New Frontiers for Apache Spark

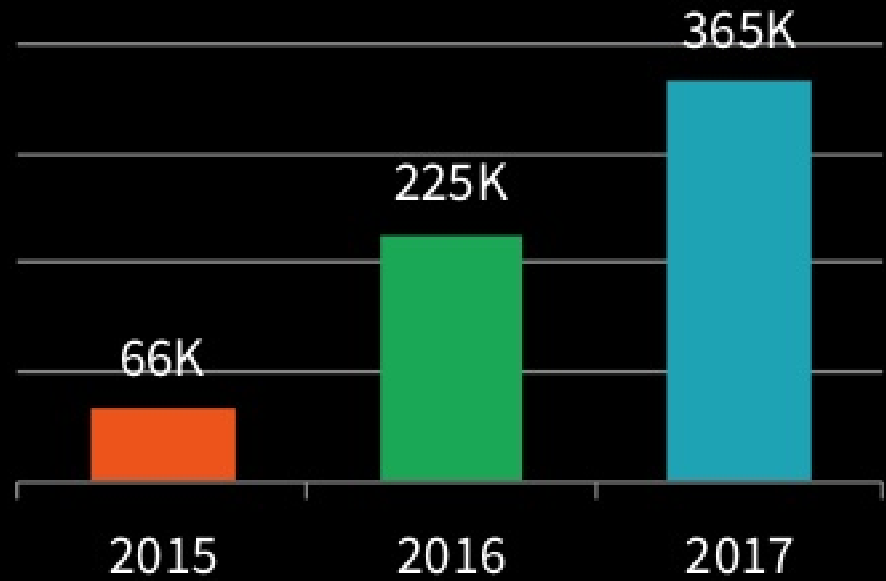
Matei Zaharia
@matei_zaharia

databricks

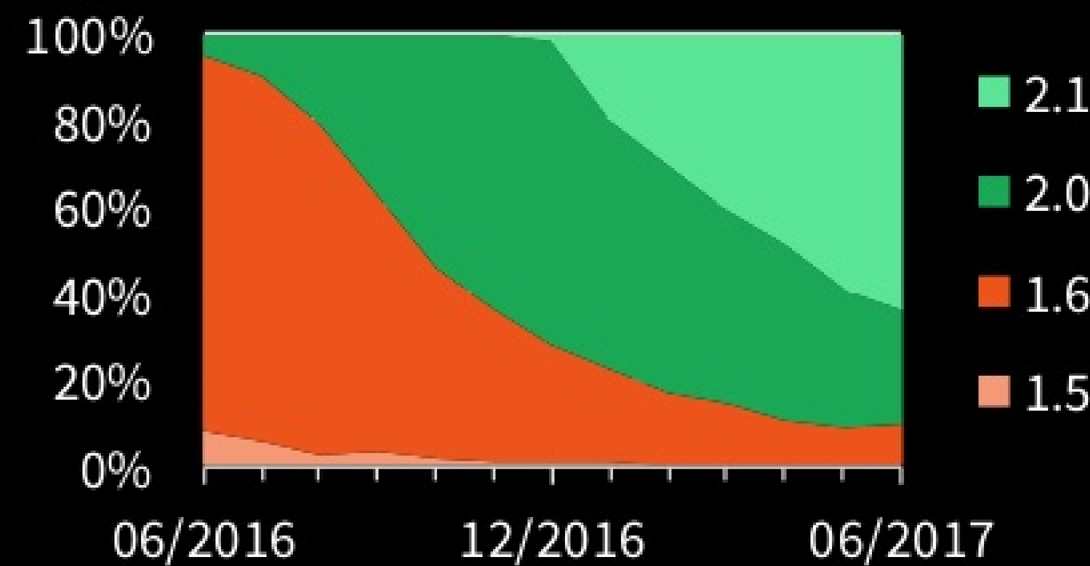
Welcome to Spark Summit 2017

Our largest summit, following another year of community growth

Spark Meetup Members Worldwide



Spark Version Usage in Databricks

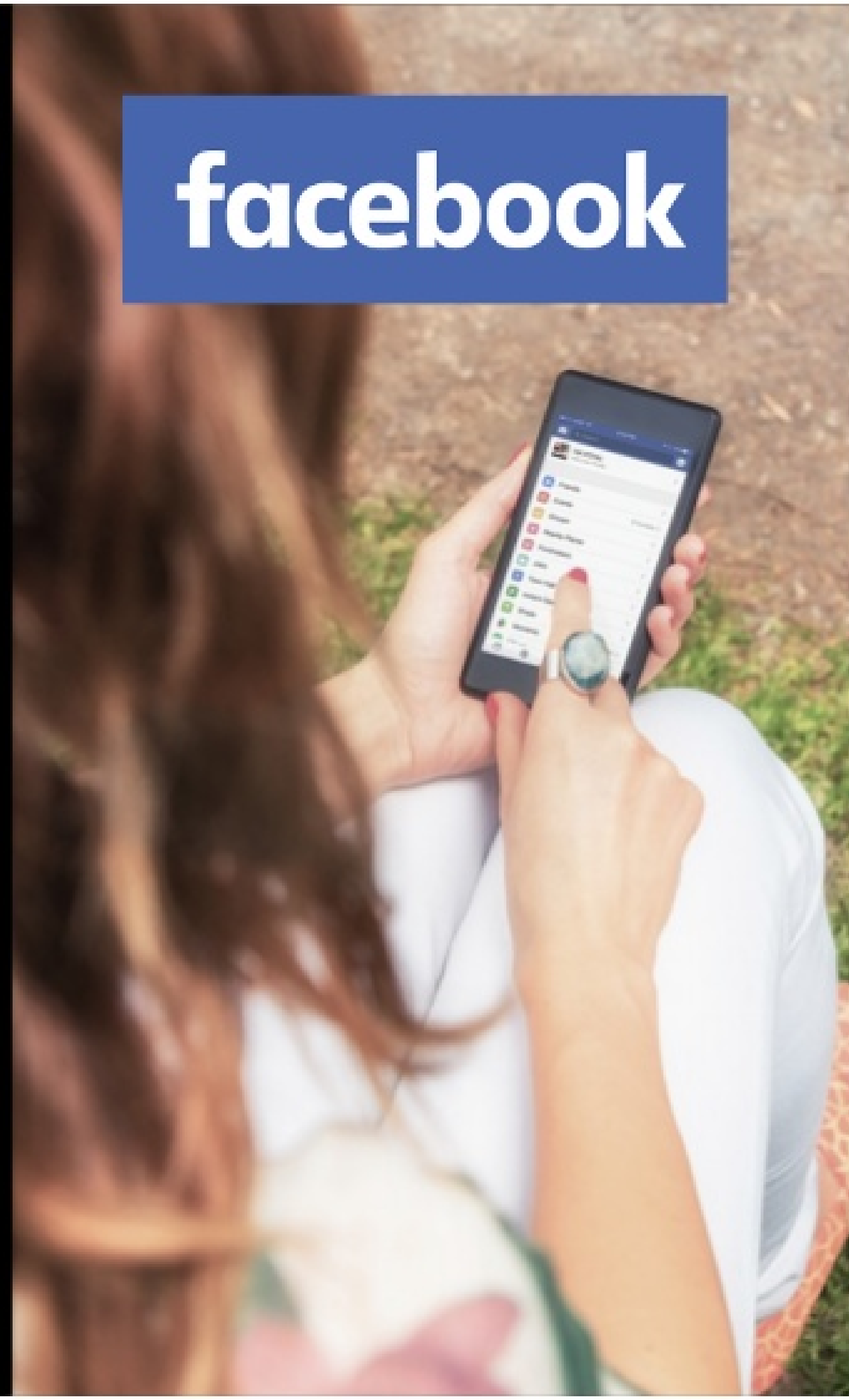


databricks

Summit Highlights

databricks


BROAD
INSTITUTE

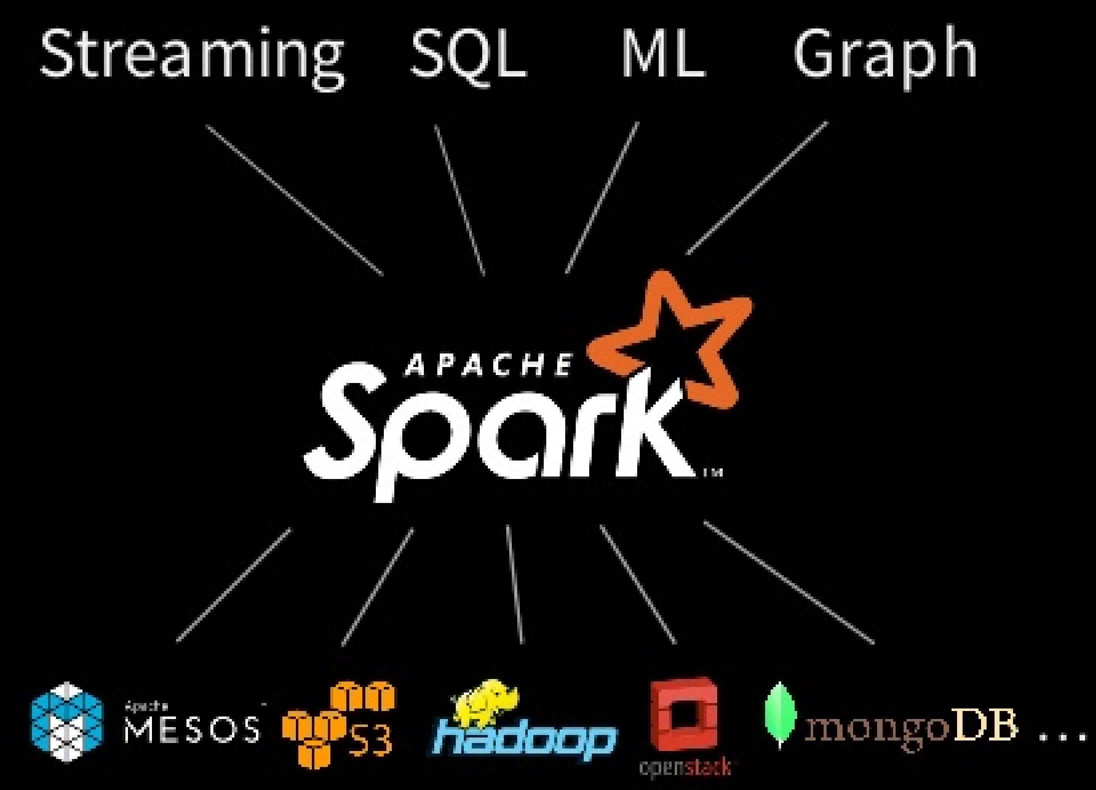

facebook


RIOT
GAMES

Apache Spark Philosophy

Unified engine for complete data applications

High-level user-friendly APIs



Coming in Spark 2.2

- Data warehousing: cost-based SQL optimizer
- Structured Streaming: marked production-ready
- Python usability: `pip install pyspark`

Currently in release candidate stage on dev list

databricks

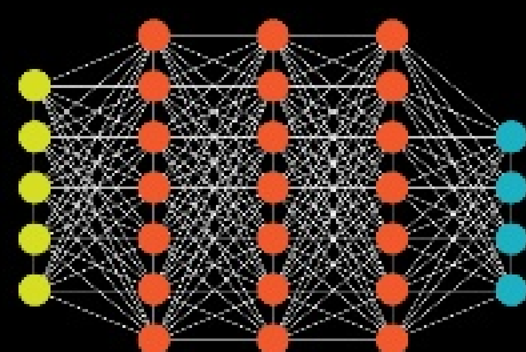
Two New Open Source Efforts from Databricks

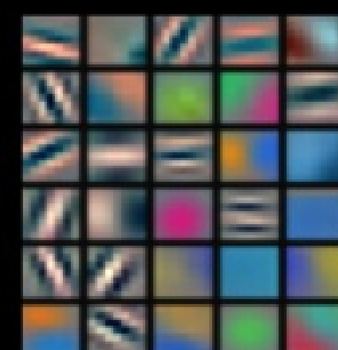
1. **Deep Learning**
2. **Streaming Performance**

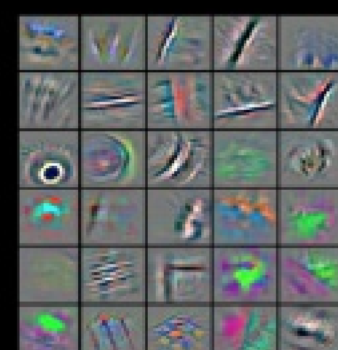
databricks

Deep Learning has Huge Potential

Unprecedented ability to work with unstructured data such as images and text







databricks

But Deep Learning is Hard to Use

Current APIs (TensorFlow, Keras, BigDL, etc) are low-level

- Build a computation graph from scratch
- Scale-out typically requires manual parallelization

Hard to expose models in larger applications

Very similar to early big data APIs (MapReduce)

Our Goal

Enable an order of magnitude more users to build applications using deep learning

Provide scale & production use out of the box

Deep Learning Pipelines

A new high-level API for deep learning that integrates with Apache Spark's ML Pipelines

- Common use cases in just a few lines of code
- Automatically scale out on Spark
- Expose models in batch/streaming apps & Spark SQL

Builds on existing engines (TensorFlow, Keras, BigDL)

databricks

Deep Learning Pipelines Demo

Tim Hunter - @timjhunter

Spark Summit 2017

Using Apache Spark and Deep Learning

- New library: Deep Learning Pipelines
- Simple API for Deep Learning, integrated with ML pipelines
- Scales common tasks with transformers and estimators
- Embeds Deep Learning models in Spark

Example:
Image classification

databricks

Example: Identify the James Bond cars

databricks

Example: Identify the James Bond cars

007

007

databricks

Good application for Deep Learning

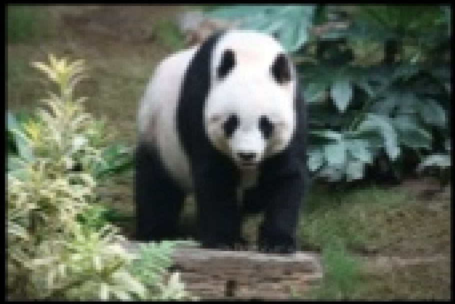
- Neural networks are very good at dealing with images
- Can work with complex situations:

INVARIANCE TO ROTATIONS

INCOMPLETE DATA

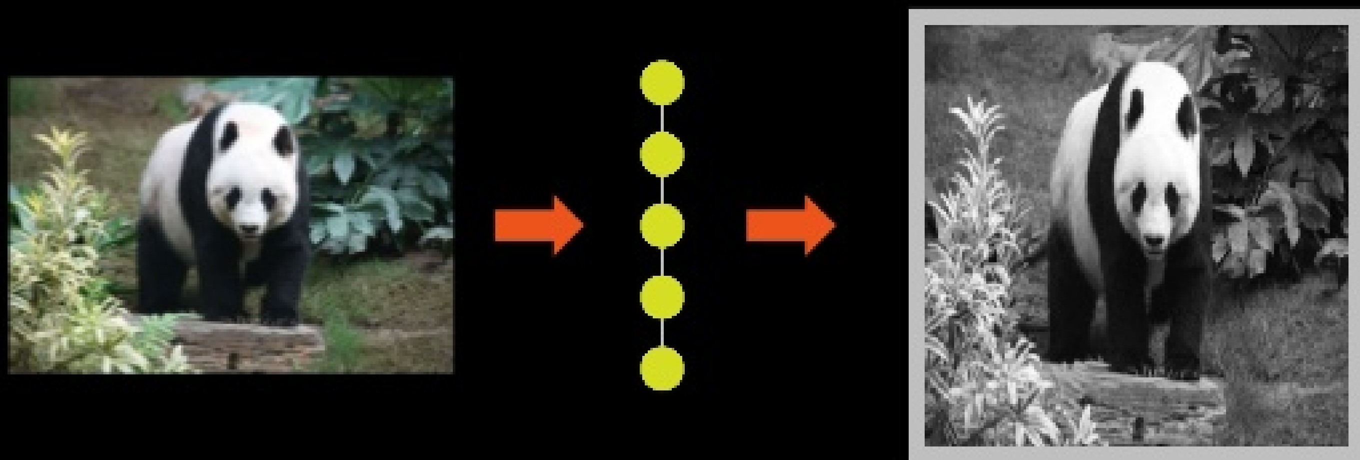
databricks

Transfer Learning



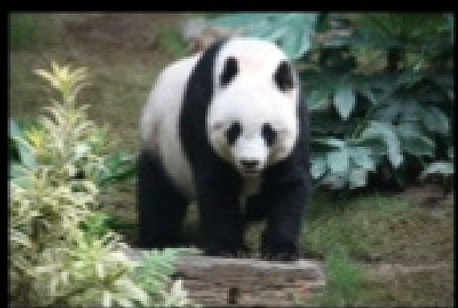
databricks

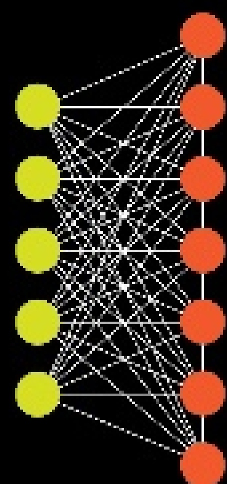
Transfer Learning



databricks

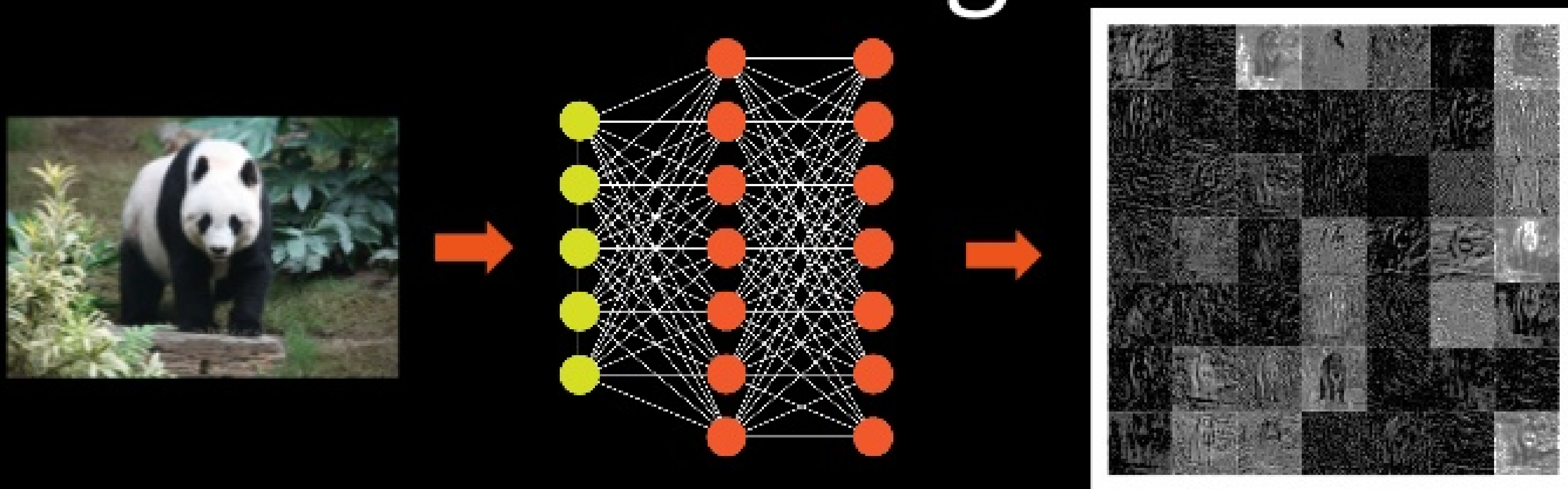
Transfer Learning



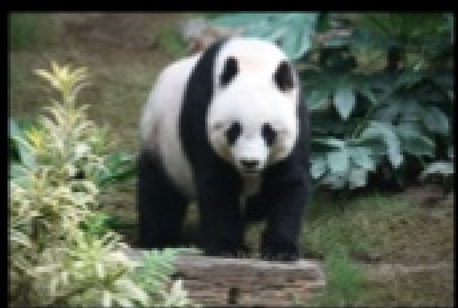


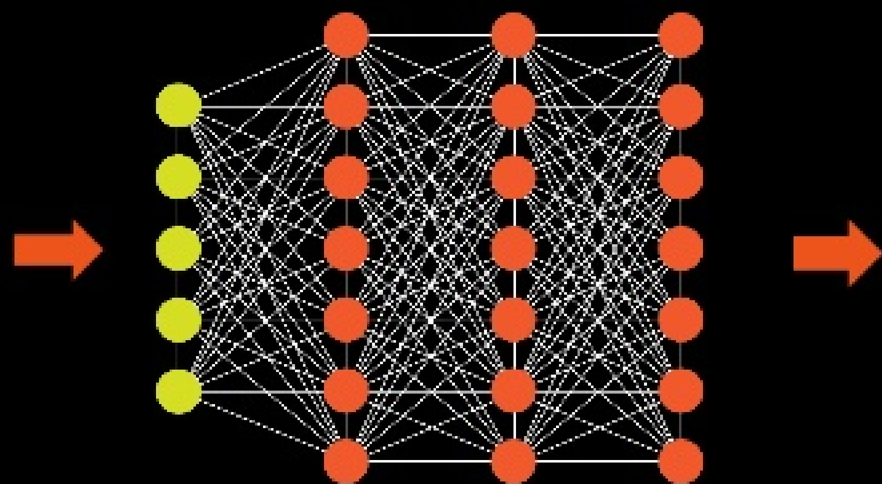
databricks

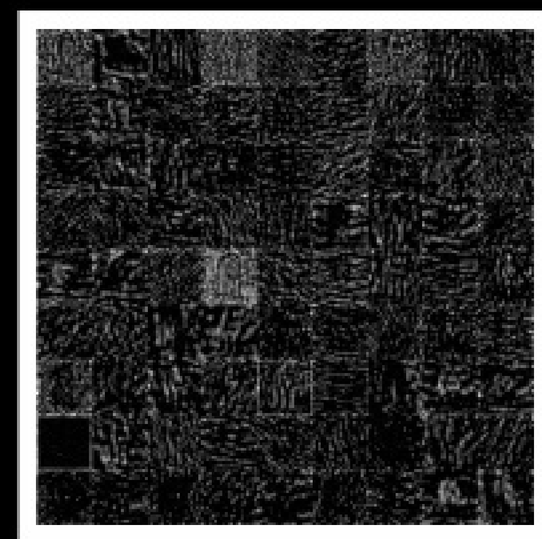
Transfer Learning



Transfer Learning

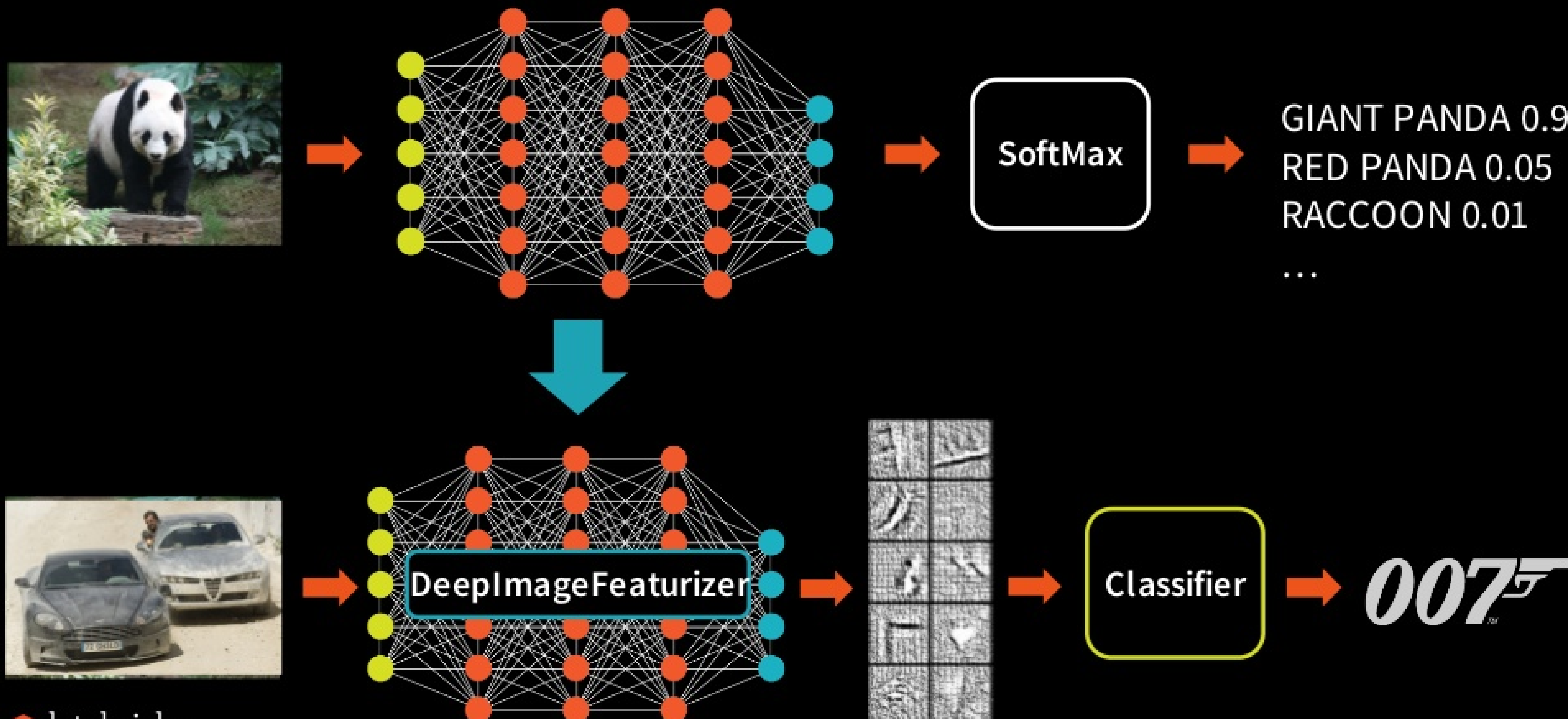






databricks

Transfer Learning



databricks

Deep Learning without Deep Pockets

- Simple API for Deep Learning, integrated with MLlib
- Scales common tasks with transformers and estimators
- Embeds Deep Learning models in MLlib and SparkSQL
- Early release of Deep Learning Pipelines
  https://github.com/databricks/spark-deep-learning

databricks

Two New Open Source Efforts from Databricks

1 **Deep Learning**

2 **Streaming Performance**

Structured Streaming: Ready For Production

Michael Armbrust - @michaelarmbrust
Spark Summit 2017

databricks

What is Structured Streaming?

Our goal was to build the *easiest streaming engine* using the power of Spark SQL.

- **High-Level APIs** — DataFrames, Datasets and SQL. Same in streaming and in batch.
- **Event-time Processing** — Native support for working with out-of-order and late data.
- **End-to-end Exactly Once** — Transactional both in processing and output.

Simple APIs: YAHOO! Benchmark

Apache Kafka streams

Filter by click type and project

```
KStream<String, ProjectedEvent> filteredEvents = kEvents.filter((key, value) -> {
  return value.event_type.equals("view");
}).mapValues((value) -> {
  return new ProjectedEvent(value.ad_id, value.event_time);
});
```

Join with campaigns table

```
KTable<String, String> kCampaigns = builder.table("campaigns", "campaign-state");
KTable<String, CampaignAd> deserCampaigns = kCampaigns.mapValues((value) -> {
  Map<String, String> campMap = Json.parser.readValue(value);
  return new CampaignAd(campMap.get("ad_id"), campMap.get("campaign_id"));
});
KStream<String, String> joined =
  filteredEvents.join(deserCampaigns, (value1, value2) -> {
    return value2.campaign_id;
  },
  Serdes.String(), Serdes.serdeFrom(new ProjectedEventSerializer(),
  new ProjectedEventDeserializer()));
```

Group and windowed count

```
KStream<String, String> keyedByCampaign = joined.selectKey((key, value) -> value);
KTable<Windowed<String>, Long> counts = keyedByCampaign.groupByKey()
  .count(TimeWindows.of(10000), "time-windows");
```

Simple APIs: YAHOO! Benchmark

Apache Spark DataFrames

```
events
  .where("event_type = 'view'")
  .join(table("campaigns"), "ad_id")
  .groupBy(
    window('event_time, "10 seconds"),
    'campaign_id)
  .count()
```

Apache kafka streams

```
KStream<String, ProjectedEvent> filteredEvents = kEvents.filter((key, value) -> {
  return value.event_type.equals("view");
}).mapValues((value) -> {
  return new ProjectedEvent(value.ad_id, value.event_time);
});

KTable<String, String> kCampaigns = builder.table("campaigns", "campaign-state");
KTable<String, CampaignAd> deserCampaigns = kCampaigns.mapValues((value) -> {
  Map<String, String> campMap = Json.parser.readValue(value);
  return new CampaignAd(campMap.get("ad_id"), campMap.get("campaign_id"));
});
KStream<String, String> joined =
  filteredEvents.join(deserCampaigns, (value1, value2) -> {
    return value2.campaign_id;
  },
  Serdes.String(), Serdes.serdeFrom(new ProjectedEventSerializer(),
  new ProjectedEventDeserializer()));

KStream<String, String> keyedByCampaign = joined.selectKey((key, value) -> value);
KTable<Windowed<String>, Long> counts = keyedByCampaign.groupByKey()
  .count(TimeWindows.of(10000), "time-windows");
```

Simple APIs: YAHOO! Benchmark

Spark SQL

```
SELECT COUNT(*)
FROM events
JOIN campaigns USING ad_id
WHERE event_type = 'view'
GROUP BY
    window(event_time, "10 seconds"),
    campaign_id)
```

kafka streams

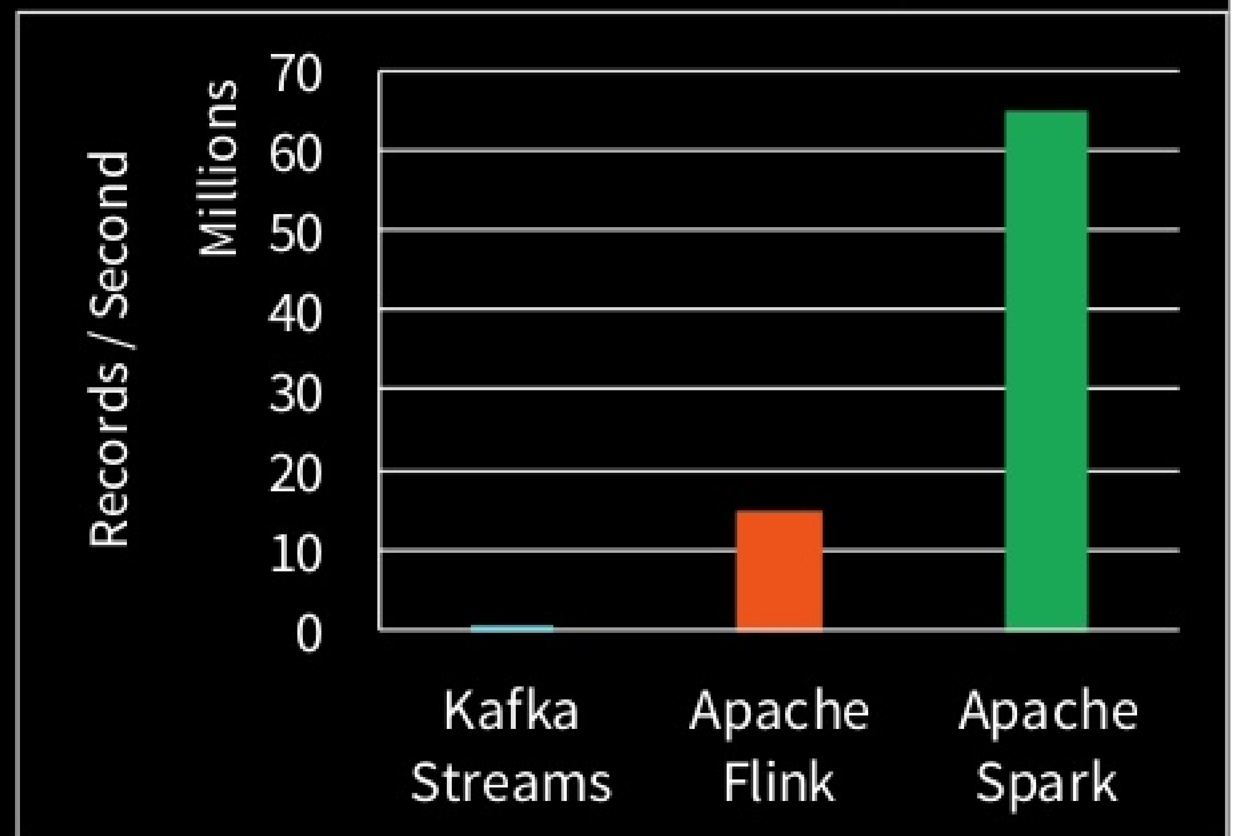
```
KStream<String, ProjectedEvent> filteredEvents = kEvents.filter((key, value) -> {
  return value.event_type.equals("view");
}).mapValues((value) -> {
  return new ProjectedEvent(value.ad_id, value.event_time);
});

KTable<String, String> kCampaigns = builder.table("campaigns", "campaign-state");
KTable<String, CampaignAd> deserCampaigns = kCampaigns.mapValues((value) -> {
  Map<String, String> campMap = Json.parser.readValue(value);
  return new CampaignAd(campMap.get("ad_id"), campMap.get("campaign_id"));
});
KStream<String, String> joined =
  filteredEvents.join(deserCampaigns, (value1, value2) -> {
    return value2.campaign_id;
  },
  Serdes.String(), Serdes.serdeFrom(new ProjectedEventSerializer(),
  new ProjectedEventDeserializer()));

KStream<String, String> keyedByCampaign = joined.selectKey((key, value) -> value);
KTable<Windowed<String>, Long> counts = keyedByCampaign.groupByKey()
  .count(TimeWindows.of(10000), "time-windows");
```

Performance: YAHOO! Benchmark

Streaming queries use the **Catalyst Optimizer** and the **Tungsten Execution Engine.**

5x
lower cost

Throughput
At ~200ms Latency



databricks

https://data-artisans.com/blog/extending-the-yahoo-streaming-benchmark

GA in Spark 2.2

- Processed over ***3 trillion*** rows last month in production
- More production use cases of Structured Streaming than DStreams among Databricks Customers
- Spark 2.2 removes the “Experimental” tag from all APIs

databricks

What about Latency?

databricks

Demo: Streaming Find James Bond

- Stream of events containing images
- Need to join with locations table and filter using ML
- Alert **quickly** on any suspicious Bond sightings…

Need < 10ms to catch him!

databricks

Continuous Processing

A new execution mode that allows fully pipelined execution.

- Streaming execution ***without microbatches***
- Supports async checkpointing and ~1 ms latency
- No changes required for user code

Proposal available at https://issues.apache.org/jira/browse/SPARK-20928

Apache Spark Structured Streaming

The easiest streaming engine is now also the fastest!

databricks

Conclusion

We're bringing two new workloads to Apache Spark:

- Deep learning
- Low-latency streaming

Find out more in the sessions today!

Thanks

Enjoy Spark Summit!

databricks