Applications of AI for Predictive Maintenance

LAB 3: TRAINING AUTOENCODER FOR ANOMALY DETECTION

LAB 3 OVERVIEW

- In this section we focus on a deep learning technique, which is called anomaly detection. This is different from traditional time series analytics.
- In a nutshell, because only a small portion of the hard drives are defective in our use case, we can treat those as anomalies.
- For example, out of three million samples we dealt with in our training set within the previous lab, only 256 samples were defective. Those are "anomalies" among the big pool of well-functioning hard drives.
- Now, if we could have a model capable of learning what a proper functioning hard drive looks like, it could detect hard drives with a different behavior pattern.

WORKSHOP STRUCTURE

LAB 1 (2 hrs.)

Training GPU XGBoost models with RAPIDS for Time Series

1. Discuss predictive maintenance, Backblaze hard drive data, challenges of dealing with large noisy datasets.
2. Scope into short-term ML automation problem.
3. Ingest raw real dataset from GPU production line for XGBoost model.
4. Format DataFrames and into DMatrices to fit into model.
5. Analyze dataset statistics (i.e. false positives, true positives.)
6. Examine XGBoost performance over the model.
7. Learn how to balance imbalanced data and measure relevant KPIs to assess the model.

LAB 2 (2 hrs.)

Training GPU LSTM models using Keras+Tensorflow for Time Series

1. Discuss Recurrent Networks and Long Short-Term Memory (LSTMs).
2. Learn how to mitigate Vanishing Gradient Problem using LSTMs and get familiarized with their cell structure.
3. Learn how to create sequences of data to feed the temporal nature of RNNs.
4. Design different RNN architectures and even create your own model.
5. Learn how to create the confusion matrix and adjust the threshold to different f-1 scores.

LAB 3 (2 hrs. + 2 hrs.)

Training Autoencoder for Anomaly Detection and Hyperparameter Optimization

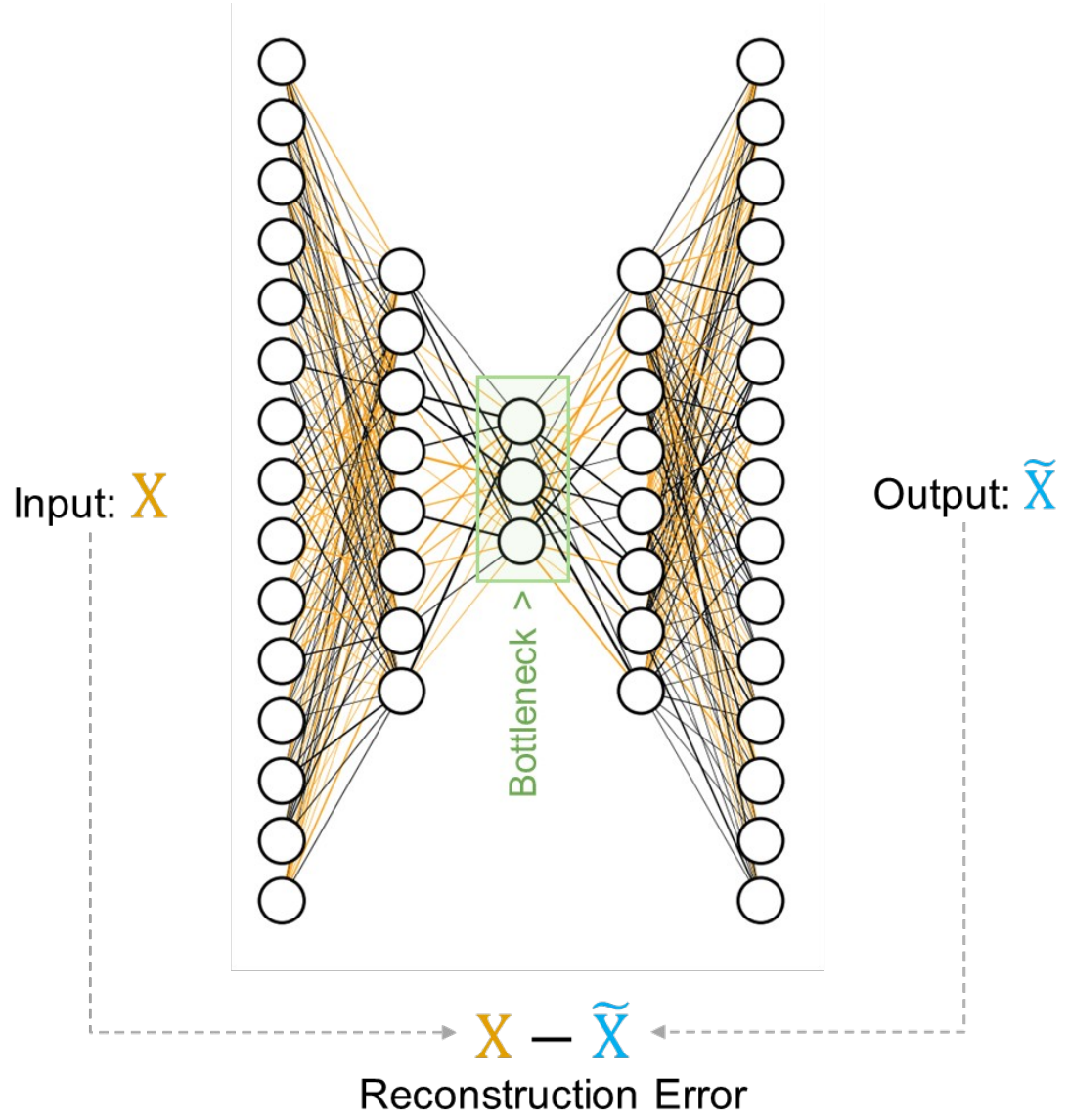
1. Focus on a different deep learning technique, which is called anomaly detection.
2. Learn how to use unsupervised learning algorithms like a deep autoencoder network to perform anomaly detection.
3. Learn how to create different autoencoder models in Keras.
4. General discussion on hyperparameter tuning and threshold settings.

AUTOENCODERS

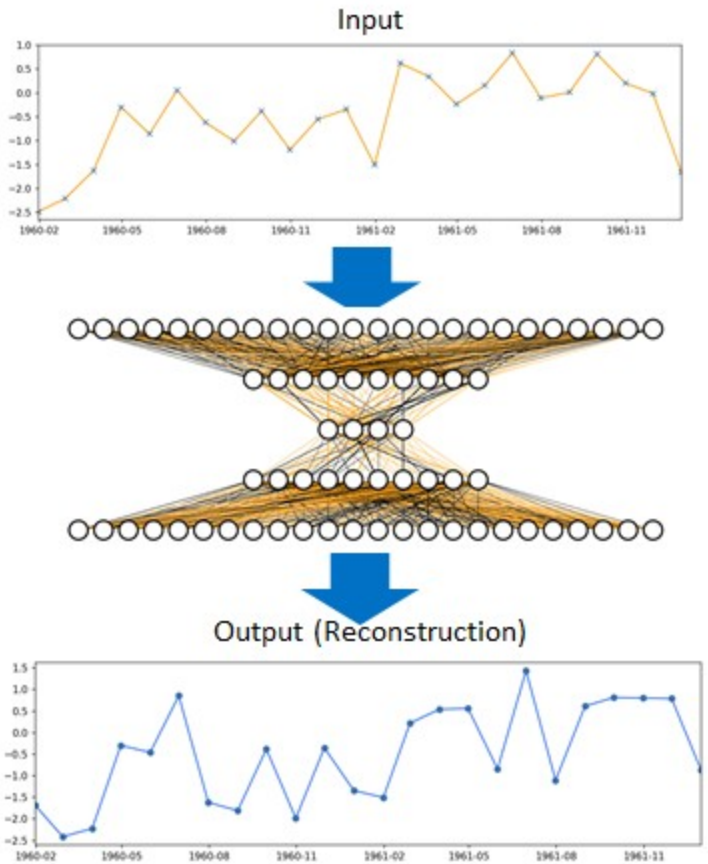
AUTOENCODERS

Deviation based anomaly detection method

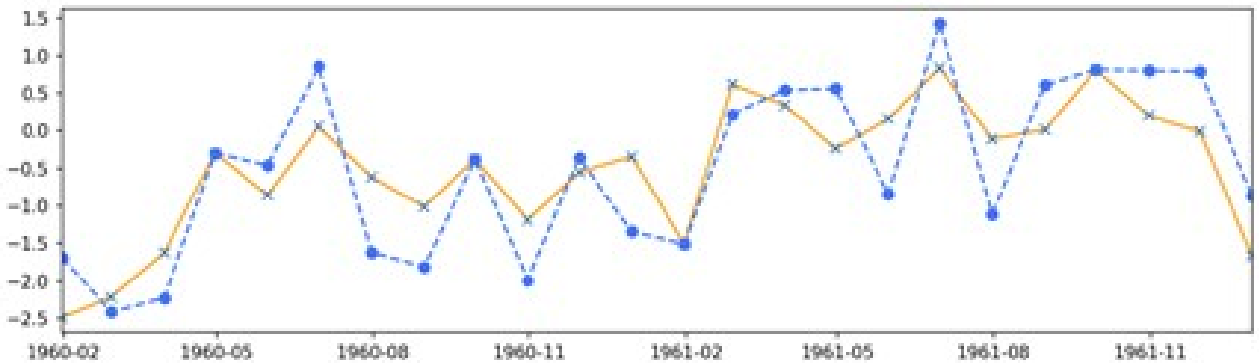
- Autoencoders are a form of unsupervised learning and have applications outside of anomaly detection
- An autoencoder consists of two parts the encoder and decoder
- Encoder is a neural network that maps the input to a (typically) lower-dimensional space
- Decoder is a neural network that maps the encoded data back to the input
- Anomalies have high reconstruction error



AUTOENCODERS



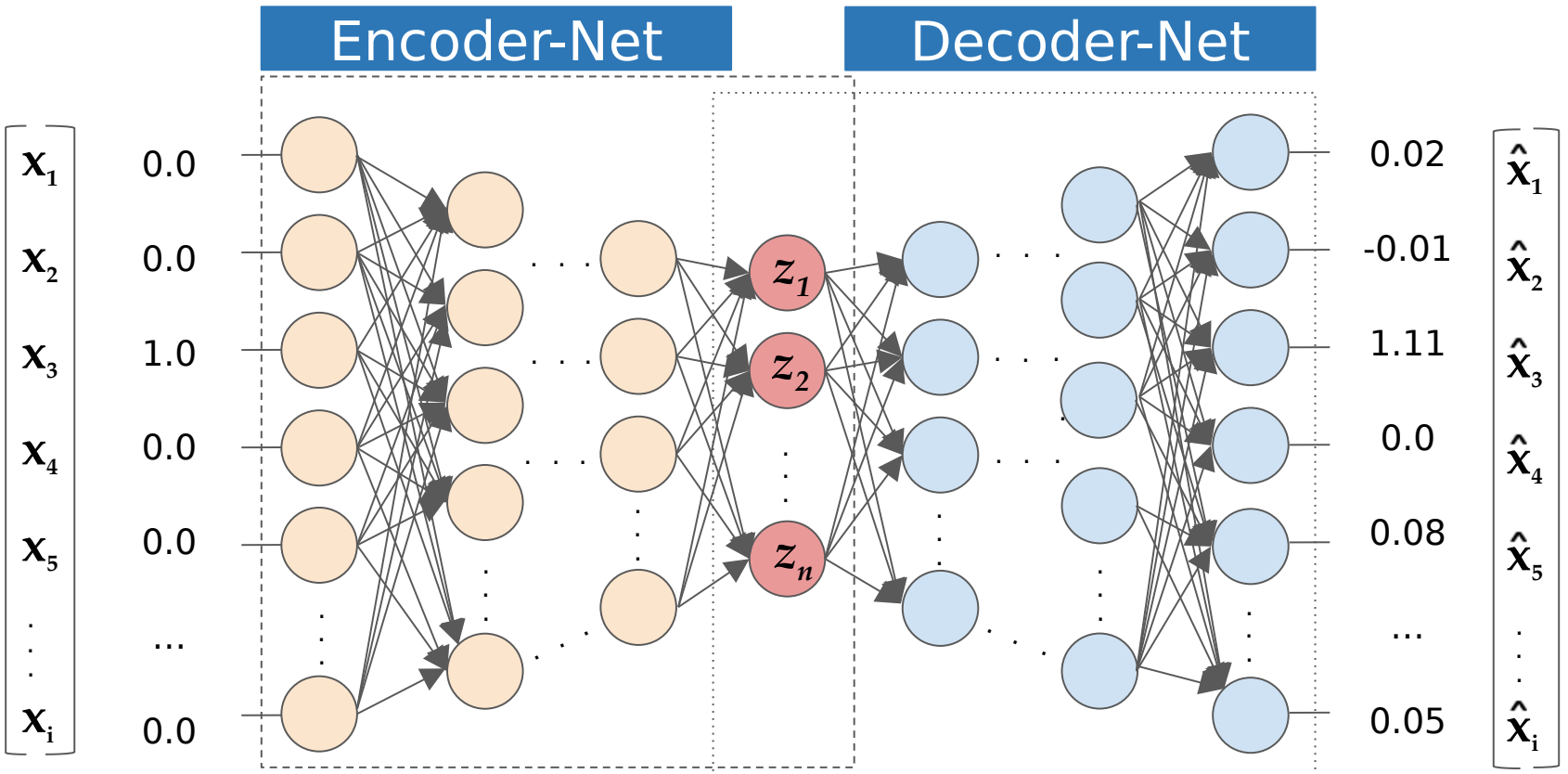
TIME SERIES RECONSTRUCTION



Reconstruction vs Input

To train this type of neural network, we can calculate the difference between the input and the output and use that loss, typically called a reconstruction loss, as our loss function for training our model

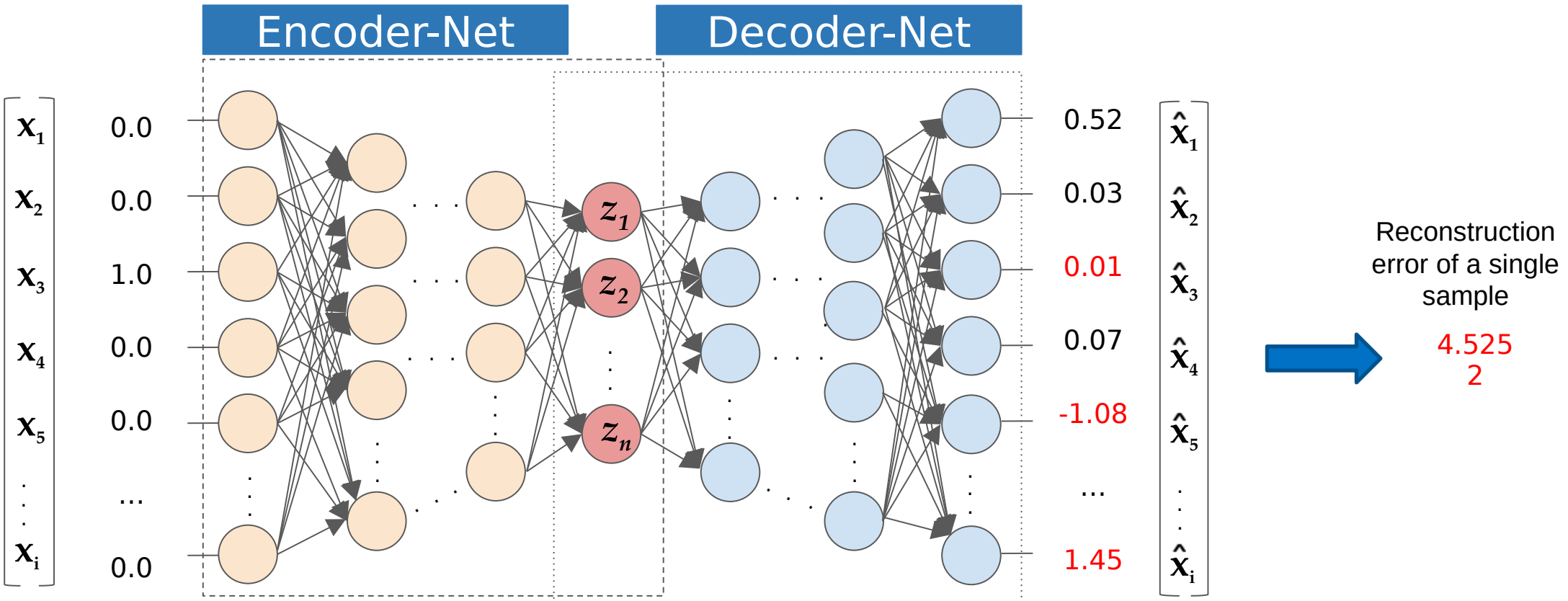
Reconstruction error of a regular sample



Reconstruction error of a single sample

0.021
5

Reconstruction error of an anomaly

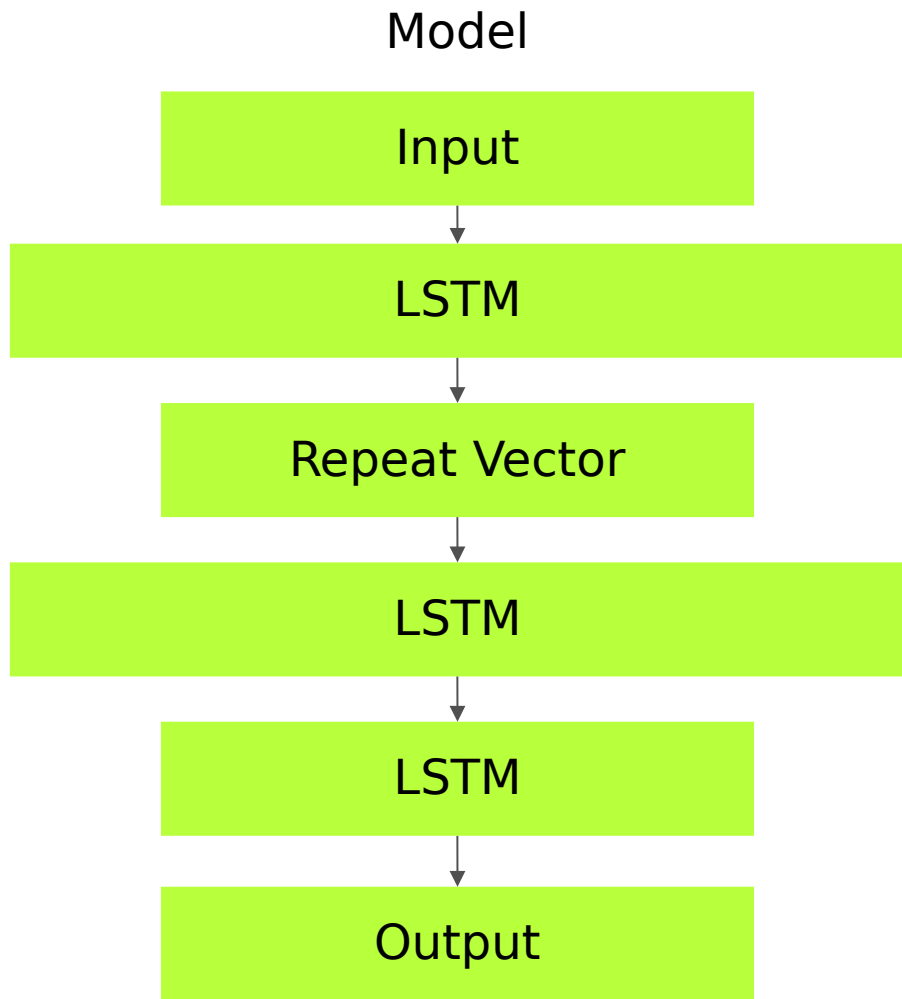


IN THIS LAB

- We will be working with 2 types of ”Autoencoders” for this lab:
  - LSTM-based AE (also know as a Seq2Seq Model)
  - 1D-Convolution-based AE

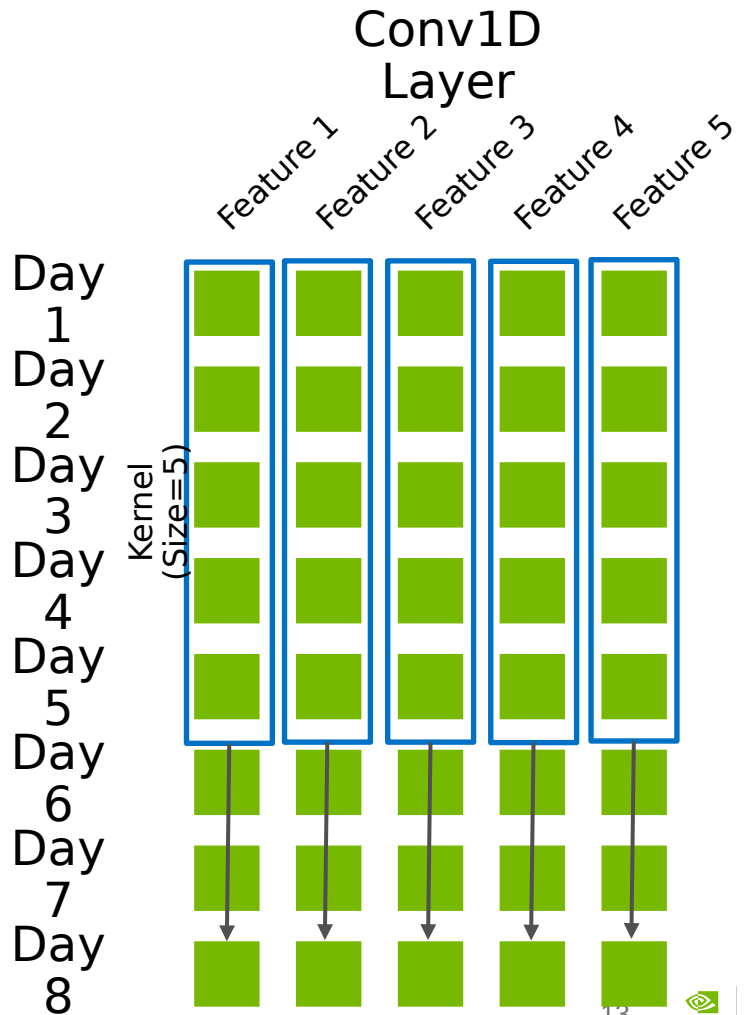
SEQ2SEQ MODEL

- Traditional LSTM-based model
- Takes sequence as input and produces sequence

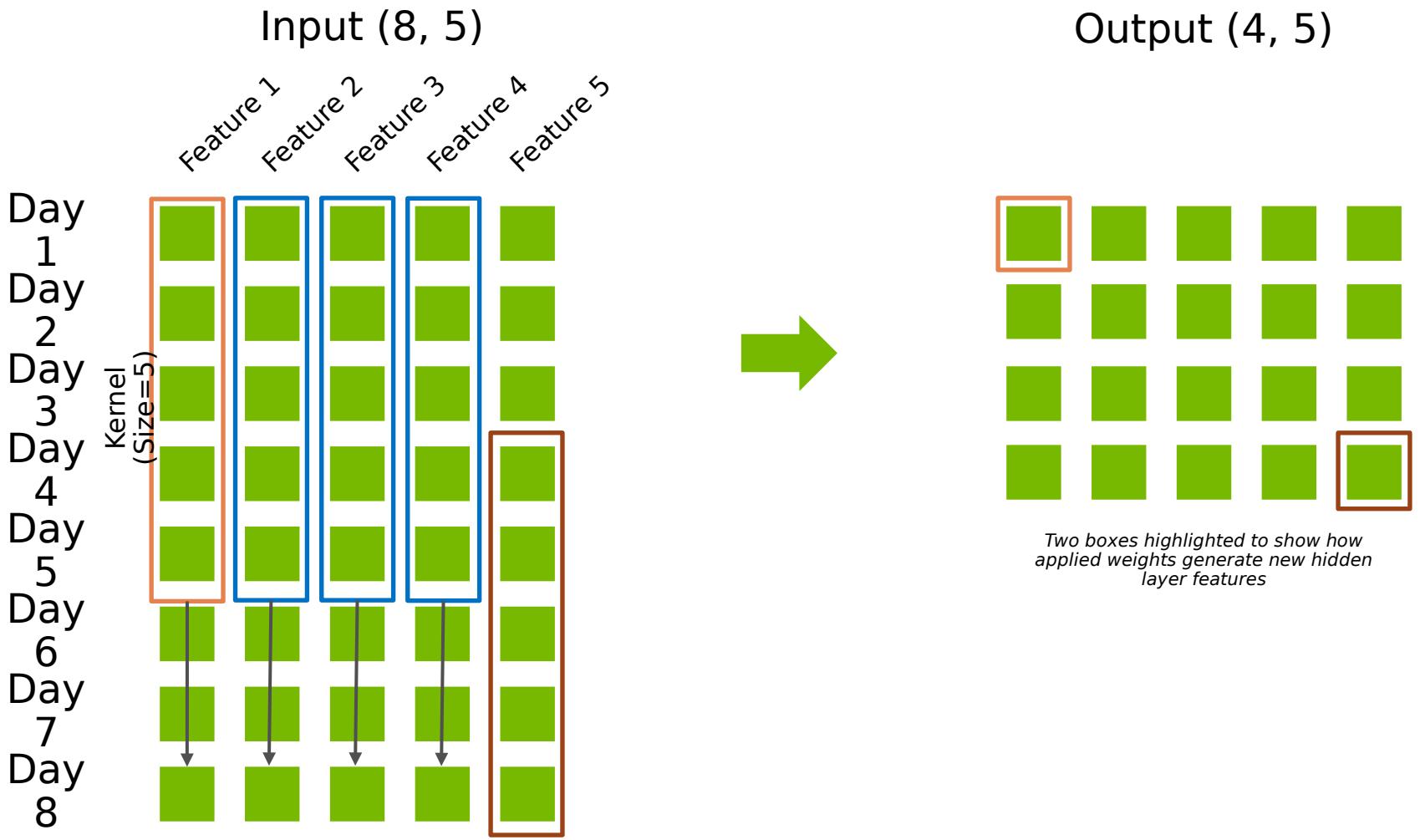


1D-CONVOLUTION AE MODEL

- Based on 1D Convolutions
- Similar to traditional 2D-Convolutions, but individual kernels focused on single features



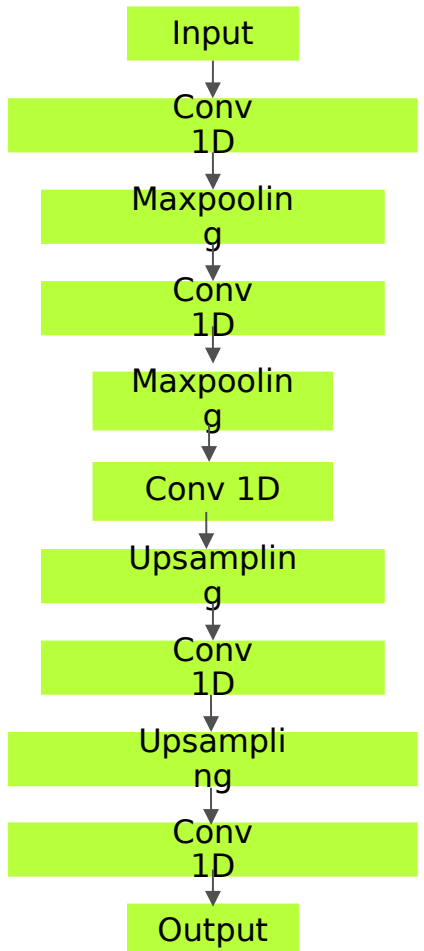
1D-CONVOLUTION AE MODEL



1D-CONVOLUTION AE MODEL

- Easier to train than traditional LSTM models
  - Less susceptible to noisy or missing data
  - Significantly faster to train
  - LSTM’s require large amounts of data to train

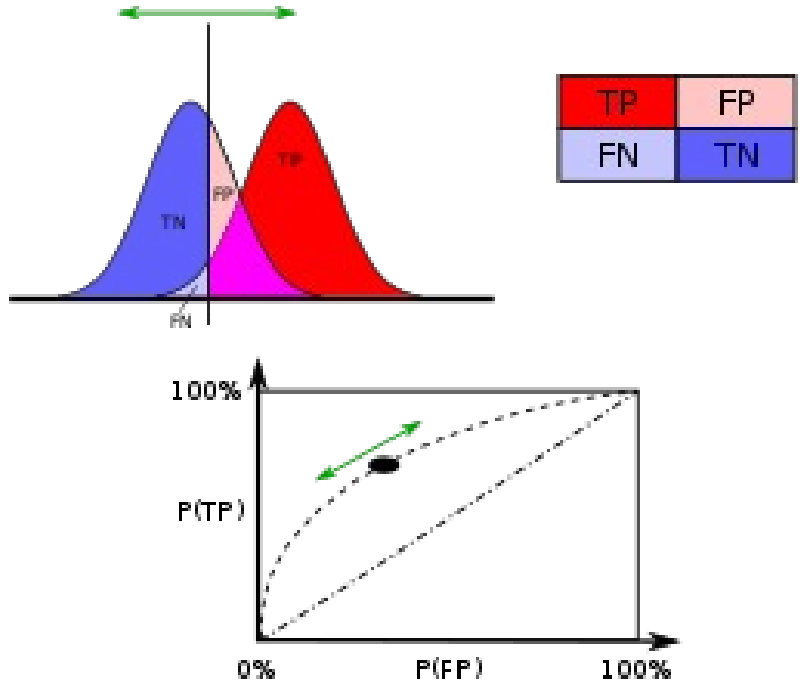
- Good paper discussing: https://arxiv.org/pdf/1803.01271.pdf



ROC CURVE

The performance of the anomaly detection model is affected by the threshold. The ROC curve helps us understand model performance by plotting the True Positive rate and False Positive rate at different thresholds.

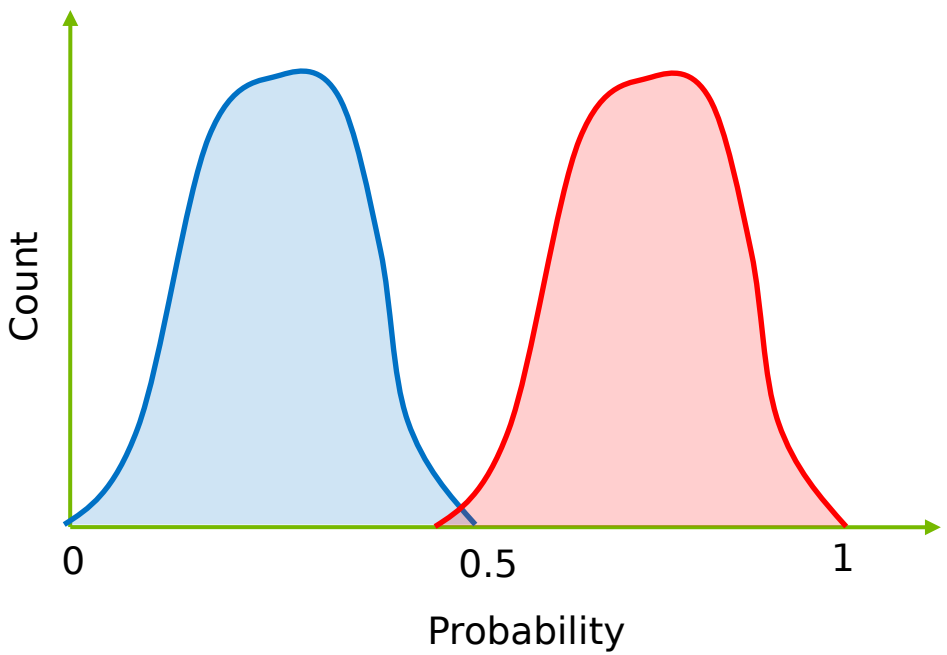
The threshold is high at the y-intercept, resulting in a 0% False Positive rate and a low True Positive rate. As the threshold decreases, the True Positive rate will increase but so will the False Positive rate.

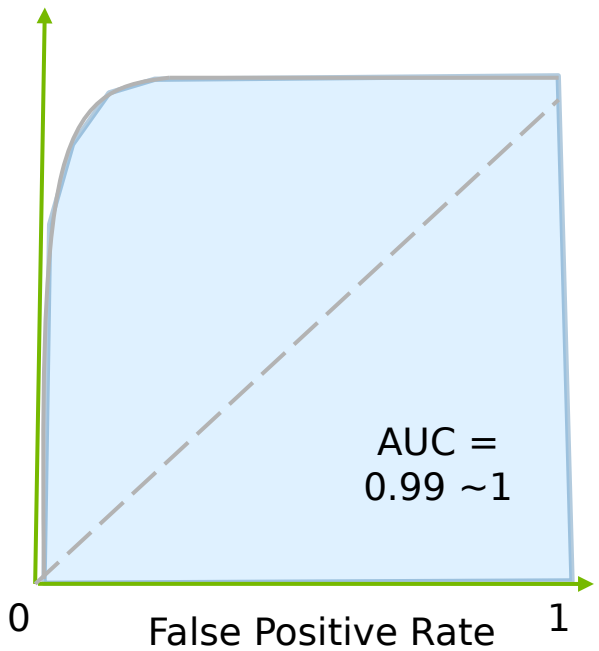


https://en.wikipedia.org/wiki/Receiver_operating_characteristic

ROC CURVE

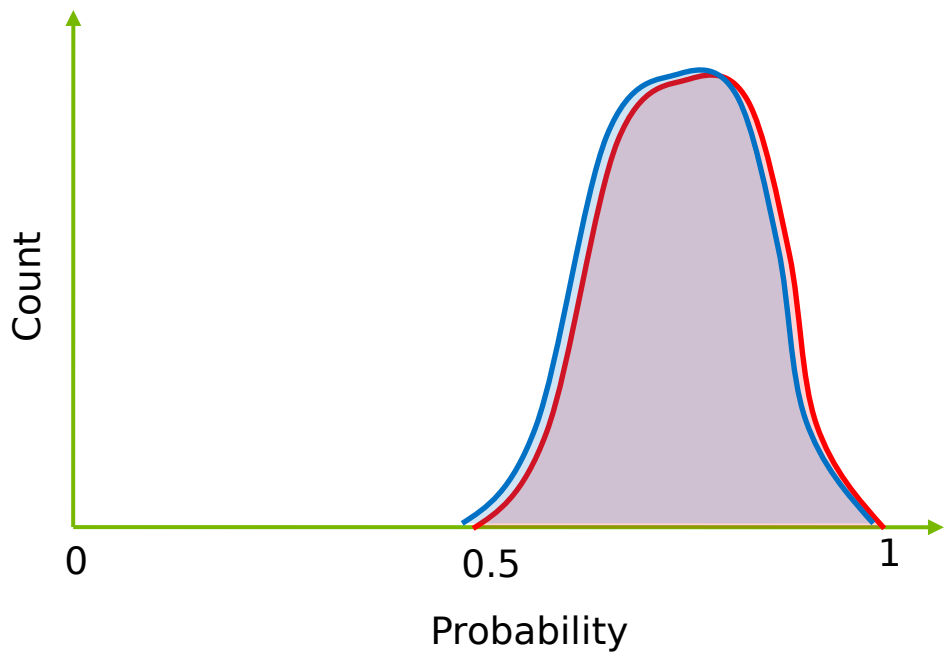
Interpretation

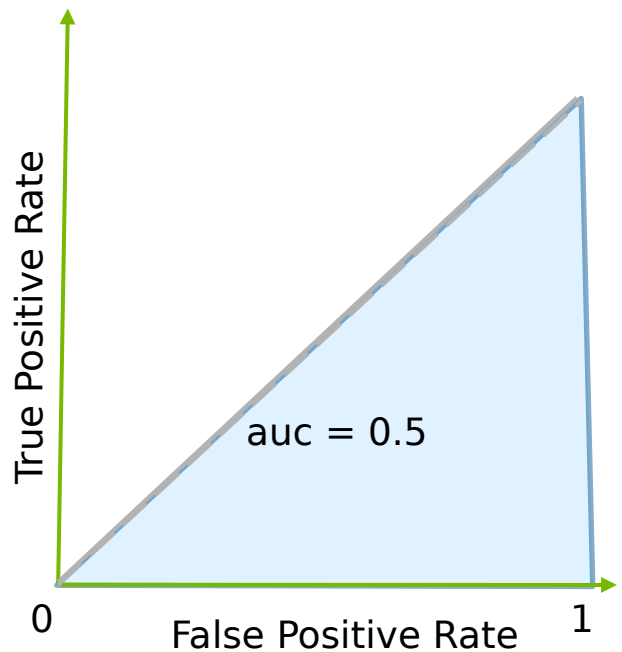




ROC CURVE

Interpretation





Good Separation Between Classes = High Performance Model/Classifier

COMMENT ON LAB 3

- Seq2Seq models look similar to traditional FC AE's

- Our model:
  - Input (16, 21)
    - 336 features
  - Latent Vector (128)
  - Repeat (16, 128)
  - Output (16, 21)

Encoder

Decoder

```
Model: "model_19"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_19 (InputLayer)        (None, 16, 21)            0
_________________________________________________________________
lstm_49 (LSTM)               (None, 128)               76800
_________________________________________________________________
repeat_vector_17 (RepeatVect (None, 16, 128)           0
_________________________________________________________________
lstm_50 (LSTM)               (None, 16, 128)           131584
_________________________________________________________________
lstm_51 (LSTM)               (None, 16, 21)            12600
=================================================================
Total params: 220,984
Trainable params: 220,984
Non-trainable params: 0
_________________________________________________________________
None
```

COMMENT ON LAB 3

We'll use a slightly different form of Keras model creation where we specify the individual layers and combine to build models. This is useful with Autoencoders when you want to split up the model to leverage the encoder or decoder.

Lab 3 LSTM AE:

```
inputs = Input(shape=(data_shape))
encoded = LSTM(lstm_width, dropout = dp_lvl, recurrent_dropout = dp_lvl, return_sequences = False,
                                    activation='relu')(inputs)
decoded = RepeatVector(data_shape[0])(encoded)
decoded = LSTM(lstm_width, return_sequences=True, activation='relu')(decoded)
decoded = LSTM(data_shape[1], activation='tanh', return_sequences=True)(decoded)
autoencoder = Model(inputs, decoded)
```

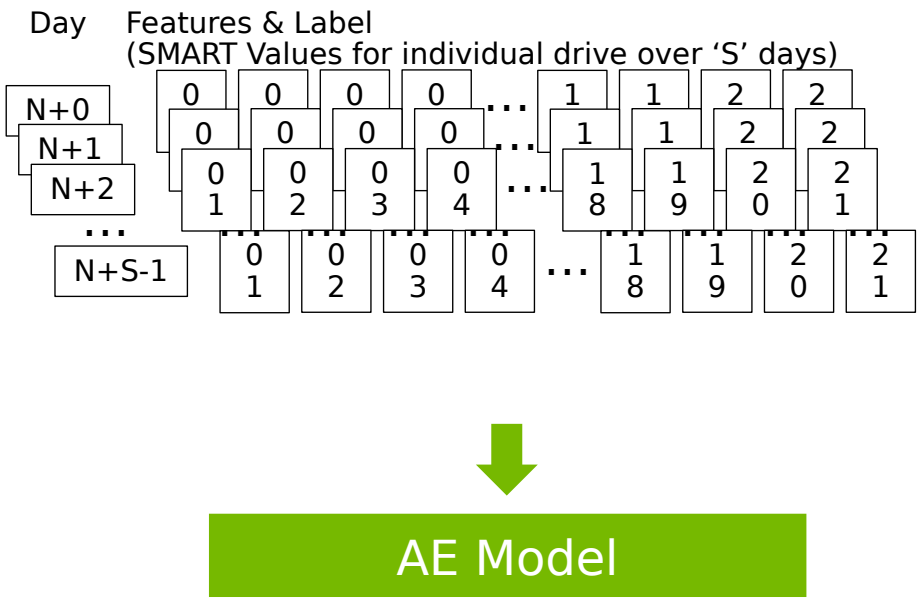
Lab 2 LSTM:

```
model = Sequential()
model.add(LSTM(units, input_shape=(x_train.shape[1], x_train.shape[2]),dropout = dp_lvl,recurrent_dropout = dp_lvl,
                                    return_sequences = True ))
model.add(LSTM(units, dropout = dp_lvl,recurrent_dropout = dp_lvl, return_sequences = True ))
model.add(LSTM(units, dropout = dp_lvl,recurrent_dropout = dp_lvl, return_sequences = False ))
model.add(Dense(units, activation='tanh',activity_regularizer=regularizers.l2(regularizer_lvl)))
model.add(Dropout (0.2))
model.add(Dense(units, activation='tanh',activity_regularizer=regularizers.l2(regularizer_lvl)))
model.add(Dense(1, activation='tanh',activity_regularizer=regularizers.l2(regularizer_lvl)))
```

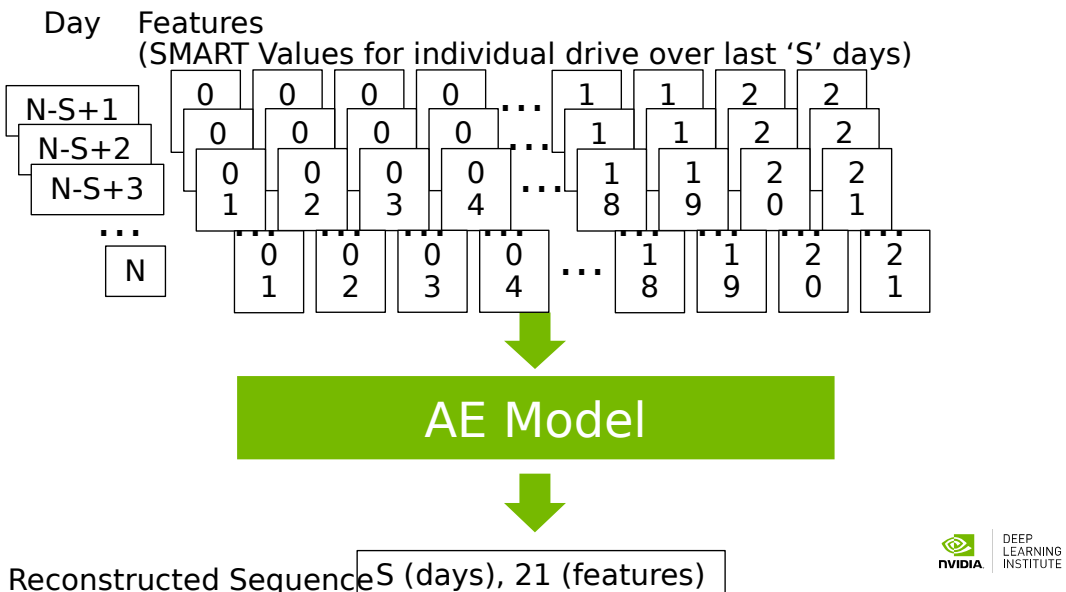
LAB 3 MODEL SUMMARY – AE'S

Train AE Model with "normal" sequences of length 'S' days

Training:
Use multi-day sequences of SMART features (from an individual **"normal"** hard drives) to train a AE. Features and Label will be the full sequence.



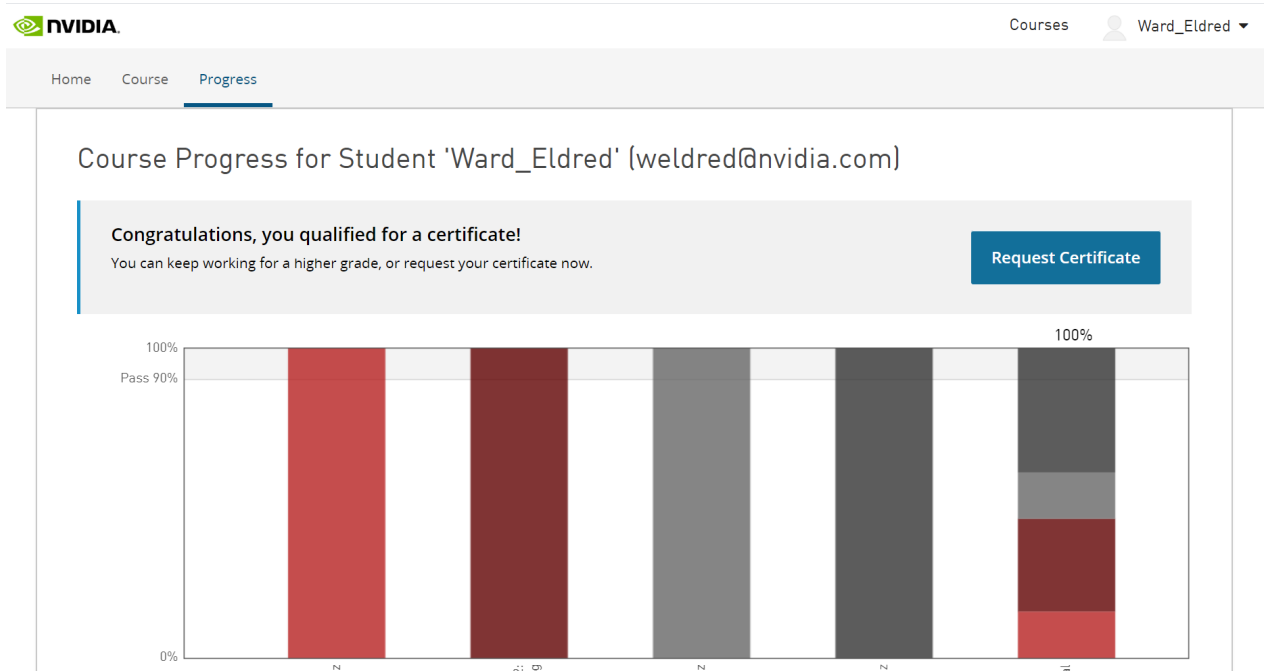
Inference:
Pass sequence of SMART values for an individual hard drive over the last 'S' days. For **"normal"** hard drives, should reconstruct with minimal error.



END OF LAB NOTES

END OF LAB

- Certificate: When you've completed the lab and Task 2 Assessment, go to "Progress" to request your certificate for completing this course.
- Feedback: From the Task 3 Course page, you can select Next to complete the course survey



STARTING THE LAB

SUMMARY

LAB 3 SUMMARY

Training Autoencoder for Anomaly Detection

- Focus on a different deep learning technique, which is called anomaly detection.
- Learn how to use unsupervised learning algorithms like a deep autoencoder network to perform anomaly detection.
- Learn how to create different autoencoder models in Keras.

HYPERPARAMETER OPTIMIZATION (OPTIONAL DISCUSSION TOPIC)

Hyperparameter Search

Hyperparameters are inputs to a machine learning/deep learning model. They are specified before the model training using a specified dataset starts. A trained model is the outcome of this process. So hyperparameters for a specific ML/DL model along with the dataset define the actual model to be used for inference.

Examples include sigma and C parameters for SVM (ML) and learning rate in LSTM/AE/CNNs in DL.

Loss functions in ML/DL represent the ability of a specified model with specified hyperparameters to adapt to a specific dataset. They can very multi-modal and change significantly with regard to change in hyperparameters.

Hence it is important to search for a good selection of hyperparameters for a particular model and a specified dataset. There are several options to do hyperparameter search/optimization : a brute force approach of grid search, randomized search, Bayesian optimization and evolutionary optimizations and many others.

Issues with Traditional Strategies

- Data Scientists have, traditionally, leveraged Grid Search to check parameter spaces - which is SLOW !!

- Models complexity is going up
  - Training is taking longer
  - Manual hyperparameter tuning is more difficult

- Need to move to more optimal search solutions…

Issues with Traditional Strategies

- Data Scientists have, traditionally, leveraged Grid Search to check parameter spaces - which is SLOW !!

- Models complexity is going up
  - Training is taking longer
  - Manual hyperparameter tuning is more difficult

- Need to move to more optimal search solutions…

```
Recent Test GridSearch I Ran:

fine_gbm_param_grid = {
    **base_grid,
    "min_child_weight": [1, 3, 5, 7, 9],
    "gamma": [0, 1, 2, 4],
    "subsample": [0.6, 0.8, 1.0],
    "colsample_bytree": [0.6, 0.8, 1.0],
    "max_depth": [1, 2, 3, 4, 5, 8],
    "reg_alpha": [0, 1],
    "reg_lambda": [0, 1],
    "learning_rate": [0.30, 0.35, 0.40, 0.45, 0.50, 0.55,
                      0.60, 0.65, 0.70],}
```

- 38,880 combinations (5*4*3*3*6*2*2*9)

- Each training took ~3 minutes

- Total Search Time = 81 DAYS

Bayesian Optimization

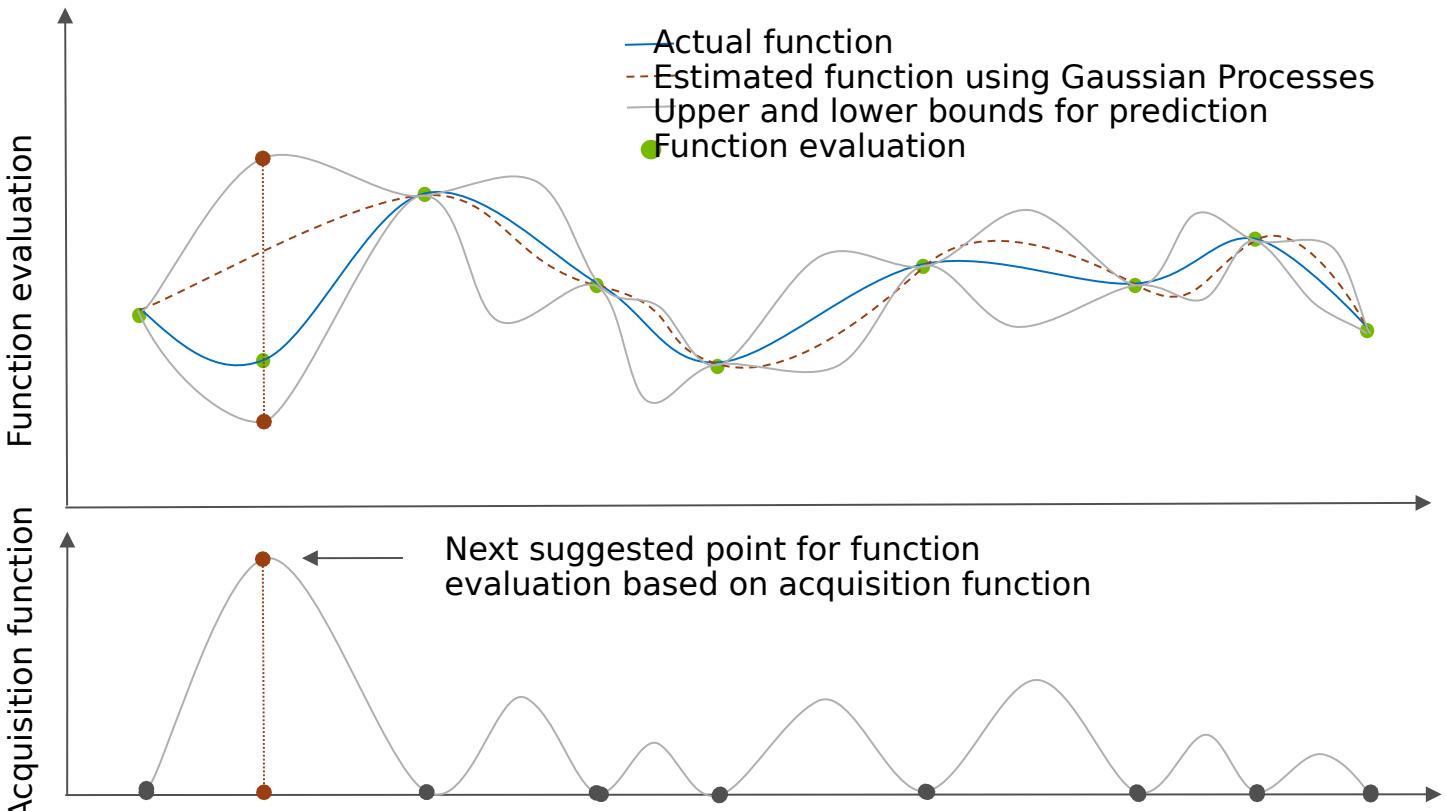
Any hyperparameter search will consist of a few initial model trainings to understand the change of the loss function with regard to the change in hyperparameters.

This parameter space can be very multi-modal. As it is often expensive to run model trainings, it is important to select the next set of hyperparameters in an optimal way. This can be done taking into account the results so far and the uncertainty associated with them.

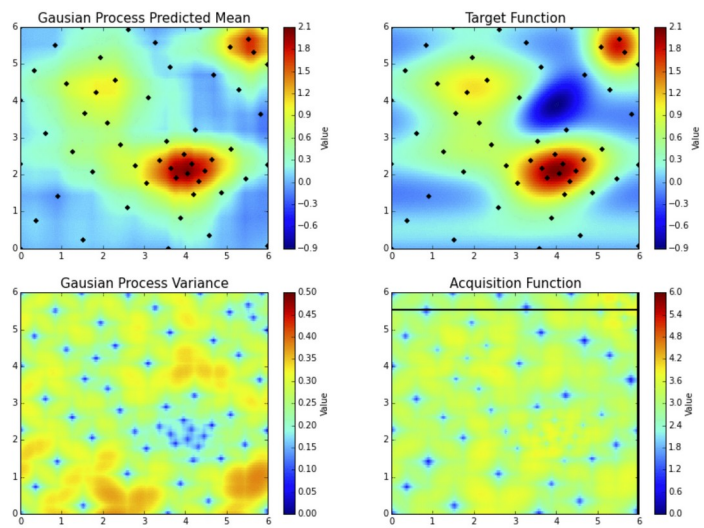
Bayesian Optimization takes this approach by sequentially selecting points based on the mean and uncertainty in a surface approximating the loss function result with regard to the hyper parameters.

“Bayesian Optimization: Open Source constrained global optimization tool for Python”
Fernando Nogueira, 2014
https://github.com/fmfn/BayesianOptimization

Bayesian Optimization



Optimization in two-dimensional hyper parameter space



https://github.com/fmfn/BayesianOptimization

Parameter Space

Using Bayesian Optimization, we will explore hyperparameter optimization for the LSTM based autoencoder explained in this Lab.

The hyperparameters are taken to be :

- Learning rate
- LSTM width

We will also use the package shown earlier to perform hyperparameter optimization

REFERENCES

Brief overview on hyperparameter optimization
https://en.wikipedia.org/wiki/Hyperparameter_optimization

Some popular opensource packages
Katib, H2O AutoML, hyperopt, scikit-optimize

Commerical services
Google Hypertune, Sigopt

A Tutorial on Bayesian Optimization
https://arxiv.org/abs/1807.02811

STARTING THE LAB

SUMMARY

LAB 3 PART 2 SUMMARY

Hyperparameter Optimization

- Hyperparameter optimization is key to maximizing model accuracy
- Traditional methods of hyperparameter search are slow and computationally expensive
- Leveraging new, optimized techniques are beneficial
  - Faster to obtain results
  - Able to sweep larger parameter spaces

COURSE SUMMARY

WORKSHOP REVIEW

LAB 1 (2 hrs.)

Training GPU XGBoost models with RAPIDS for Time Series

1. Discuss predictive maintenance, Blackblaze hard drive data, challenges of dealing with large noisy datasets.
2. Scope into short-term ML automation problem.
3. Ingest raw real dataset from GPU production line for XGBoost model.
4. Format DataFrames and into DMatrices to fit into model.
5. Analyze dataset statistics (i.e. false positives, true positives.)
6. Examine XGBoost performance over the model.
7. Learn how to balance imbalanced data and measure relevant KPIs to assess the model.

LAB 2 (2 hrs.)

Training GPU LSTM models using Keras+Tensorflow for Time Series

1. Discuss Recurrent Networks and Long Short-Term Memory (LSTMs).
2. Learn how to mitigate Vanishing Gradient Problem using LSTMs and get familiarized with their cell structure.
3. Learn how to create sequences of data to feed the temporal nature of RNNs.
4. Design different RNN architectures and even create your own model.
5. Learn how to create the confusion matrix and adjust the threshold to different f-1 scores.

LAB 3 (2 hrs.)

Training Autoencoder for Anomaly Detection

1. Focus on a different deep learning technique, which is called anomaly detection.
2. Learn how to use unsupervised learning algorithms like a deep autoencoder network to perform anomaly detection.
3. Learn how to create different autoencoder models in Keras.
4. General discussion on hyperparameter tuning and threshold settings.

FINAL THOUGHTS

- You’ll have access to this course for a year, so feel free to come back at your leisure and review the content.
- Hopefully you can apply some of your learnings to your own data. We’d love to hear about any interesting findings you might have.
- Please take the course survey to provide us feedback on what you enjoyed and what we can improve.