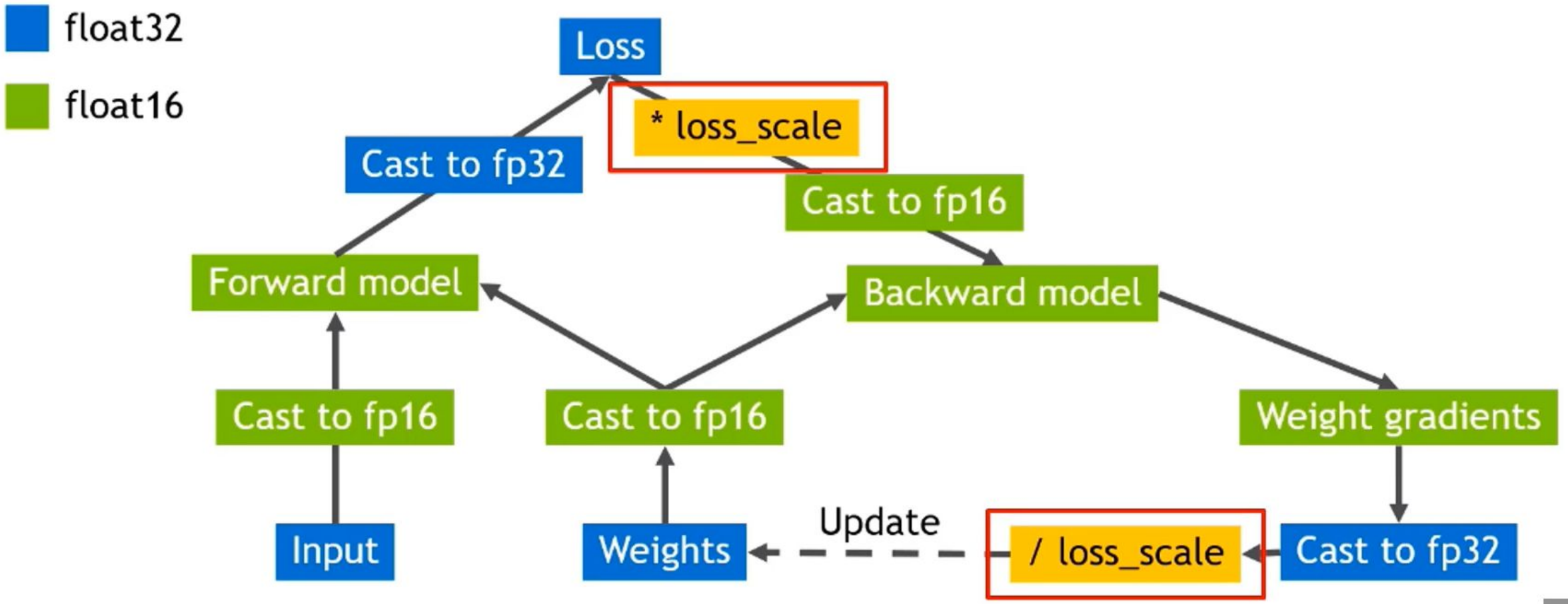
KNOW YOUR HARDWARE AND HOW TO LEVERAGE IT

- In 2017 NVIDIA released a paper entitled *Mixed Precision Training* where they have described precisely an algorithm of neural network training using 32-bit and 16-bit floats







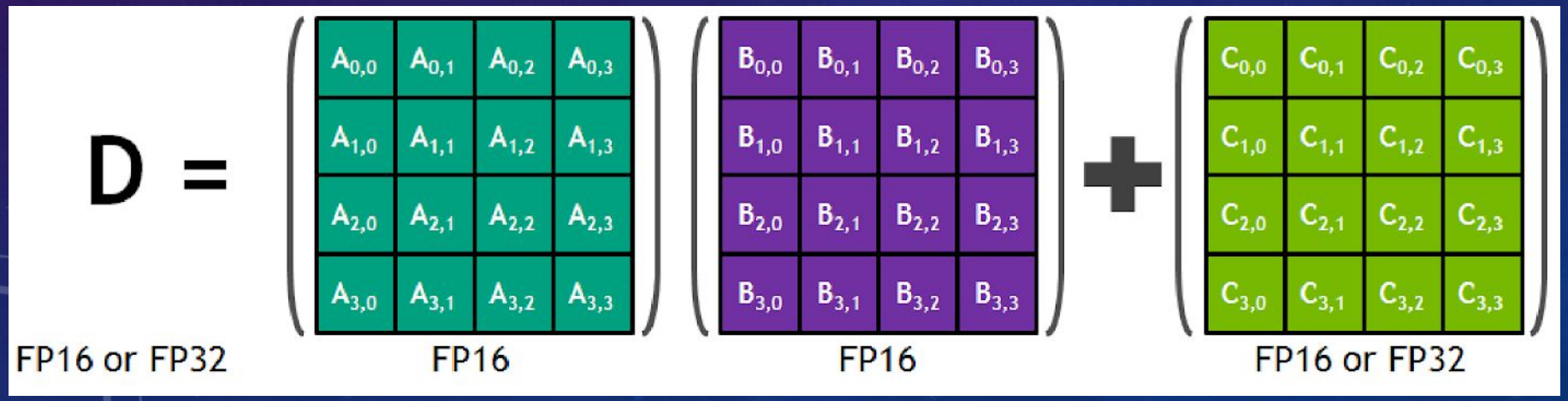

float32
float16
Loss
* loss_scale
Cast to fp32
Cast to fp16
Forward model
Backward model
Cast to fp16
Cast to fp16
Weight gradients
Input
Weights
Update
/ loss_scale
Cast to fp32

KNOW YOUR HARDWARE (AND SOFTWARE)

- Mixed Precision Training wouldn't give anything if we didn't have hardware that can take advantage of FP16
- NVIDIA wants users to use it, so they release tools for every major framework for automatic mixed precision, which perform training process in mixed precision by themselves with automatic loss scaling
- Our mentor talk about Mixed Precision Training on PL in ML '18 conference: https://www.youtube.com/watch?v=zbBUExOG-To

Speed-up: 1.8x

Overall: 41.5x



MAKE IT SCALABLE

- Our last step was training the model on multiple GPUs using Horovod – framework built at Uber for distributed neural network training
- The parallel model training is equivalent to training on a one GPU with multiplied batch size, so it may not be always helpful
- Training with Horovod can inflict some funny issues that we encountered:
  - Processes running on GPUs may die ungracefully just at the end of training if each of them don’t get equal number of batches (easiest case: number of batches not divisible by number of GPU)
  - Each GPU should get different batch, otherwise you could be just training on one GPU 😊

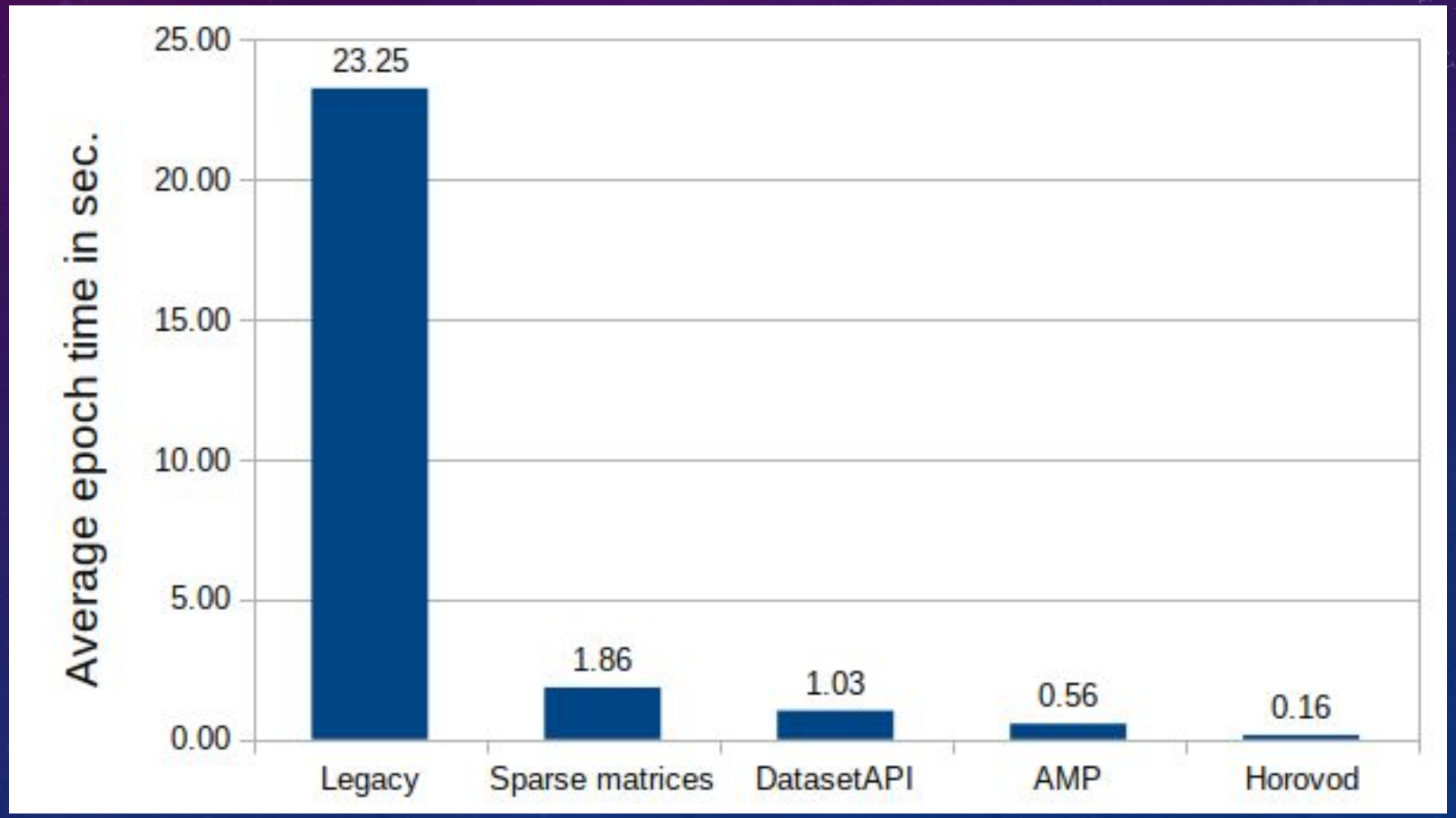
Speed-up:

3.5x

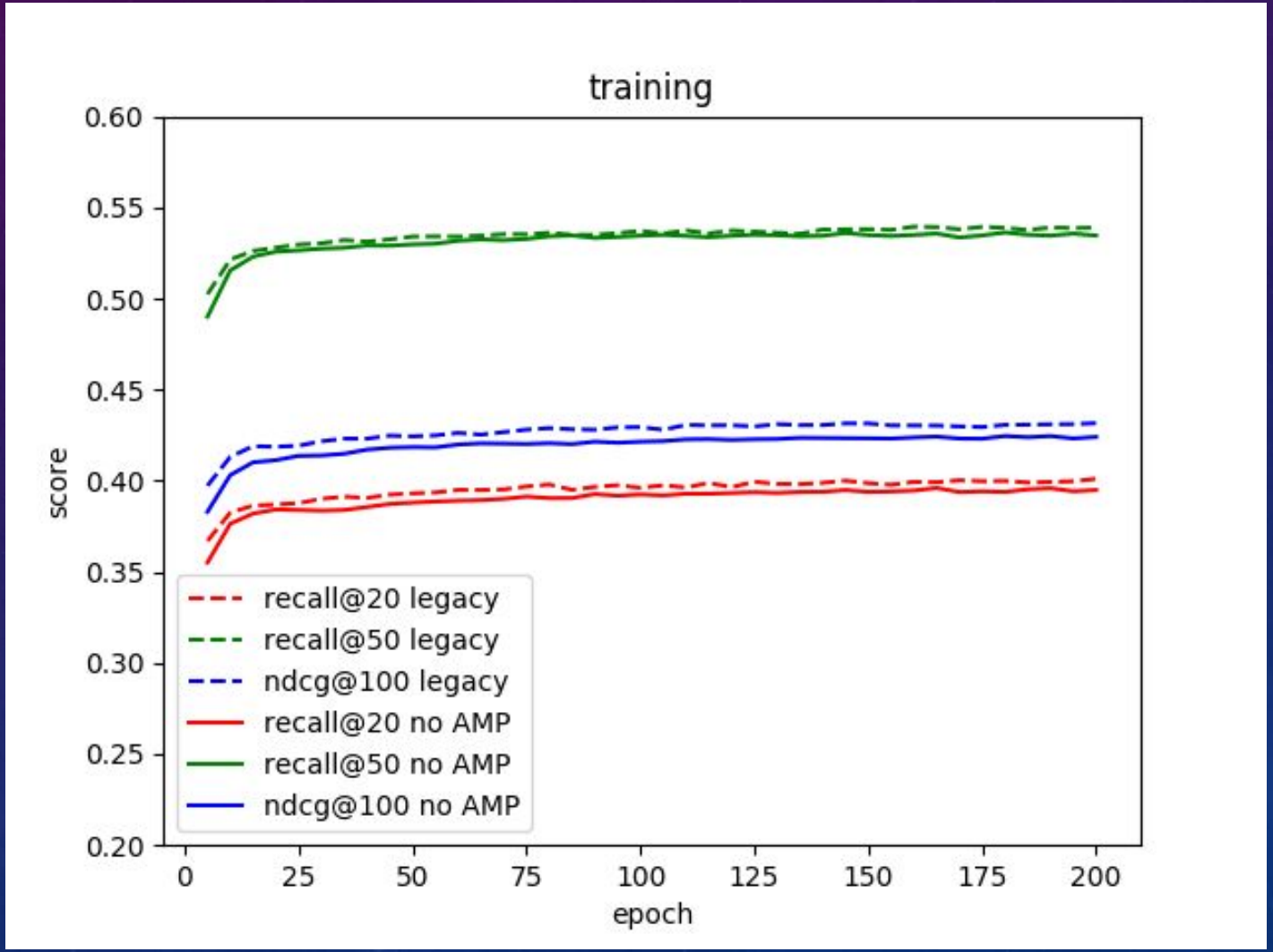
Overall:

145.3x

FINAL RESULT



PLOT TWIST – WE MESSED SOMETHING UP ☹️



BIBLIOGRAPHY

- Variational Autoencoders for Collaborative Filtering
- https://en.wikipedia.org/wiki/Precision_and_recall
- https://en.wikipedia.org/wiki/Discounted_cumulative_gain
- https://en.wikipedia.org/wiki/Evaluation_measures_(information_retrieval)
- https://www.tensorflow.org/guide/performance/datasets
- https://arxiv.org/abs/1710.03740
- https://devblogs.nvidia.com/nvidia-automatic-mixed-precision-tensorflow/
- https://www.youtube.com/watch?v=i1fIBtdhjIg
- https://github.com/NVIDIA/apex
- http://kvfrans.com/variational-autoencoders-explained/