skymind

DEEP LEARNING FOR ENTERPRISE

Speaker: **Adam Gibson**
adam@skymind.io

Network Intrusion with Deep Learning

- **Brief overview of Network Intrusion**
- **Anomaly Detection in general**
- **Alternative example as case study**
- **Why Deep Learning for this? Isn’t it overkill?**
- **Example walk through**
- **Demo**
- **End**

Brief overview of Network Intrusion

- **Network Intrusion: Detect the hackers. Finding malicious traffic.**
- **Malicious traffic attempting to compromise servers for financial gain or other reasons**
- **Traffic could be internal or external**
- **Bad actors exists in and outside your network**

Anomaly Detection in General

- **Anomaly Detection: Finding a rare pattern among normal patterns**
- **These “patterns” are defined as “any activity over time”**
- **Other examples include credit card fraud, identity theft, or broken hardware**

Simbox fraud for telco

- Costs telco over 3 billion yearly
- Route calls for free over a carrier network
- Need to mine raw call detail records to find
- Find and cluster fraudulent CDRs with autoencoders (unsupervised)
- Beats current rules and supervised based approaches

orange™

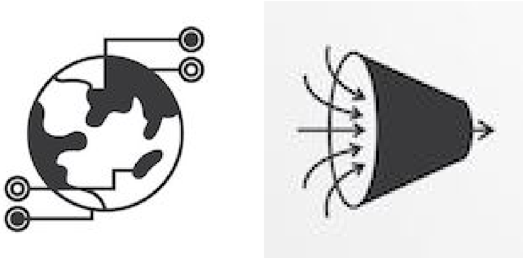
Why Deep Learning for this? Isn’t it overkill?

- Deep Learning is good at surfacing patterns in behavior
- We look at the billions of dollars of R&D being spent by adtech to surface *your* behavior to target you with ads
- Your behavior can just be “tends to watch cat pictures on youtube at 2pm” - google wants to know that
- We typically think of Deep Learning for media

Example walk through

We start with a real-time data source: logs streaming in from Wifi, Ethernet, the Network...

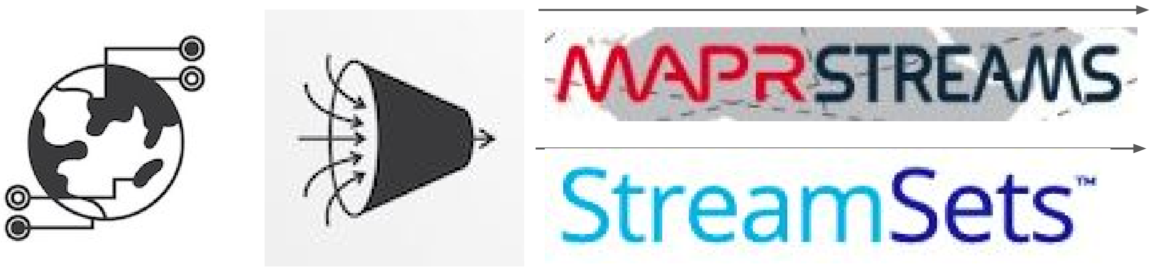
Data Source



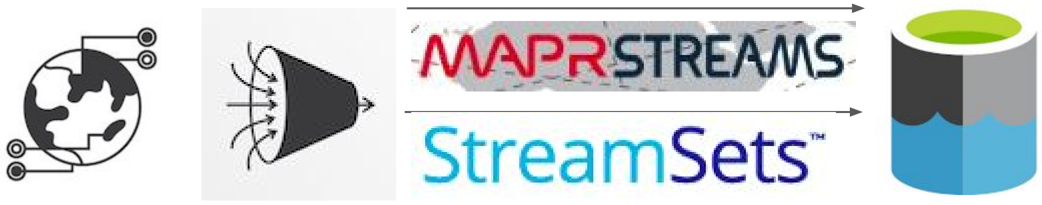
An NGINX web server is recording HTTP requests and sending them straight to a stream such as streamsets or kafka

Data Source

Data Spout



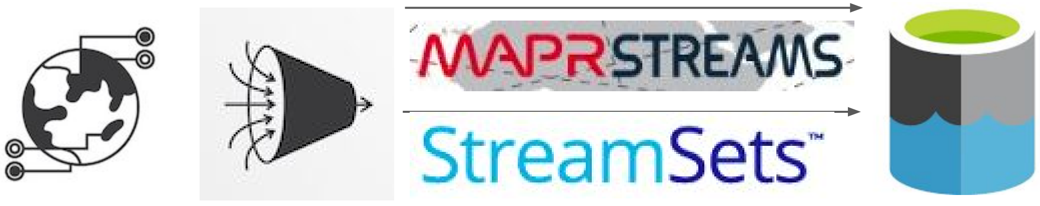
The data lands in a data lake, a unified file system like MapR's NFS or HDFS, where it is available for analytics.



Data Source

Data Spout

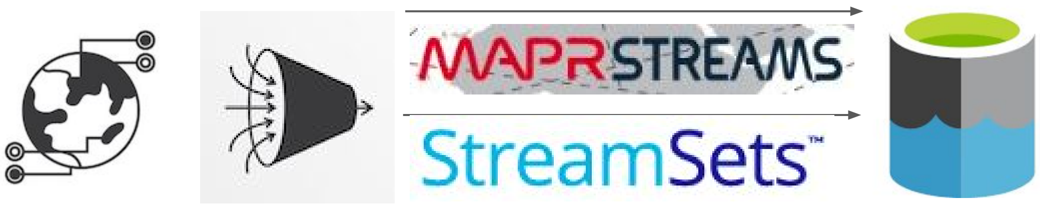
At this step, we can begin preprocessing the raw data. Up to and including this step, everything has run on CPUs, because they involve a constant stream of small operations.



Data Source

Data Spout

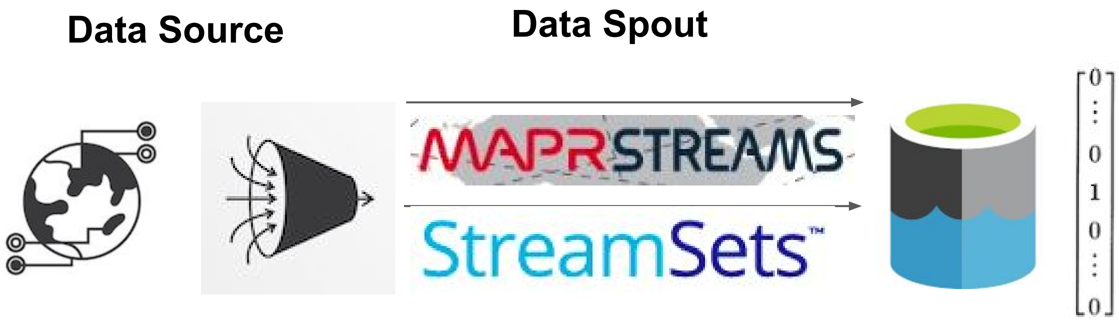
Preprocessing the data includes steps like normalization; standardizing diverse data to a uniform format; shuffling log columns; running find and replace; and translating log elements to a binary format.



Data Source

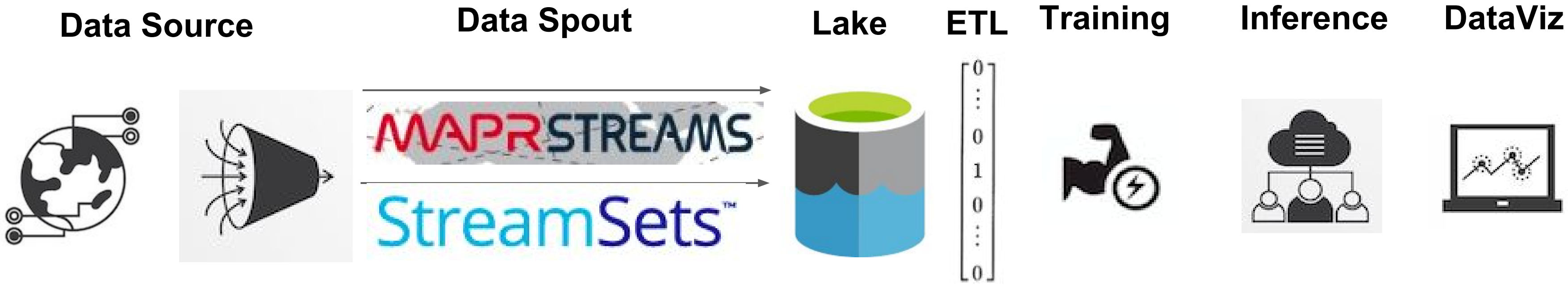
Data Spout

From there, the vectorized logs are fed into a neural net in batches, and each batch is dispatched to a different neural net model on a distributed cluster.



In this case, we've trained a neural net to detect network intrusion anomalies as part of a project with Canonical.

The last three stages, and above all training, run on GPUs.



Data Source

Data Spout

Lake

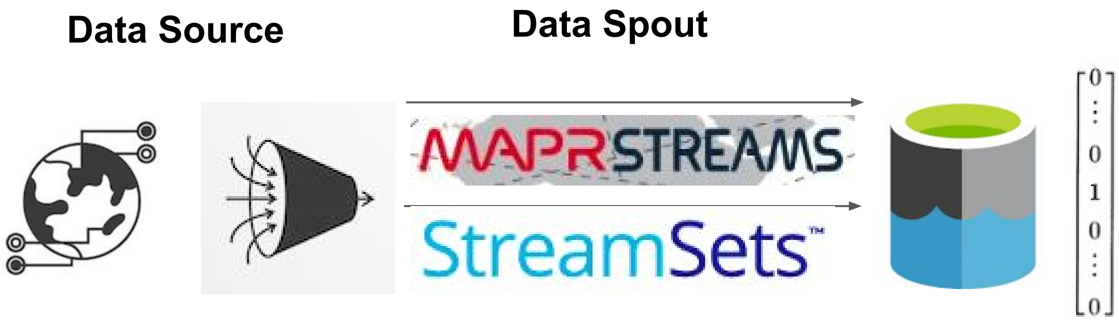
ETL

Training

Inference

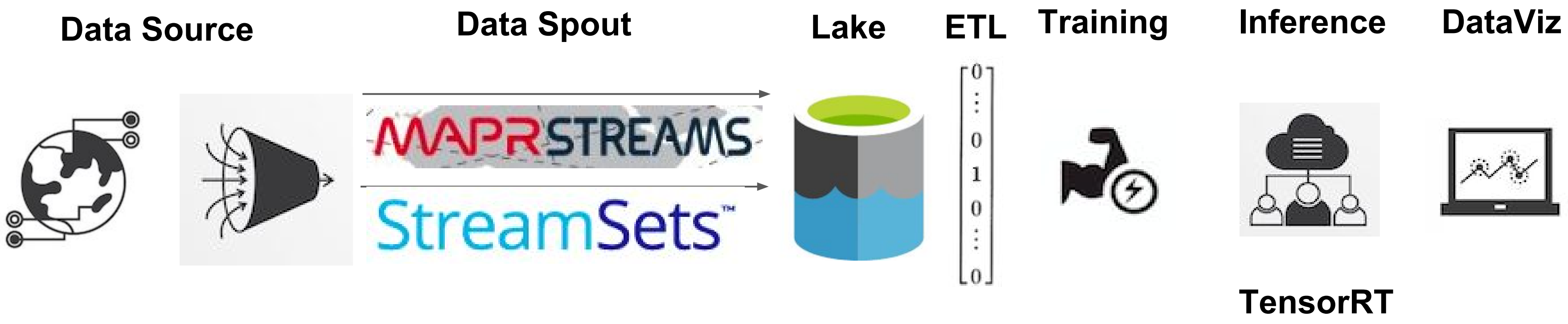
DataViz

From there, the vectorized logs are fed into a neural net in batches, and each batch is dispatched to a different neural net model on a distributed cluster.



In this case, we've trained a neural net to detect network intrusion anomalies as part of a project with Canonical.

New data can be fed to the trained models from multiple sources via Kafka/MAPR Streams or via a REST API. All connected to TensorRT.



Data Source

Data Spout

Lake

ETL

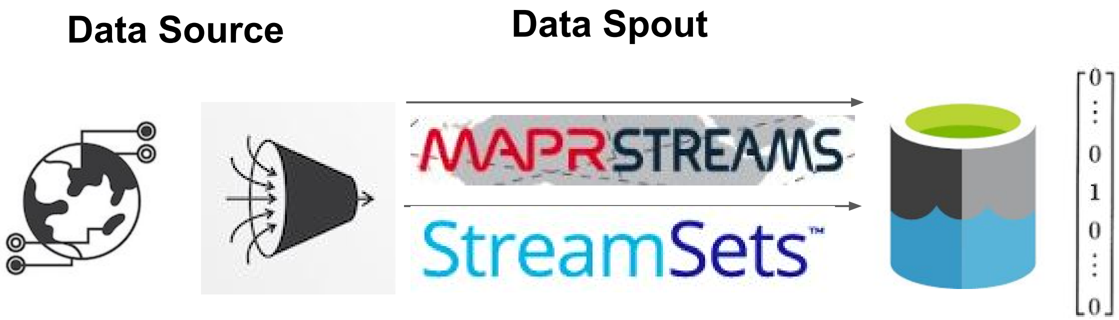
Training

Inference

DataViz

TensorRT

From there, the vectorized logs are fed into a neural net in batches, and each batch is dispatched to a different neural net model on a distributed cluster.



In this case, we've trained a neural net to detect network intrusion anomalies as part of a project with Canonical.

For example, web logs record the user's browser: Internet Explorer, Google Chrome or Mozilla Firefox. These three browsers have to be represented with numbers in a vector.

The method of representation is called one-hot encoding, where a data scientist will use three digits and call IE 100, Chrome 010 and Firefox 001. Which is how browsers are engineered to become numerical features that can be monitored for significance.

They will be added to a vector that contains other information about users, such as the region in which their IP address is located. Geographic regions, too, are converted into binary and stacked on top of other log elements.

Over the course of training, neural nets learn how significant or irrelevant those features are for the detection of anomalies and assign them weights.

Demo

THANK YOU