Master Thesis Final Presentation

Leveraging Deep Learning Solutions for Predictive Maintenance of Industrial Datasets

Rupam Bhattacharya

Under the supervision of :
Prof. Dr. Jan Křetínský

Fakultät für Informatik
Technische Universität München

17.11.2017

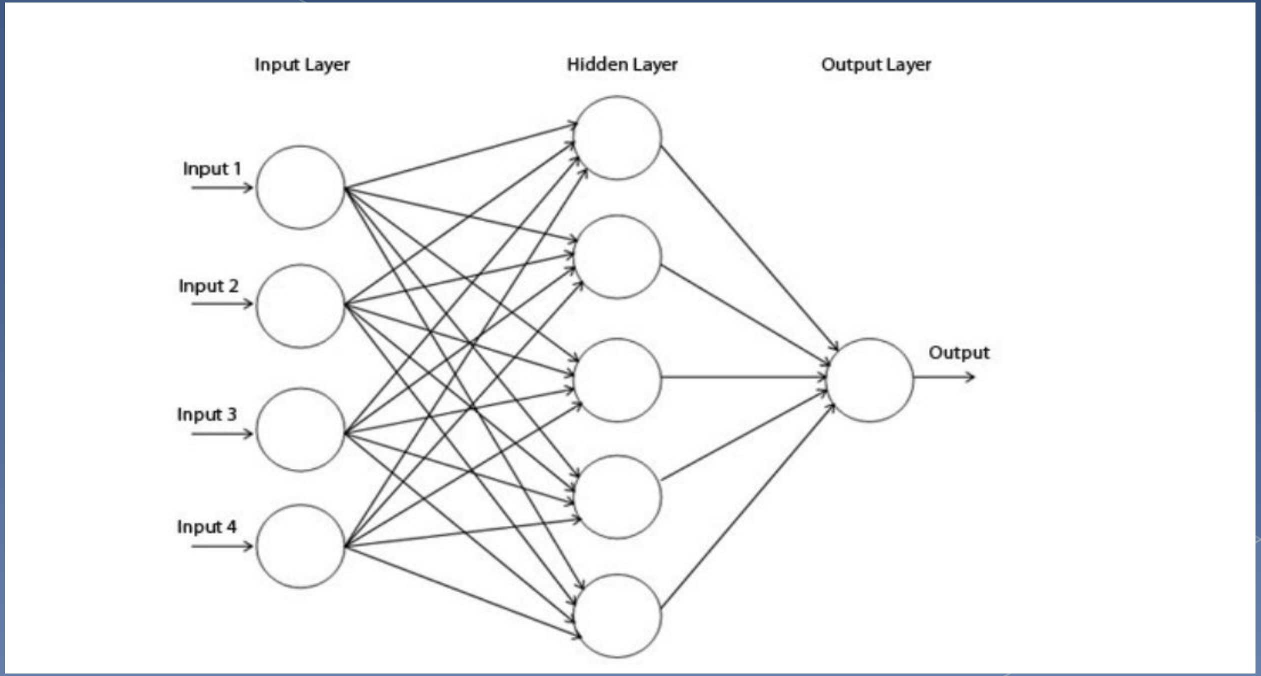
AGENDA

Introduction - Deep Learning

- **Hierarchical method of learning representations or abstractions of data**
- **Recent upsurge in popularity of deep learning networks : scale of data and scale of computation**

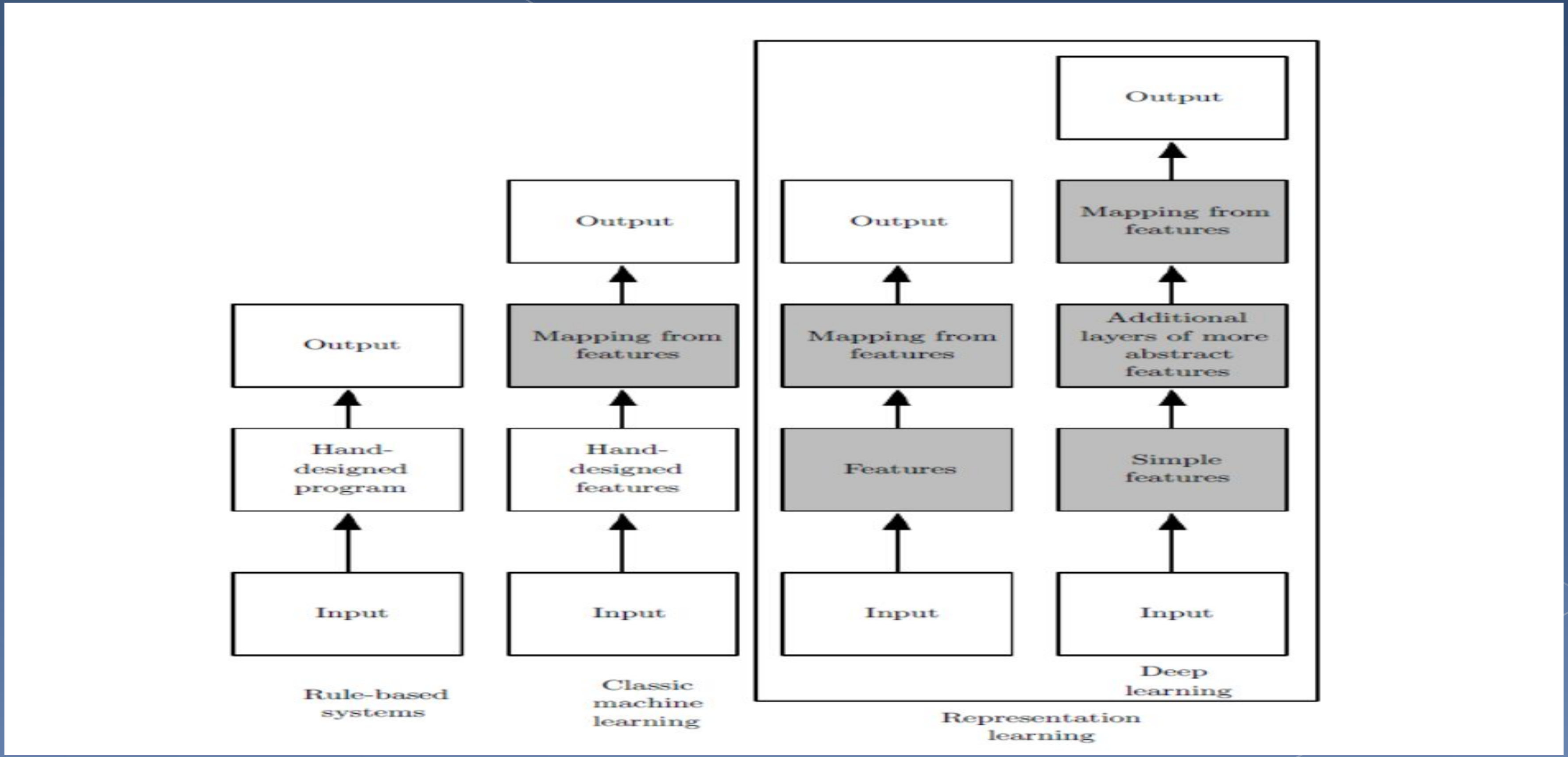
- Patterns presented to the network via the input layer, which communicates to one or more hidden layers

- The hidden layers then link to an output layer where the answer is output



Introduction - Deep Learning

Three types of Machine Learning algorithms - Supervised Learning, Unsupervised Learning and Reinforcement Learning



Introduction - Predictive Maintenance

Industry 4.0 and the Internet of Things

- Control and regulate entire production systems in a holistic and dynamic manner
- High reliability and availability
- Identification of Defective components that could soon lead to a system shutdown
- On-board solutions and off-board solutions for Predictive Maintenance

Introduction - Predictive Maintenance

- On-board solutions - computationally inexpensive, small memory footprint
- Off-board solutions - larger computational resources and access to historical data
- Commonly used approaches for predictive maintenance - Remaining Useful Life Prediction, Deviation Detection and Supervised Classification
- Deviation Detection easiest of all

Project Objectives

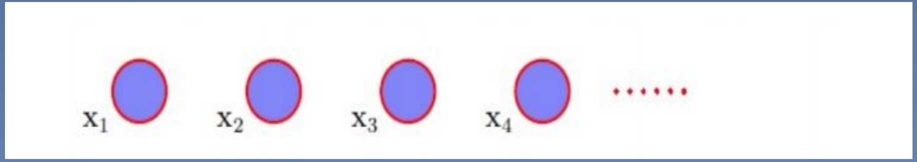
- Provide solutions for providing predictive maintenance of batteries in industrial machines, in a way that both maintenance and repair activities can be predicted beforehand

- Leverage deep learning solutions by exploring the application of sequence learning to represent the failure profiles and understand the patterns that lead to failures in batteries

Fundamentals

Sequence Learning

- Sequential data in real life includes video, speech, stock market, sensor data, and so on
- Sequences have complex temporal dependencies and a lot of hidden information
- Problem with sequential data - what time instant affects future outcomes and how far to look for patterns
- Sequential data cannot be always identically independent distribution as the assumption of statistical independence is not enough
- Simplest way to model a sequence of observations is to treat them as independent, corresponding to graph without links



Fundamentals

Optimization Algorithms

- Used to minimize (or maximize) a Loss function i.e. a mathematical function dependent on the prediction model’s ability to learn to compute the target values from the set of predictors

- Two types of optimization algorithms - First order optimization algorithms minimize or maximize a Loss function using its Gradient values with respect to the parameters and Second-order optimization algorithms use the second order derivative which is also called Hessian to minimize or maximize the Loss function

- Gradient Descent - most important technique in training and optimizing neural networks. The parameters of the prediction model are updated and tuned in a way to minimize the Loss function

- Backpropagation used commonly, in which the dot product of input signals and their corresponding weights are calculated first.Then an activation function is applied to those sum of products, which transforms the input signal to an output signal

Fundamentals

Optimization Algorithms

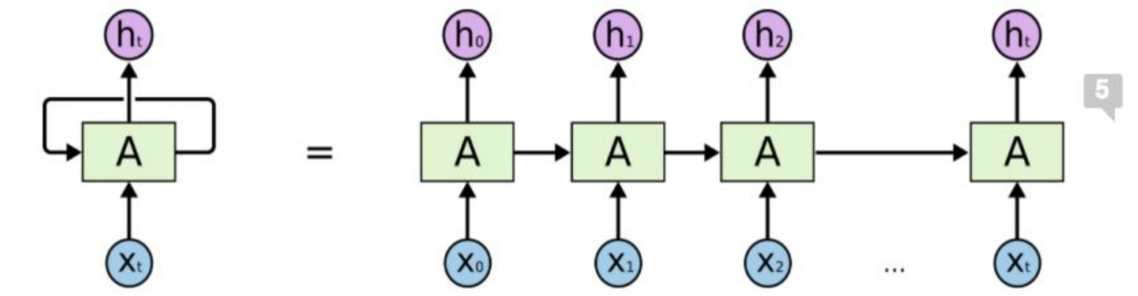
- Gradient Descent calculates the gradient of the whole data set, it performs only one update - slow and hard to control
- Gradient Descent Variants - Stochastic Gradient Descent and Mini Batch Gradient Descent
- Stochastic Gradient Descent faster. Problem - complicates convergence due to frequent updates
- Mini-batch gradient Descent takes the best out of both and frequently used
- Gradient Descent further optimized with various algorithms:
  - Momentum
  - Nesterov accelerated gradient
  - Adagrad
  - AdaDelta
  - Adaptive Moment Estimation (Adam)

- Adam - Most efficient of all the algorithms. In addition to storing an exponentially decaying average of past squared gradients like AdaDelta, Adam also keeps an exponentially decaying average of past gradients, similar to momentum

Fundamentals

Recurrent Neural Networks

- Neural networks which have loops in them allowing information to persist
- Recurrent nets allow to operate over sequences of vectors: Sequences in the input, the output, or in the most general case both.
- RNNs combine the input vector with their state vector with a fixed (but learned) function to produce a new state vector



An unrolled recurrent neural network.

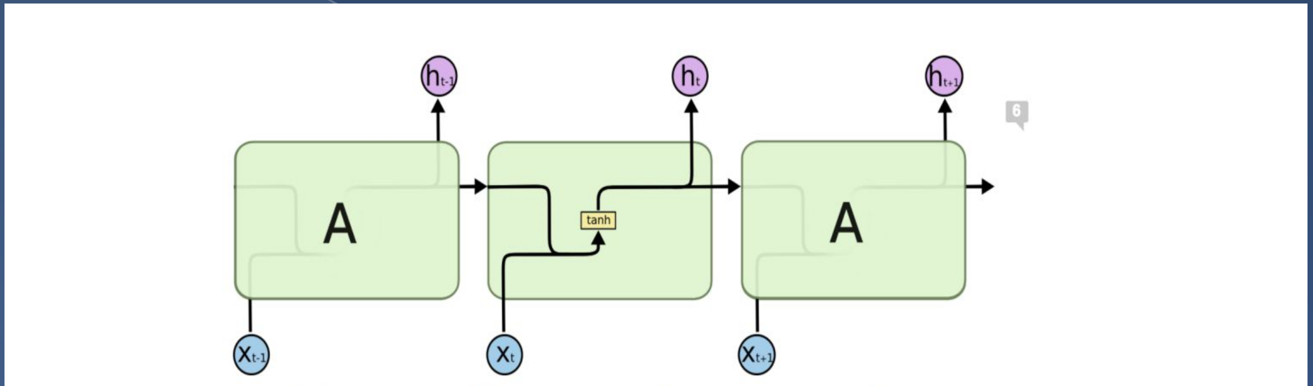
Fundamentals

Long Short Term Memory (LSTM)

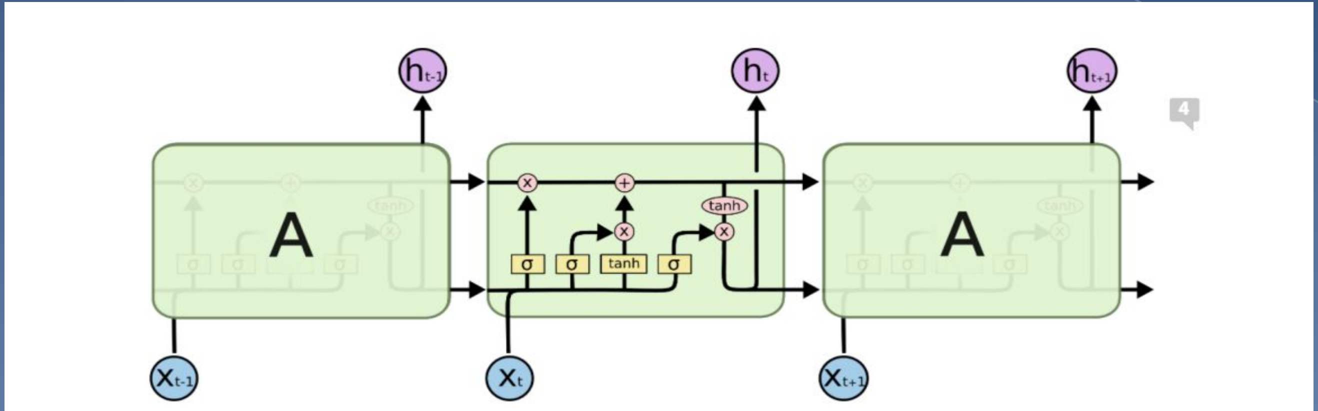
- The central idea behind the LSTM architecture is a memory cell which can maintain its state over time, and nonlinear gating units which regulate the information flow into and out of the cell
- LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates
- Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation
- An LSTM has three of these gates, to protect and control the cell state.
- LSTMs are effective at capturing long-term temporal dependencies without suffering from the optimization hurdles that plague simple recurrent networks (SRNs)

Fundamentals

Long Short Term Memory (LSTM)



The repeating module in a standard RNN contains a single layer.



The repeating module in an LSTM contains four interacting layers.

Data Set & Tools

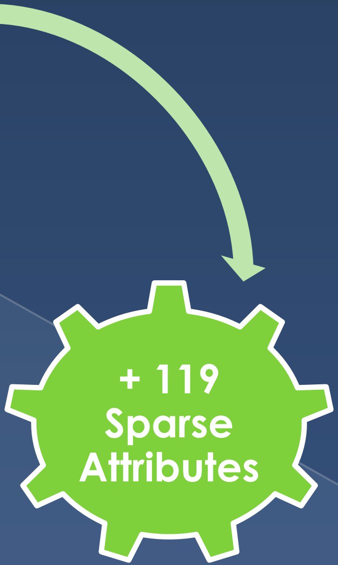
The Data Set

- Battery dataset for calculating remaining useful life of aero engines was considered which contains failure events for about half a million events from last three years
- Batteries continuously cycled with randomly generated current profiles
- Reference charging and discharging cycles are also performed after a fixed interval of randomized usage in order to provide reference benchmarks for battery state of health
- Size ~ 9 GB

WHAT is described by the data ?

Battery data set (~ 4 GB)
931 attributes

- timestamp
- battery type
- state of charge
- historical state of charge
- stat aging insistence value
- stat aging capacity value
- number of clamping cycles
- tension value
- charge time
- discharge time
- charge time value
- discharge time value
- sum of state of charge
- displayed state of charge
- charging cycle time
- discharging cycle time
- state of health, etc



Failure Event data set
118 attributes

- failure events
- source
- source type
- timestamps of failures for all types of batteries

Aeroengine data set (~ 3GB)
155 attributes

- engine model,,,,, start time
- end time
- down time
- battery type
- timestamp

Tools

- Visualizations, Data Preparation and Descriptive Mining
  - Rapidminer
  - Python
  - Keras
  - Apache Splunk
  - Apache Kafka
- Predictive Mining and Deployment
  - Apache Splunk
  - Tensorflow
  - AWS EMR
- Computation

    Nvidia GeForce GTX Titan X

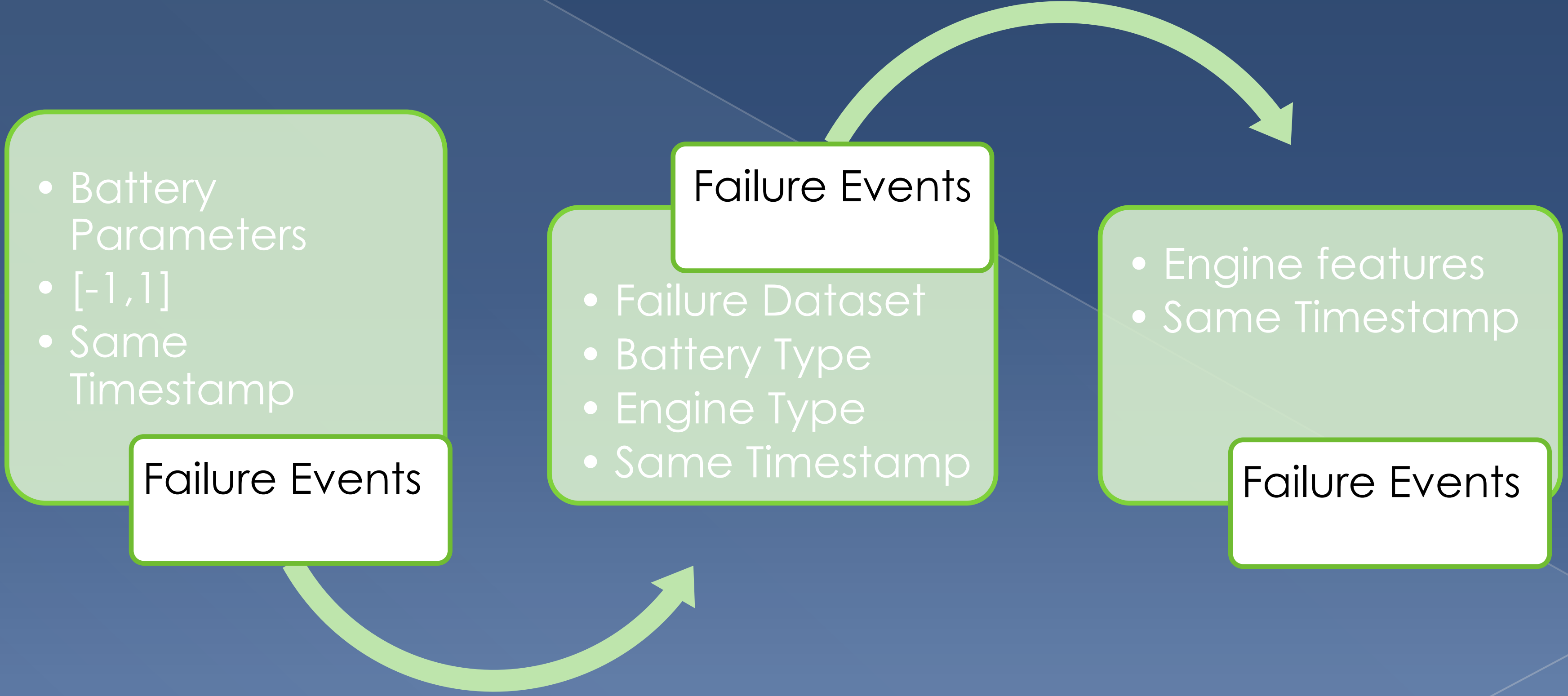
Descriptive Mining - Feature Transformation & Elimination

Feature Transformation and Elimination

- Null value Attributes and Missing Value Attributes removed
- Dataset had data from different time range, the events from the failure, aero engine and battery dates were put within same time range.
- Battery types and engine tyres were segmented
- The failure nodes were segregated from the failure event dataset and matched to each battery type and engine type
- Information gain and Gini Index were used to assign weights to the datasets to understand the correlation
- Attributes of battery data was serialized using Google Protocol Buffer into tf.Sequence.example format as Tensorflow is being used for the sequence to sequence labelling

Feature Transformation

- Labels of every sequence are rescaled to be in range of [-1, 1]
- Data Normalisation using MinMaxScaler
- Data Normalisation using Timesteps
- Making input sequence by using memory in batches (Keras)



Predictive Mining Approaches

Model Selection

Challenge: Selecting one of the feasible prediction tasks

- Deviation Detection

  Threshold values or ranges could be set for particular feature or feature set for predictive maintenance

- Remaining Useful Life Prediction (RUL)

  Regression problem

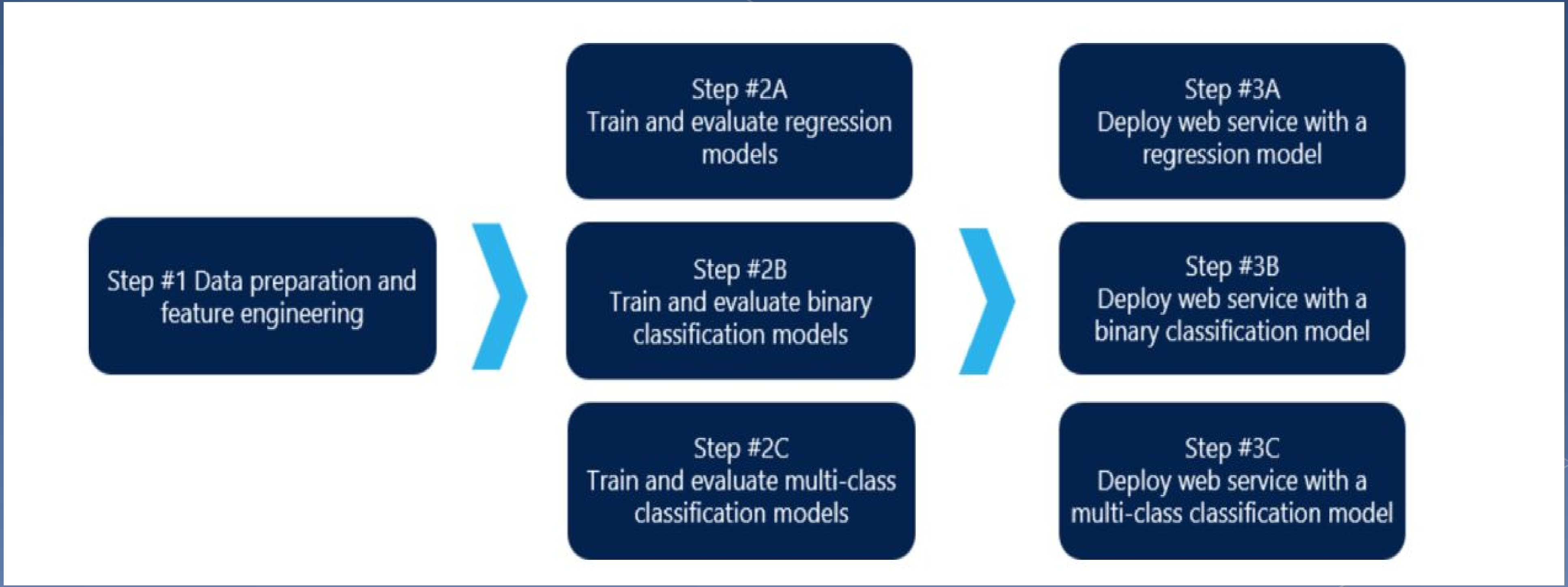
  One RUL model per machine is derived during training and stored in a database as references of degradation patterns. The database is deployed on-board and an algorithm tries to match the degradation of the monitored system to one of the references systems stored in the database

- Supervised Classification

  Monitor the individual correlations and classify accordingly to failure events.

Selected Prediction Approach

Both Remaining Useful Life and Supervised Classification approaches considered in this work



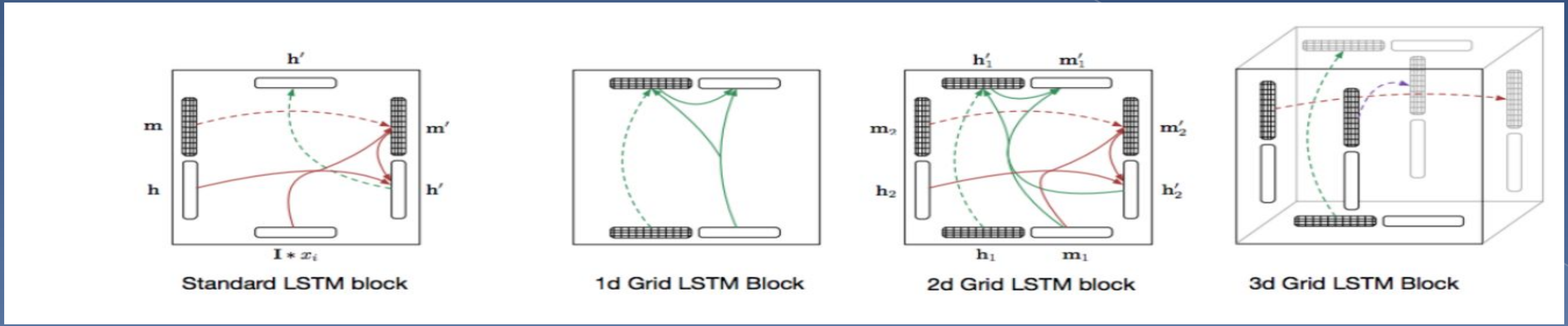
Step #1 Data preparation and feature engineering
Step #2A Train and evaluate regression models
Step #2B Train and evaluate binary classification models
Step #2C Train and evaluate multi-class classification models
Step #3A Deploy web service with a regression model
Step #3B Deploy web service with a binary classification model
Step #3C Deploy web service with a multi-class classification model

Model Selection - Grid LSTM

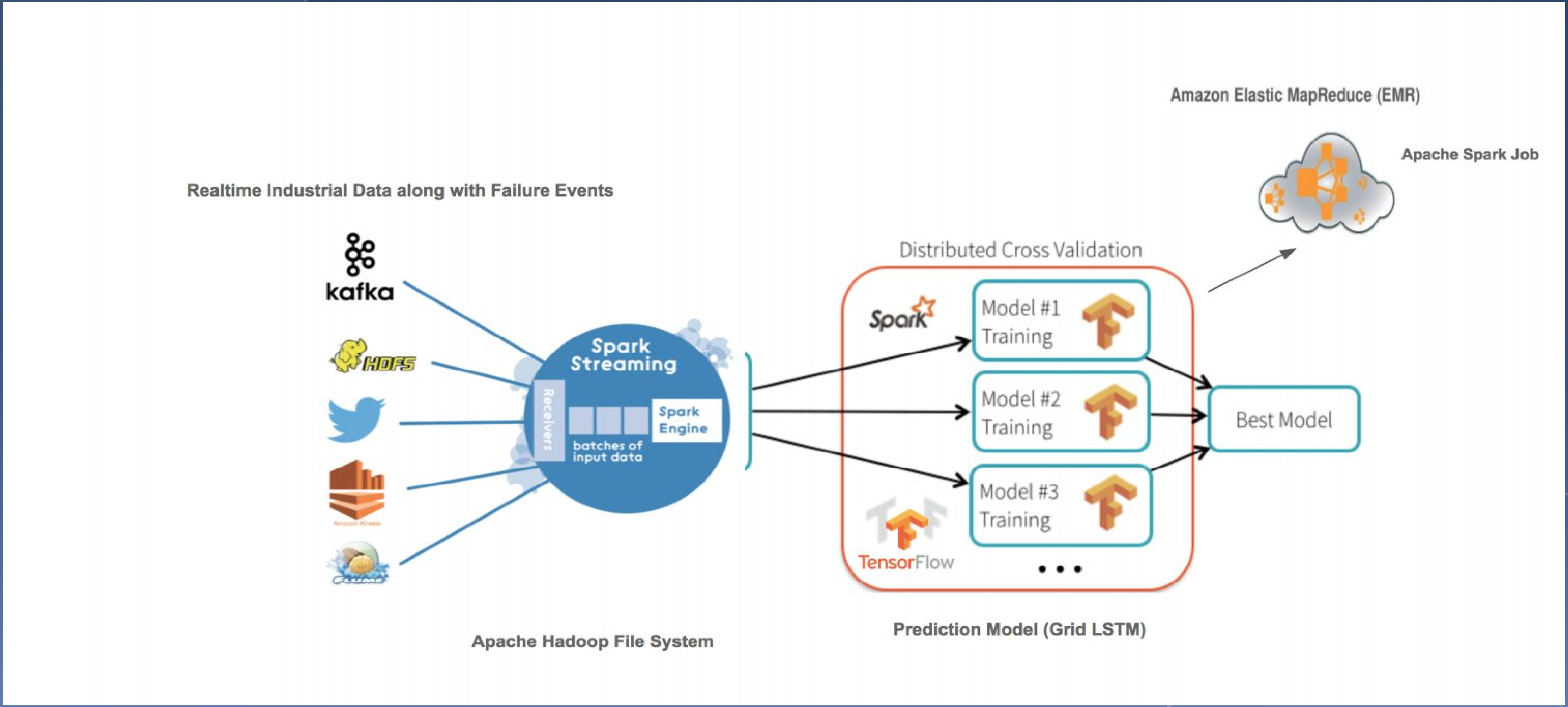
- A network of LSTM cells arranged in a multidimensional grid that can be applied to vectors, sequences or higher dimensional data such as images.
- The network differs from existing deep LSTM architectures in that the cells are connected between network layers as well as along the spatiotemporal dimensions of the data.
- The network provides a unified way of using LSTM for both deep and sequential computation.
- Grid LSTM has been used to define a novel two-dimensional translation model, the Reencoder where translation is done in a two-dimensional mapping. One dimension processes the source sequence whereas the other dimension produces the target sequence

Model Selection - Grid LSTM

- Grid LSTM deploys cells along any or all of the dimensions including the depth of the network.
- In the context of predicting a sequence, the Grid LSTM has cells along two dimensions, the temporal one of the sequence itself and the vertical one along the depth.
- To modulate the interaction of the cells in the two dimensions, the Grid LSTM proposes a simple mechanism where the values in the cells cannot grow combinatorially



Architecture



Implementation

- LSTMs let model extract abstract features out of the sequence of failed events in the window rather than engineering those manually
- LSTMs able to use larger window sizes and use all the information in the window as input
- LSTM layers expect an input in the shape of a numpy array of 3 dimensions (samples, time steps, features) where samples is the number of training sequences, time steps is the look back window or sequence length and features is the number of features of each sequence at each time step
- We train multi layered LSTM (2-LSTM and Stacked LSTM)
- The first layer is a Grid LSTM layer with 100 units followed by another Grid LSTM layer with 50 units.
- Dropout is also applied after each LSTM layer to control overfitting.
- Final layer is a Dense output layer with single unit and sigmoid activation since this is a binary classification problem.

Implementation

- Model prediction is done at each time step. We use a regressor to approximate a sequence of vectors. We repeat the same procedure replacing the Grid LSTM layers with only LSTM layers
- We use mini-batches of size 15 and optimize the network using Adam and a learning rate decay of 0.95.
- Both hyperparameter tuning and deployment at scale done with Apache Spark
- Hyperparameter Tuning : Use Spark to broadcast the common elements such as data and model description, and then schedule the individual repetitive computations across a cluster of machines in a fault-tolerant manner
- Deploying models at scale: Use Spark to apply trained neural network model on our dataset. The model is first distributed to the workers of the clusters, using Spark's built-in broadcasting mechanism. Then this model is loaded on each node and applied to the sequences.

Results

Training and Testing

- For each failure events, we construct a LSTM model (with 80 percent for training and 20 percent for testing)
- Allow the model to classify the sequences that affect the battery data
- Mean pooling is used on sequences to generate input for an extra neural network layer to further learn the feature representation
- The output of the neural network layer is learned by the survival model
- The failure metric is correlated to the prediction of failure sequences from the failure events
- Training done for 100,000 steps which ran for a week
- Accuracy is measured per sequence

Results

- Real time Predictive Maintenance Template with Apache Kafka, Apache Spark, Hadoop with AWS EMR

- Grid LSTM outperforms traditional LSTM for higher dimensional spatio temporal data when input data is noisy

	Accuracy	Precision	Recall	F-1 Score
Grid LSTM	0.77	0.75	0.79	0.74
LSTM	0.73	0.74	0.77	0.77

- LSTM and recurrent neural networks are an upgrade over Hidden Markov models for sequence classification

- Feature Engineering in real time production is easier with Apache Spark

Conclusion & Learnings

Some Learnings

- Grid LSTM tackles vanishing gradient problem and the exploding gradient problem better than traditional LSTM
- Choice of the appropriate loss function has a huge impact
- Deep Learning requires extensive study
  - Algorithms/Implementations partially poorly documented
  - No “common approach”
- Data Preparation/Transformation
  - Is key for good results
  - Takes most of the time

Some Findings

- Number of hidden layers has a huge impact
- Try different feature sets, techniques and parameters
- Computational view of existing attention models is that it allows for a total of O(T^2) computation
- Consider spatiotemporal dimensions of the data
- Grid LSTM data can be n-dimensional

Future Work - I

- More extensive hyperparameter search and training iterations
- Experimenting with variants of scheduled sampling learning strategy for recurrent networks
- Use of different architectures with different number of layers and nodes
- Predicting Remaining Useful Life (regression)
- Use of TensorFlowonSpark (Yahoo)

Future Work - II

- Application of proposed network architecture to other predictive maintenance models
- Application of Hierarchical Multiscale Recurrent Neural Networks and Highway Networks (LSTM relies on pre-defined boundaries)

- Research on Biologically Plausible Deep Learning approaches
- Research on Capsules to deal with back propagation

THANK YOU

ANY QUESTIONS ?