--Contributions--

The paper describes StreamFlow, data pipeline to address industrial challenges for continuous learning over big data streams.

Authors propose an incremental algorithm to summarize high-velocity data.
They introduce a stream fusion technique to generate summaries for big industrial data.

The final result of the framework is a feature vector ready to use in machine learning tasks.

The system has been deployed in a banking system, processing billions of daily traffic operations.

Using summarized data generated by StreamFlow
results in up to 2 orders of magnitude speedups in training time
without compromising predictive performance.

--Data--

They used real-world
network traffic data streams from BNP Paribas to evaluate the
proposed approach in industrial settings.

Then, They collected telecommunication logs to learn the
behaviour of the traffic in two time periods
for a day and a week.

They obtained 20,000 samples for a day and
90,000 samples for a week.

--Related work--

We can see that there were previous works,
but none of them solved all problems streamflow solves.

--Challenges--

So what are the main challenges in Industrial Big Data.

Authors group them into three categories:
big data fusion,
summarization and the
interoperability of data processing tools.

Main issue for data fusion is the setup of streaming
pipelines to process, parse and structure unstructured log data.

Authors summarize big data in order to produce rich feature
vectors suited for artificial intelligence models.

Then there is interoperability.
The interoperability between big data tools and AI methods raises various data engineering issues
to satisfy real-time requirements such as latency, scalability and monitoring.

Finally, big industrial data is usually heterogeneous, so it
is critical to process this data properly before training.

--The operational pipeline

Streamflow has eight main data processing steps. (show them)

Depending on the big data application they used some or all of the steps.

1) Data ingestion: consists of collecting data across multiple sources
and databases in a batch and streaming fashion.

2) Data processing for indexing and storage:
the collected data need to be indexed and stored for further
efficient processing.

3) Data modelling and normalization: is needed to deal
with time conflicts, joins, and address queries across
heterogeneous data sources in a streaming or batch incremental fashion.

4) Data enrichment: aims to compute statistics for each
sliding window. They compute the aggregation function for
each feature to enrich the initial stream event with new
features (trend, means, slope, sum, etc.).

5) Online and incremental updates: for each feature,
sensor, and item, they update relevant information in a
knowledge base.

6) Summarizing: for each sliding window, they output a
vector summarizing data across a period.

7) Streams fusion: computes all the feature vectors across
all sources to get a final feature vector that will be used
to train the predictive model.

8) Data stream learning:
Apply machine learning to the the summaries
and feature vectors.

I'll now describe online summarization algorithm for machine learning
(steps 6 and 7 of our pipeline)

The first step of the algorithm consists of collecting streaming data from all the sources and selecting the set of relevant features for each item.

The second step aggregates streams over time for each feature.
For numerical features, they compute statistics
(minimum, maximum, average, standard deviation) over the current sliding window.
For categorical features, they compute counts and ratios over the current
sliding window for each category.

The third step consists of
merging at a given time t all aggregated streams computed
for all data sources before timestamps t.

Finally, the last step
is the fusion of all the feature vectors from each feature
computed during a given batch resulting in a single feature
vector representing a summary of all the input data streams.
Such a feature vector can be used in ML tasks or big data
mining use cases.

-- They focused on monitoring IT operations within a
banking information system. The data used are heterogeneous
logs collected from multiple data sources and different levels
of applications and processes (servers, networks, users, etc.).
We processed more than 21.5 terabytes of heterogeneous
log data, representing more than 50 billion transactions.
Every five minutes, an automated query is done
to update/train machine learning jobs continuously,
leading to the pics in search rate & latency.

For evaluation purposes, They measured how fast the system
can index and output search queries for information retrieval tasks.

-- In the upeer graph we can see that their solution considerably improves the
performance of a simple model such as Linear Regression

We can see elapsed time is linearly dependent the number of samples increases
by order of 10, the time to summarize 10 million of log events increases by order of 10.

In left graph, the orange line is higher because, in the 8 features case,
they have categorical features with many categories, each of them becoming a new
feature in the final feature vector (feature explosion).
For the 14 features case, the majority are numerical and involve a
simple calculation of their statistical mean.

We also see in right graph that the number of sources has no impact on the
algorithm's time complexity.

-- They tested both batch models (Linear Regression, Decision Trees, Random Forest)
and online/incremental machine learning methods (Adaptive Random Forest, Hoeffding Trees)
comparing the results Before StreamFlow (big raw data) vs

After StreamFlow (big data summaries obtained on a weekly and daily basis).

For metrics they used area under the
ROC curve and F1-score

So after the fusion and summarizing on sliding windows,
the resulting feature vector provides better or similar performance
while optimizing time for the majority of scenarios.

Their solution considerably improves the
performance of a simple model such as Linear Regression,
while for Random Forest, we obtain similar
results with our summarization having (156x speedup) for daily summarization,
and (36x speedup) for weekly summarization.

The system used a cluster of 21
machines.

Tools - Apache Kafka, Logstash pipelines.

F1-score - harmonic mean of precision and recall
2/(recall^-1 + precision^-1)

Precision = true positives/all positives
recall = true positives/(true positives + false negatives)

ROC plot true positive rate against false positive rate at various thresholds

We aim to train efficient online ML methods to classify log
events in real-time. Such a model processes each sample
(event streams) once, discards it, updates the model and makes
predictions continuously.