SPARK+AI
SUMMIT EUROPE

Using Deep Learning in Production Pipelines
to Predict Consumers' Interest

Mathieu DESPRIEE, Tinyclues

#SAISML2

Who am I ?

Mathieu DESPRIEE

ML Engineering Manager at tinyclues
In charge of ML Scalability & Operability

Contributor to Spark, MxNET

TINYCLUES IN A FEW WORDS

AI-FIRST
SAAS SOLUTION

DEEP LEARNING FOR CAMPAIGN
TARGETING & PLANNING

ACROSS ALL
CHANNELS

DRIVE MORE REVENUE
AND ENGAGEMENT

DESIGNED FOR
MARKETERS

USING ANONYMIZED
FIRST PARTY DATA

SEAMLESSLY INTEGRATED WITH
YOUR MARKETING STACK

FIRST CAMPAIGNS IN
2 WEEKS

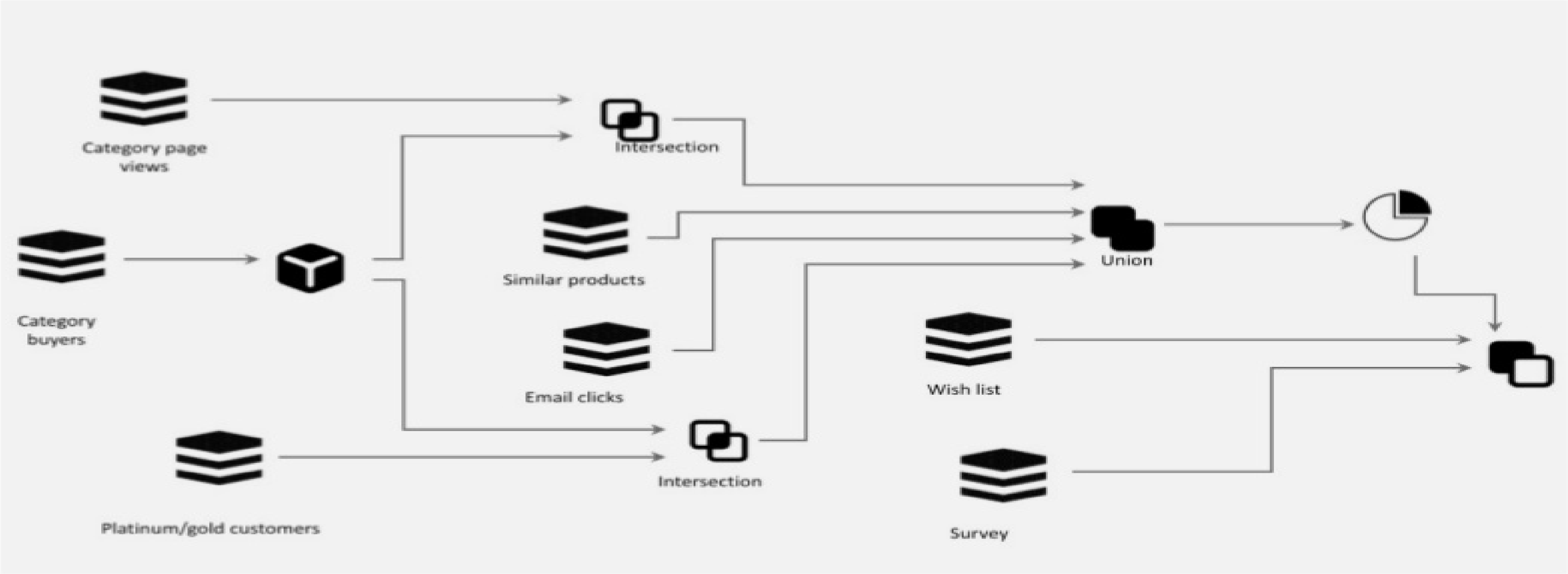
Instead of...

INTUITION

RFM

DEMOGRAPHICS

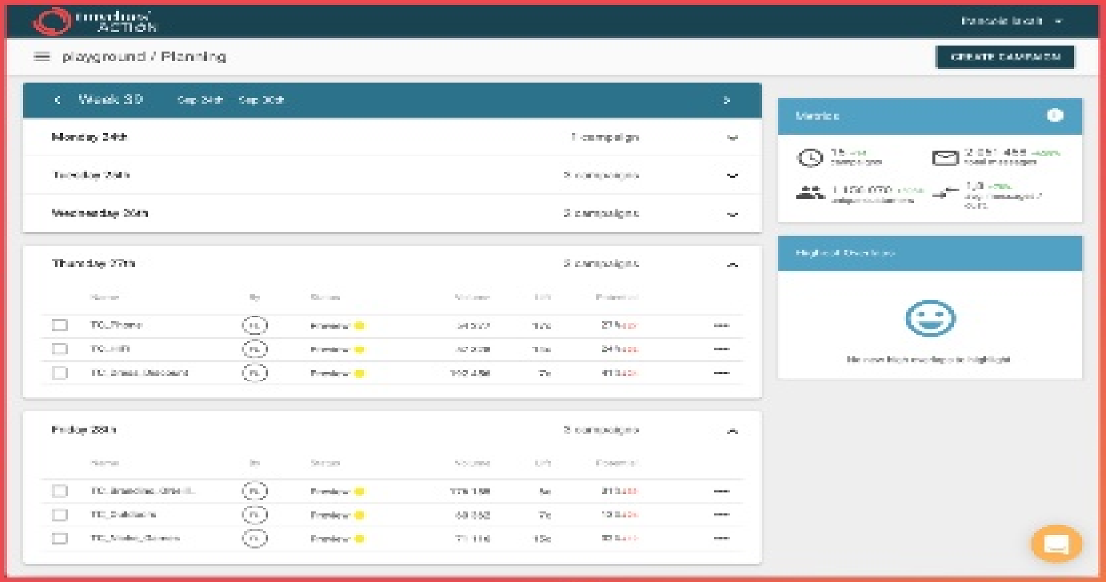
LAST PURCHASE



AI-FIRST: INTELLIGENT CAMPAIGNS IN JUST A FEW CLICKS

SPOT ON

TARGET THE FUTURE BUYERS FOR ANY PRODUCT, IN THE DAYS FOLLOWING A CAMRAIGN



START FROM YOUR CAMPAIGN IDEAS AND BUSINESS GOALS

STRATEGIC

PRIORITIZE BASED ON VALUE & MARKETING FATIGUE

EASY

BUILD SEGMENTS IN MINUTES – NO GUESSWORK & NO DATA-SCIENCE NEEDED

QUANTIFIED

ANTICIPATE & MAXIMIZE CAMPAIGN PERFORMANCE

SUCCESS STORIES

$1B+
+115%
CAMPAIGN REVENUE
Oui.sncf
TRAVEL
10M+ customers

$1B+
+30%
CAMPAIGN REVENUE
fnac
RETAIL
10M+ customers

$1B+
+151%
REVENUE PER EMAIL
LACOSTE
RETAIL / FASHION

$1B+
+178%
CAMPAIGN REVENUE
ACCOR HOTELS Feel Welcome
HOSPITALITY
5M+ customers

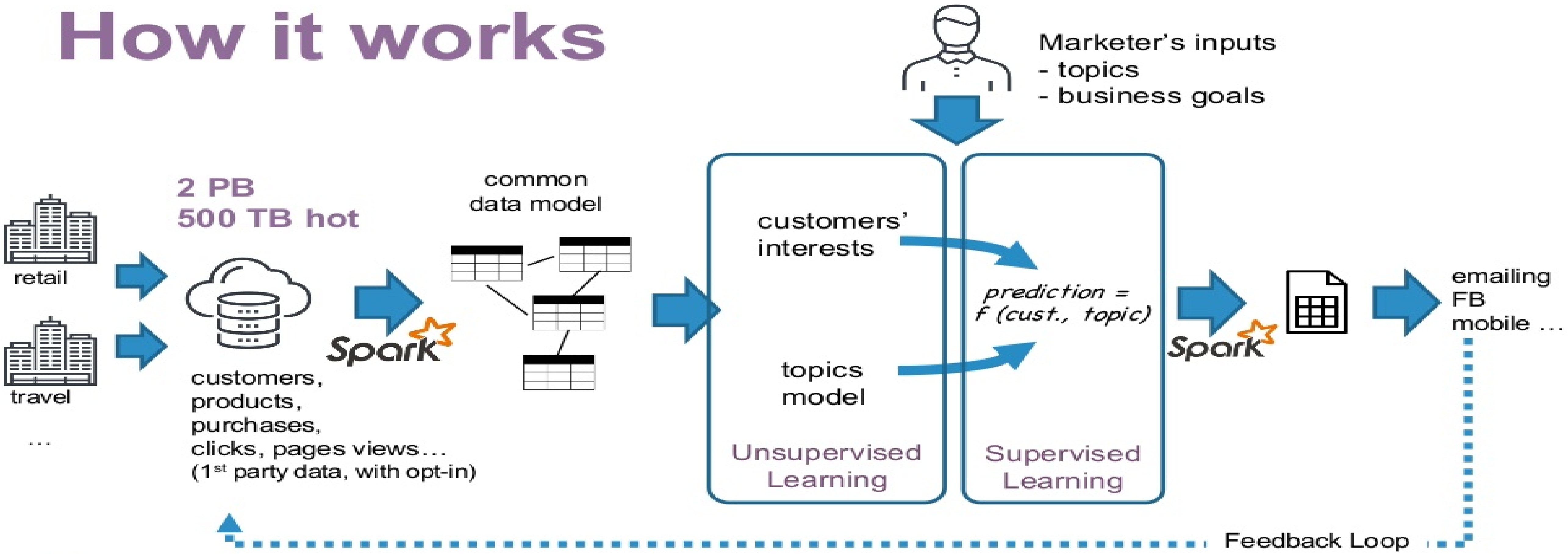
$1B+
+30%
CAMPAIGN REVENUE
Club Med
HOSPITALITY

+20%
CAMPAIGN REVENUE ONLINE & IN-STORE
CYRILLUS PARIS
RETAIL / FASHION

$1B+
+80%
CAMPAIGN REVENUE
E-COMMERCE
10M+ customers

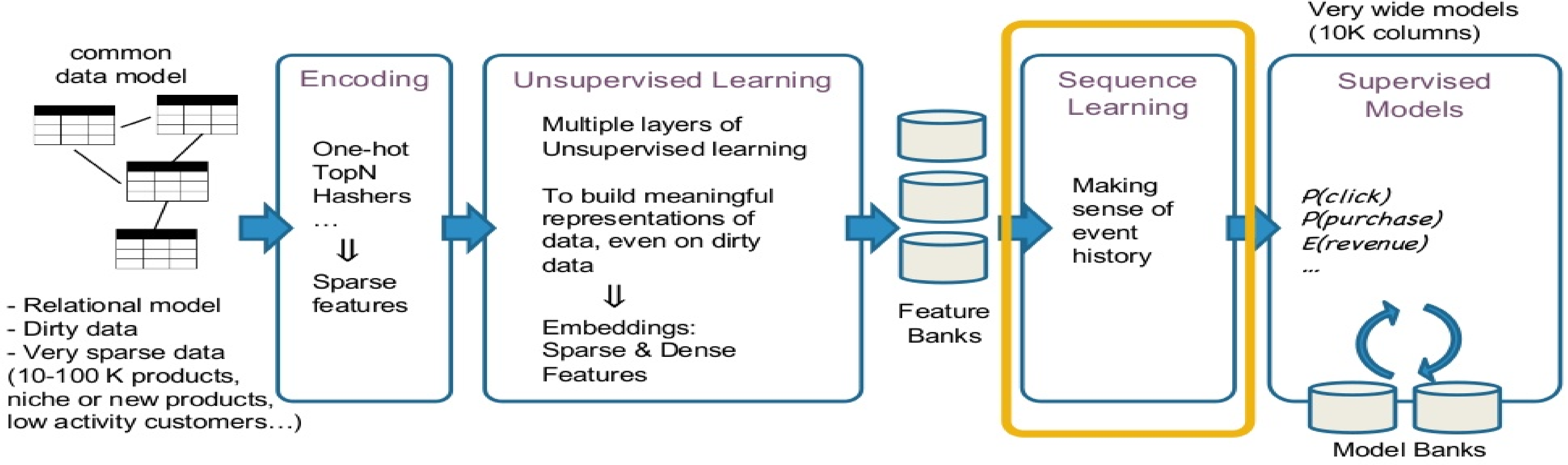
+60%
REVENUE PER EMAIL
VESTIAIRE COLLECTIVE
RETAIL / FASHION

How it works



Marketer's inputs
- topics
- business goals
2 PB
500 TB hot
retail
travel
...
customers,
products,
purchases,
clicks, pages views...
(1st party data, with opt-in)
Spark
common
data model
customers'
interests
topics
model
Unsupervised
Learning
prediction =
f (cust., topic)
Supervised
Learning
Spark
emailing
FB
mobile ...
Feedback Loop

Predictive architecture

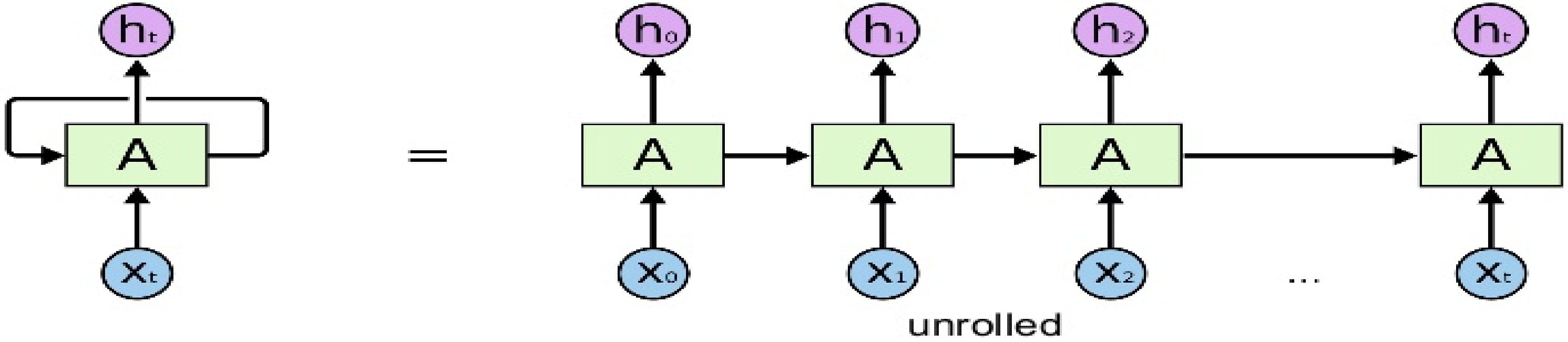


Working with events

- Events (page views, clicks...) are not always meaningful individually, but sequences are
- Events are ordered, and are not independent from each other
- Sequences are not all of the same length
- Can we find a model architecture leveraging on these characteristics ?

RNN: Recurrent Neural Networks

- Networks with loops in them, allowing information to persist.



- They also allow input sequences of variable length
  - you just need to pad data with zeroes $X_0..X_k$

Illustrations by Christopher Olah - Check out his blog! colah.github.io

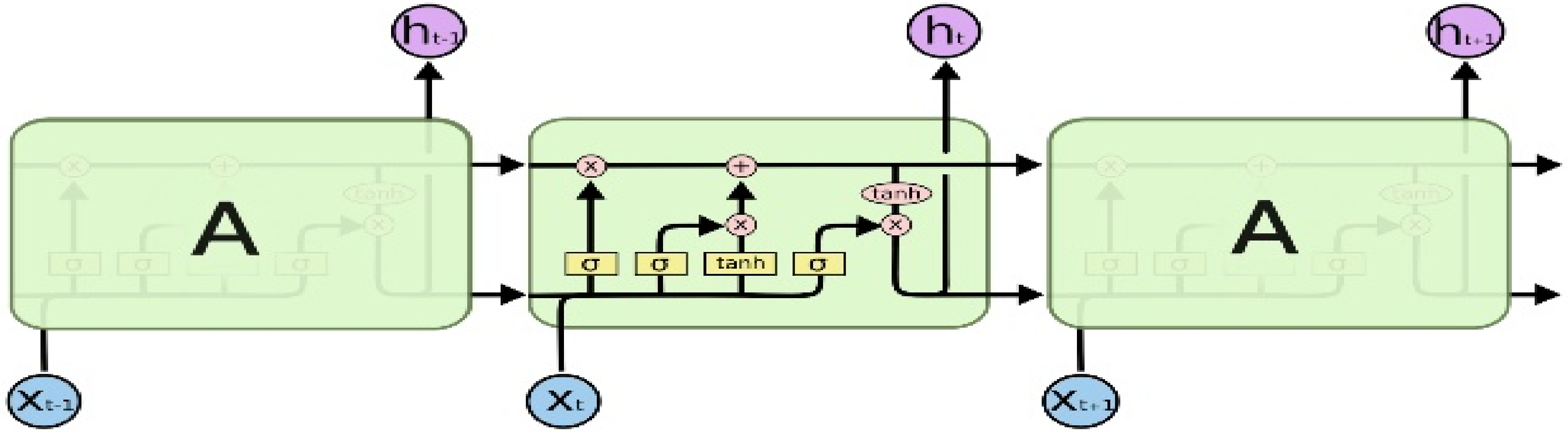
LSTM: Long Short-Term Memory

- Classical RNN can't learn long-term dependencies with gradient descent because of vanishing or exploding gradients: they tend to forget in the long-term

- LSTMs are explicitly designed to avoid this problem
  - Invented by Hochreiter & Schmidhuber (1997)
  - Many variants exist

- LSTM have great results in speech recognition, translation, and are building block of assistants.

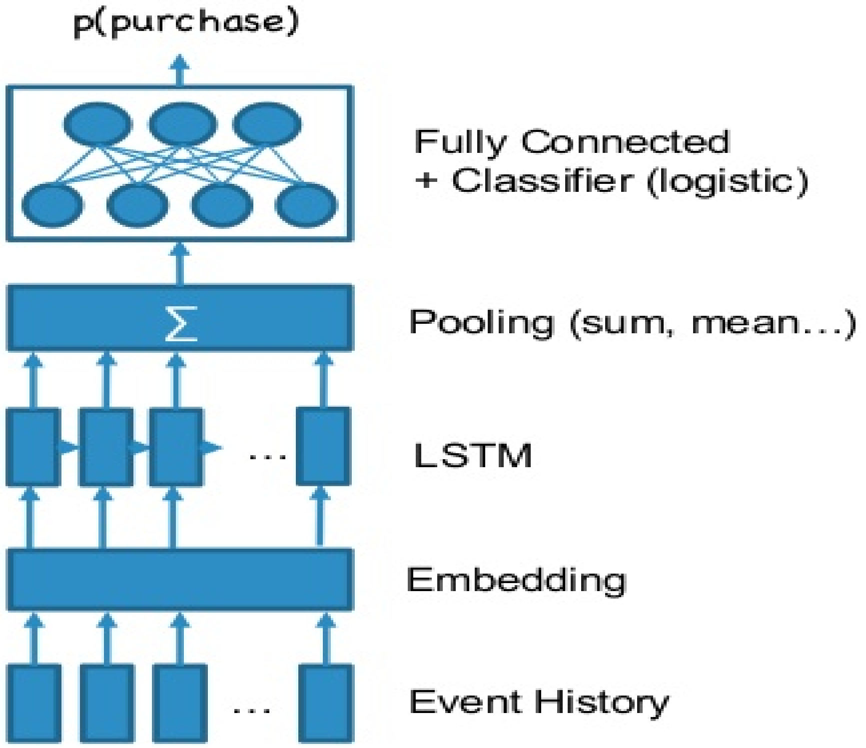
LSTM



- Cell-state is transmitted from cell to cell (top line) without going through a non-linear activation function
- The rest of the cell controls:
  - how the input and previous outputs modifies the cell state,
  - how the state and input are combined together to form output

Example of LSTM for purchase prediction

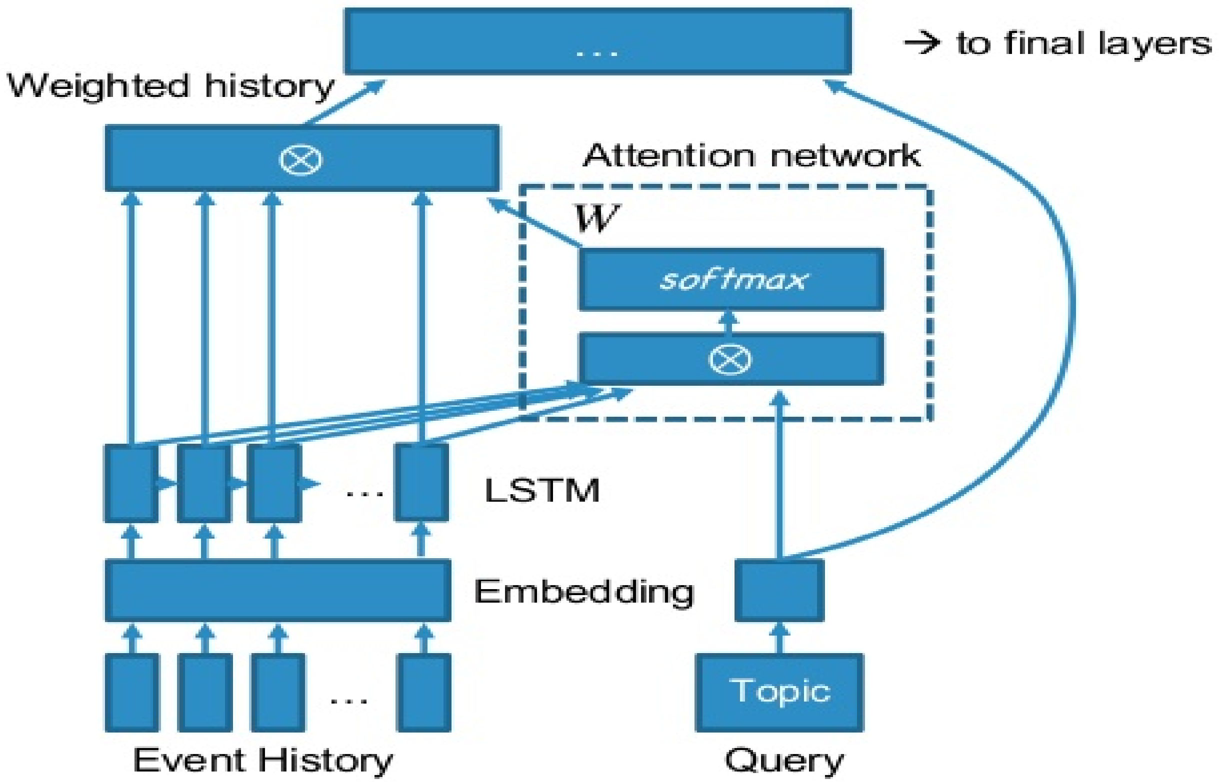
- Inputs are events (page views, clicks), padded to a max sequence length, and passed through an embedding layer
- LSTM output is aggregated with a pooling layer, and some densely connected layers
- Through gradient descent, the network will learn what event sequences are more likely to lead to the target (purchase)



<u>"embedding"</u>: representing an information by a vector in a space, capturing its essential meaning

Bringing some focus: Attention Model

- Attention models are a way to make the network focus on elements of interest
- Introduce the notion of “Query” = what the network should focus on (topic)
- The attention network outputs weights W that modulate the importance of each LSTM output, given the topic
- This weighted and interpreted history is then used in subsequent layers for classification



Implementation & Execution Engine

- Desired technology for production:
  - Easily distributable with Spark
  - Callable from Scala
  - Interoperability with Python (used in our Lab)
  - High performance out-of-the-box
    - should benefit from native libs
    - ability to use GPU for training phases
  - Good support of Sparse tensors

Apache MxNET

- Deep Learning framework, incubated at Apache
- Back-end in C++
- Multiple front-end languages: scala, python, Julia, R, …
- Integration with linalg native libs, including Intel MKL
- Benchmarks with GPUs are very good
- Aims at interoperability with others frameworks (ONNX, MxNET as Keras backend, etc.)

LSTM w/ Attention in Python

```
class NetLSTM(mx.gluon.nn.HybridSequential):
    def __init__(self, embedding_shape, **kwargs):
        super(NetLSTM, self).__init__(**kwargs)
        with self.name_scope():
            self.embedding_shape = embedding_shape
            self.embeddeding = self.params.get("embeddeding", shape=embedding_shape, grad_req='null')
            self.lstm = mx.gluon.rnn.LSTM(hidden_size=16, layout="NTC")
            self.lstm_pool = mx.gluon.nn.Dense(units=1, flatten=False)
            self.deep1 = mx.gluon.nn.Dense(units=10, activation="relu", flatten=False)
            self.deep2 = mx.gluon.nn.Dense(units=1, activation="sigmoid", flatten=False)

    def hybrid_forward(self, F, hist, query, *args, **kwargs):
        embeddeding = kwargs["embeddeding"]
        query = F.reshape(query, shape=(0, 1))

        history = F.Embedding(data=hist, weight=embeddeding, \
                    input_dim=self.embedding_shape[0], output_dim=self.embedding_shape[1])
        query = F.Embedding(data=query, weight=embeddeding, \
                    input_dim=self.embedding_shape[0], output_dim=self.embedding_shape[1])

        simh = F.broadcast_mul(history, query)
        lstm = self.lstm(simh)
        lstm_pool = self.lstm_pool(lstm)
        softmax = F.exp(F.sum(lstm_pool,axis=2))
        softmax = F.broadcast_div(softmax, F.sum(softmax, axis=1, keepdims=True))
        attention_weights = F.reshape(softmax, shape=(0, 1, -1))
        aggregate_history = F.batch_dot(F.L2Normalization(attention_weights, mode='instance'),\
                                F.L2Normalization(history, mode='instance'))
        interaction = query * aggregate_history
        concat = F.concat(aggregate_history, interaction, query, dim=2)
        deep = self.deep1(concat)
        deep = F.concat(concat, deep, dim=2)
        deep2 = self.deep2(deep)
        out = F.reshape(deep2, shape=(0,))
        return out
```

- Written using (the very young) Gluon API

- Similar to Keras

- Can be compiled into a “Symbol” model (symbolic graph of execution)

Using in Scala a model trained in Python

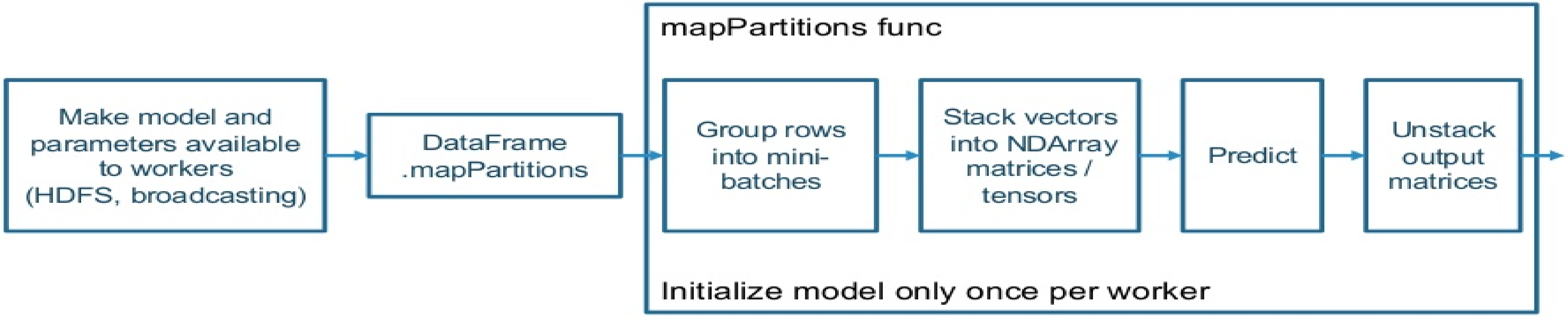
```
In [29]: model.export(epoch=40, path="s3://test-mde-us/mxnet/model")
```

```scala
val symbol = Symbol.load(symbol_path)
val params = NDArray.load2Map(parms_path)

/* ... */
val mod = new Module(symbol, dataNames = inFields, labelNames = IndexedSeq(),
                     fixedParamNames = Some(params.keySet))
mod.bind(shapes, labelShapes = None, forTraining = false)
mod.initParams(new ZeroInit(), argParams = params)

/* ... */
while (batchIter.hasNext) {
  val batch = batchIter.next()
  mod.forward(batch, Some(false))
}
```

Inference from a Spark pipeline



- Pay attention to resource usage on workers
  - Ensure backend processing will have available CPU cores and plenty of RAM
- Pay attention to data copy operation that will happen between Spark Vectors (from java memory) and MxNET native backend

Learnings about MxNET in Scala

- MxNET is very young
  - Mixed API styles
  - Scala API is lagging Python's
    - No sparse tensors yet
  - Unhelpful error messages, especially when they come from the C++ backend
  - In Lab context (notebook, fast iteration), it feels heavy
  - Some non-trivial boilerplate code to write
  - ONNX interop has limitations

- Yet, it's very promising, and fast out-of-the-box
  - Many binaries available with hw optimization
  - Usable in Scala/Spark pipelines

Take aways

- To find the “tiny clues” within your data, you need powerful learning algorithms to discover their latent meaning

- LSTM with Attention Models are especially good when it comes to sequences of events

- Using inference models from Scala/Spark code is easy and performant with MxNET, even if this framework has still a lot of limitations

Thank you !

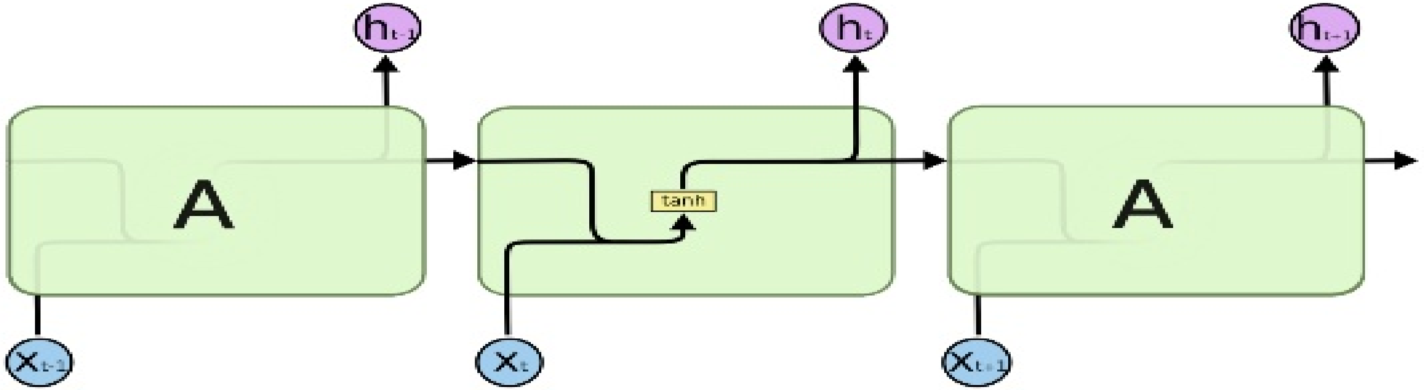
Questions ?

Reach me on twitter @mdespriee

Backup

Classical RNN have limitations

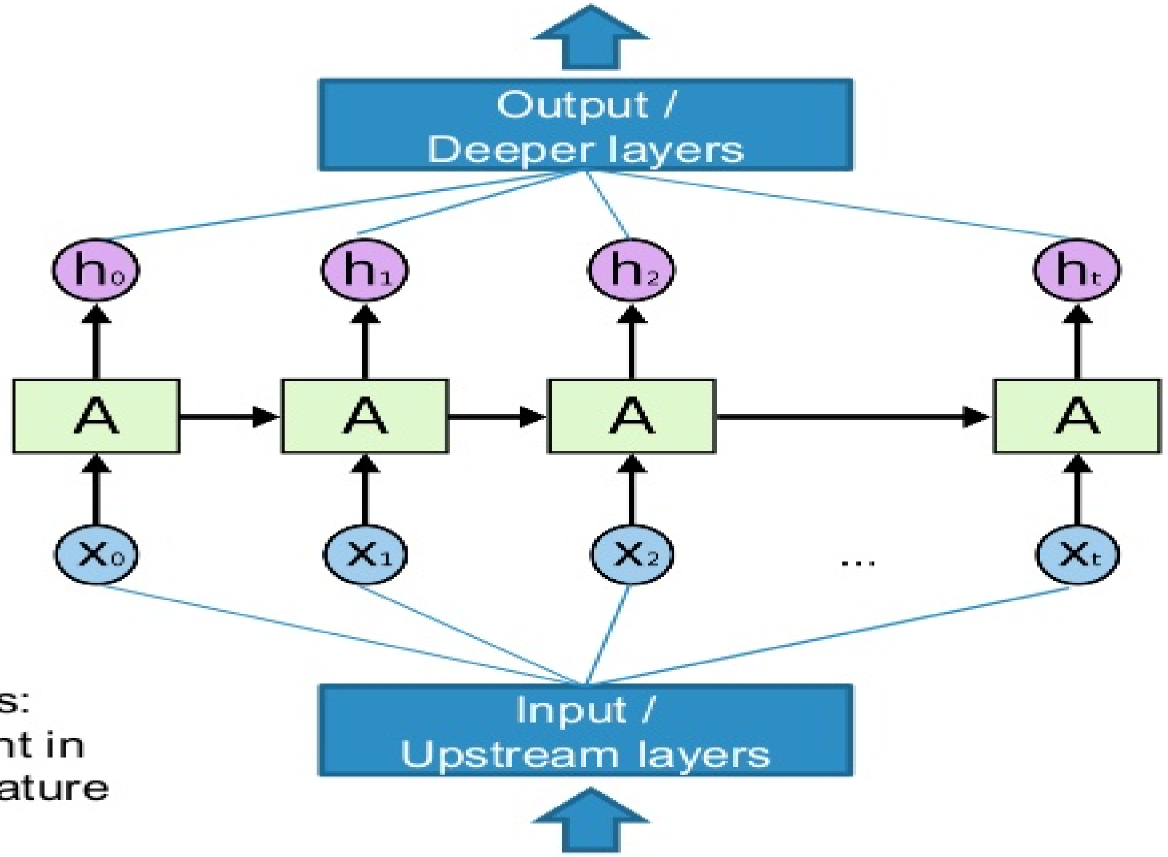
- Vanilla RNN can't learn long-term dependencies with gradient descent because of vanishing or exploding gradients



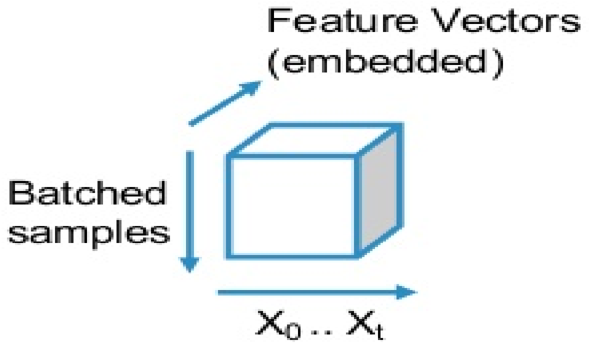
Illustrations by Christopher Olah (check out his blog!) http://colah.github.io/

RNN

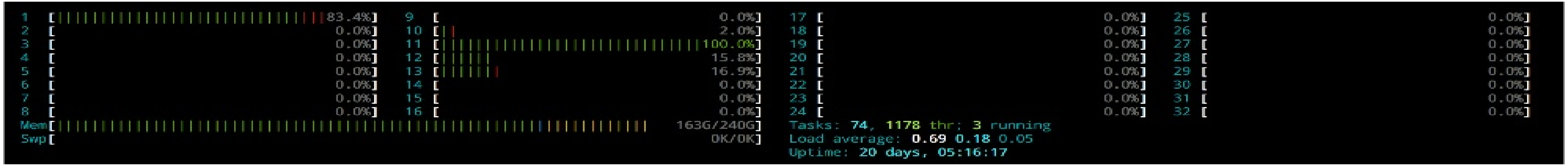
<u>"embedding"</u>: a compact mathematical representation of an information into a vector, capturing its semantics



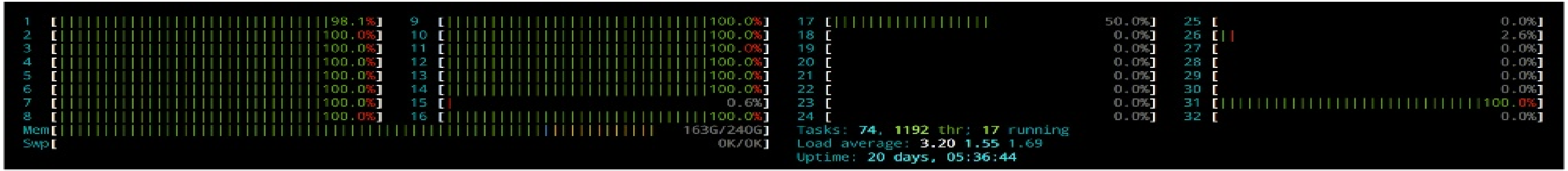
Input from upstream layers: Each X is a value at a point in time, represented by a Feature Vector



MKL impact

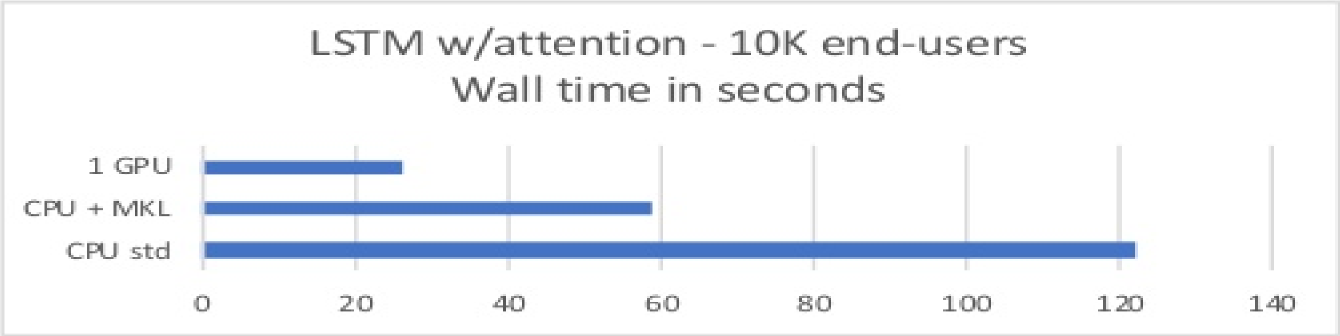


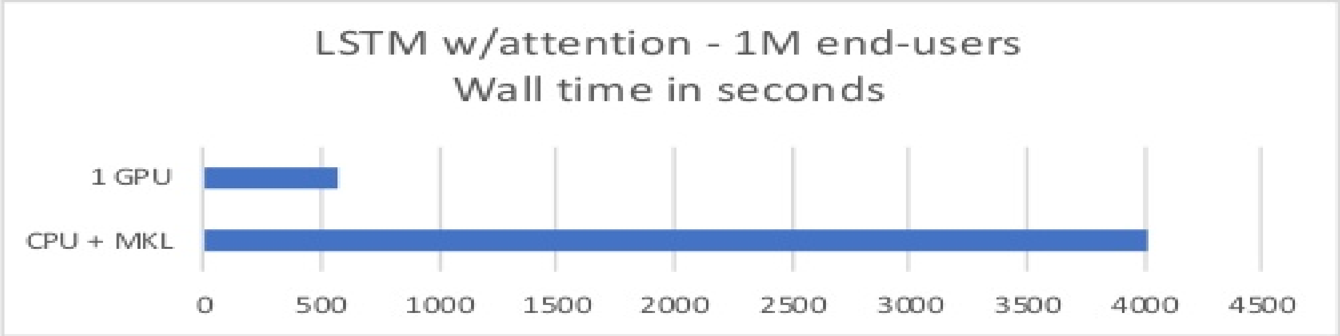
CPU times: user 4min 12s, sys: 10.4 s, total: 4min 22s Wall time: 2min 2s



CPU times: user 15min 41s, sys: 40.6 s, total: 16min 21s Wall time: 59.6 s

Performance of training





<u>Disclaimer</u>: These measurements are indicative, and vary a lot depending on many parameters.