WELLS FARGO

Using NVIDIA CUDF to Simplify and Accelerate Data Prep for Credit Card Algo. Prediction

March 19, 2019

Richard Liu
Vice President

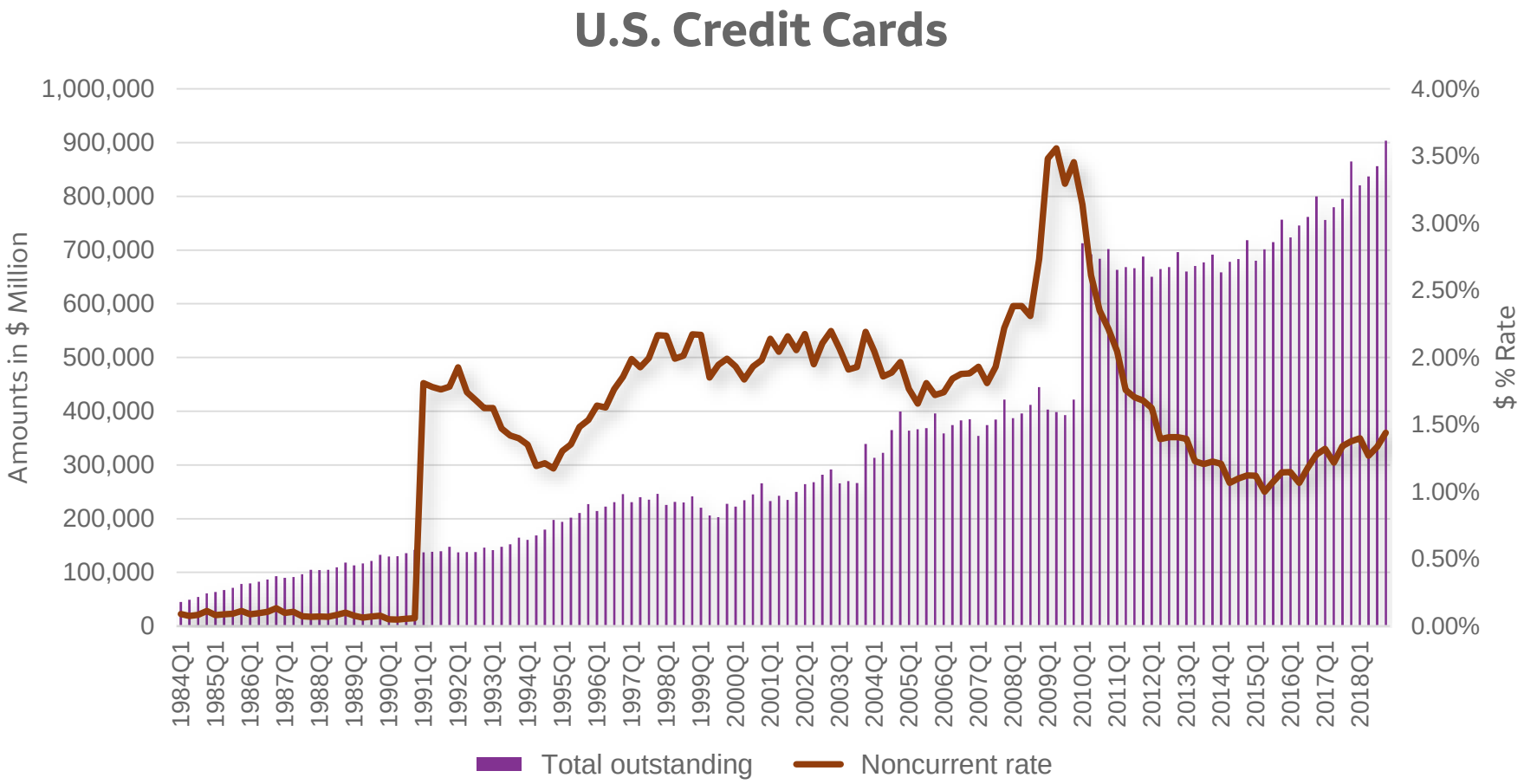
Agenda

- Macro economics trends
- Behavioral surplus
- Paradigm shift
- Deep dive to the data
- How RAPIDS/cuDF helps

Perspective on the challenges

Business case:

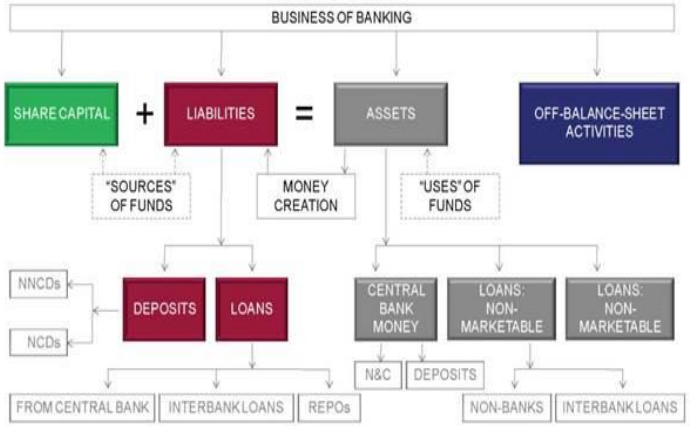
Credit card business now faces the challenges on risk management and more importantly on payment or transaction behavior. The conventional balance sheet data approaches can hardly afford such new requirement.



Source from FDIC: Loan Performance (as of 2018/Q4)

Trade secret on behavioral surplus

Traditional



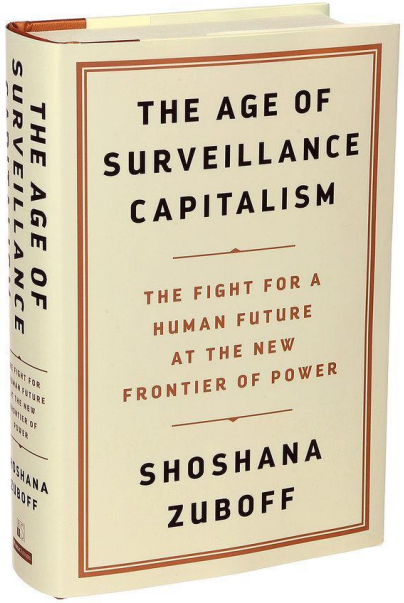
Pool level thinking

Book keeping

Digital Age

Action to behavior to data to prediction

Surveillance capitalism



Paradigm Shift

But … How to walk the talk?

Now We Have New Way To Look At Data

Examples (simulated data for illustration purpose)

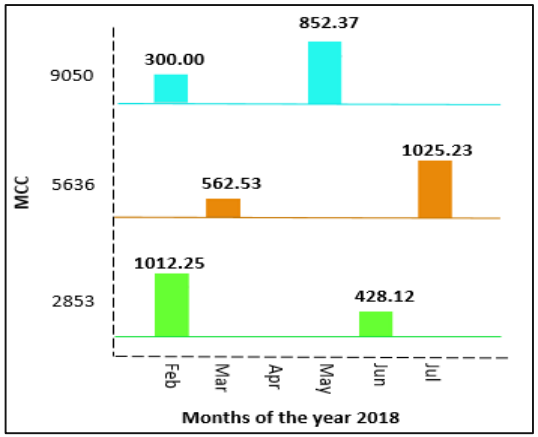
Customer ID: cust_id
Merchant category code: mcc
Transaction date: trans_date
Dollar amount: trans_amt

cust_id	mcc	trans_date	trans_amt
100695	2853	2-2018	1012.25
100695	5636	3-2018	562.53
100695	9050	2-2018	300
100695	5636	7-2018	1025.23
100695	2853	6-2018	428.12
100695	9050	5-2018	852.37

Array objects after pivoting process:
[array of mcc],
[array of trans_date],
[array of trans_amt]

```
[10695[2853,    5636,   9050,   5636,    2853,   9050  ],
      [2-2018,  3-2018, 2-2018, 7-2018,  6-2018, 5-2018],
      [1012.25, 562.53, 300,    1025.23, 428.12, 852.37]]
```

Neuroscience observation on customer behavior
(Visualization)



Why RAPIDS cuDF

Progress so far has largely been toward demonstrating general approaches for building narrow systems rather than general approaches for building general systems. Progress toward the former does not entail substantial progress toward the latter.

AlphaGo and AI Progress. Retrieved October 24, 2017, from http://www.milesbrundage.com/blog-posts/alphago-and-ai-progress.

Our expectation:

- The efficient way to deal with very sparse data against computation
- Performance with ease of programming (Python Pandas like)
- Much better return on GPU solution investment

The advantage of modern computation:

Functional language:

```
increment :: [int] -> [int]
increment = map (1+)
```

How RAPIDs Helps On Transaction Over Time Horizon

Easier yet efficient way to resolve the chronic “horizon stacking” data

Conventional

```
SELECT
        COUNT(),
        SUM(),
        STD(),

        PARTITION BY () # Window function
FROM
 … LEFT JOIN …
GROUP BY …
```

Distributed over GPU cores

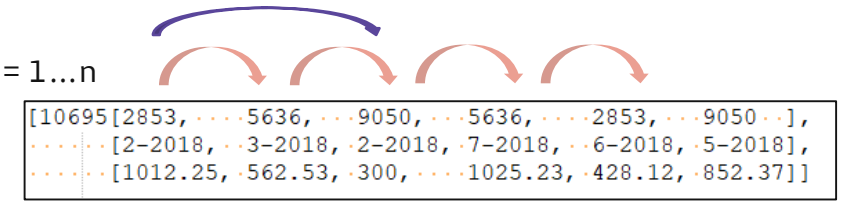
```
PATTERN = HORIZON();

(0 until array.length)
        .map(i = PATTERN
        .addData( attributes(i),array(i)))
```

Data object

Smart distributed computation by RAPIDS

cust_id	mcc	trans_date	trans_amt
100695	2853	2-2018	1012.25
100695	5636	3-2018	562.53
100695	9050	2-2018	300
100695	5636	7-2018	1025.23
100695	2853	6-2018	428.12
100695	9050	5-2018	852.37

Time interval = 1…n

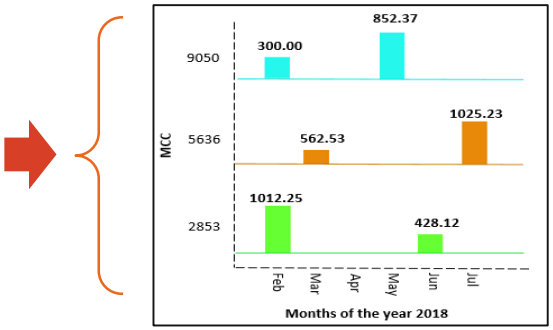


Challenges from DL computation

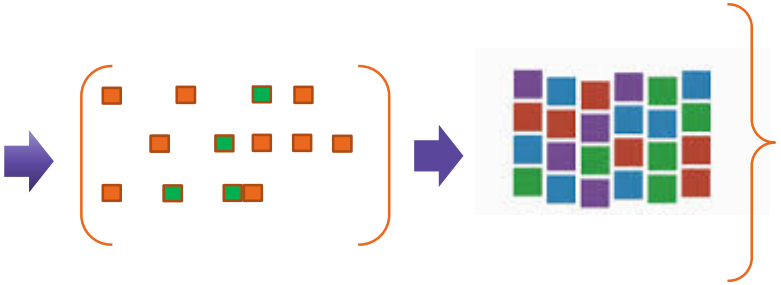
With conventional table way, how to find a departure from the prevailing deep learning zeitgeist that prizes learning from scratch, tabula rasa.

Table with system records

cust_id	mcc	trans_date	trans_amt
100695	2853	2-2018	1012.25
100695	5636	3-2018	562.53
100695	9050	2-2018	300
100695	5636	7-2018	1025.23
100695	2853	6-2018	428.12
100695	9050	5-2018	852.37



Hebbian learning like representation

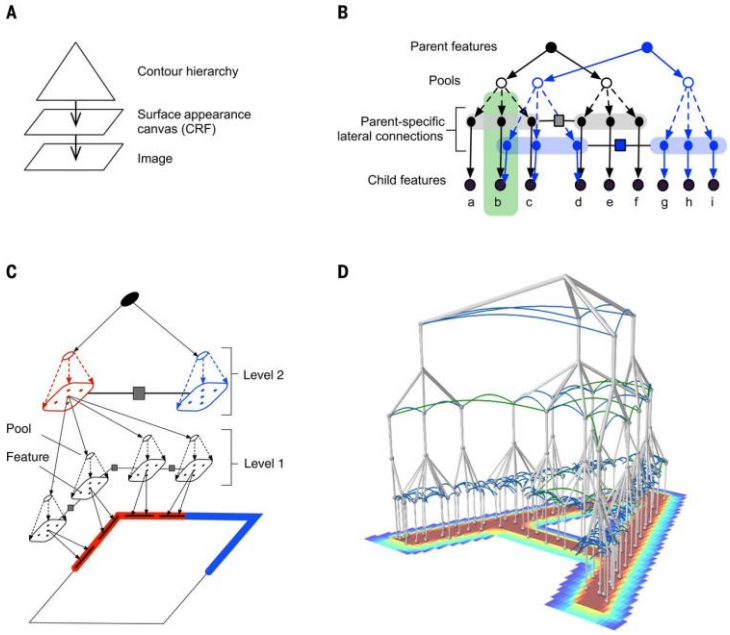


SDR

CuDF with Better format for Deep Learning Like Computation

Inspiration from Recursive Cortical Network, Hierarchical Temporal Memory

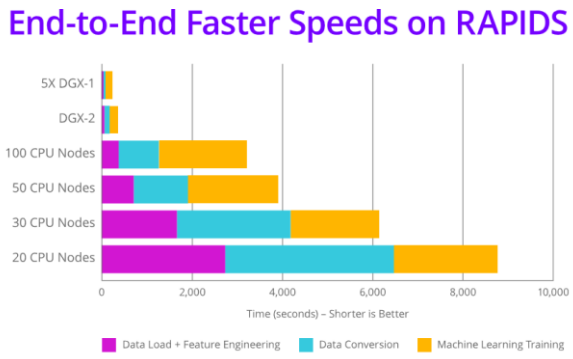
```
function feature_map(hierarchical, data[1.. 𝒯 ], 𝒞)
        levels[1..𝒟] <- hierarchical.levels
        for l <- 1 to 𝒟 do
                regions <- levels[l].regions
                for all r in regions do
                Until spatialPooling converged for r
                  for t<-1 to 𝒯 do
                    spatialPooling(r, data[t])
                  end for
                end Until
                for c <- 1 to 𝒞  do
                  for t <- 1 to 𝒯 do
                    spatialPooling(r,data[t])
                    Sparse_Data_Representation <- pivoted_array
                    Time_Horizon_Pooling(r, Sparse_Data_Representation)
                  end for
                end for
          end for
        end for
end function
```

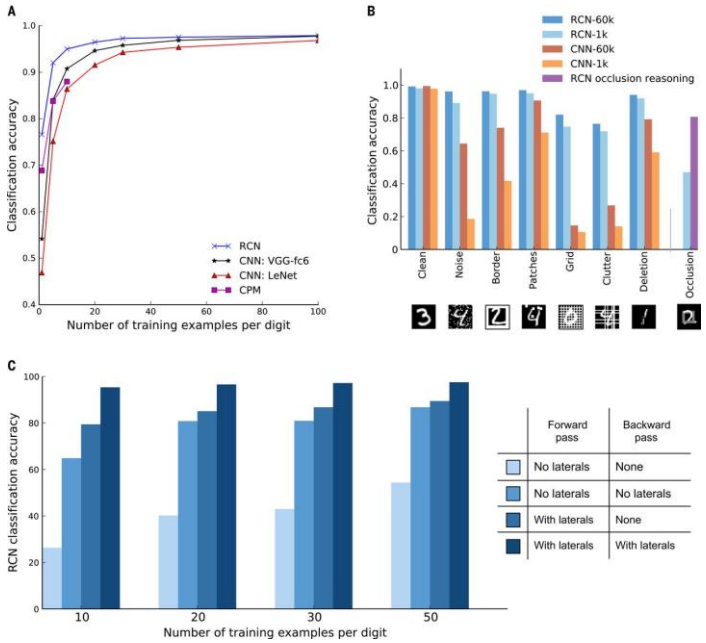


* *Dileep George et al. Science 2017;358:eaag2612* (published by AAAS)

How Much RAPIDs Helps

- Speed, speed, speed! Things you should know by yesterday.
- More time to think (smart machine for smart people).
  - Feature engineering (more and accurate)
  - Computational significance (less data yet robust to noise)





Dileep George et al. Science 2017;358:eaag2612 (published by AAAS)
Github scripts: https://github.com/vicariousinc/science_rcn

WELLS FARGO

Thank you