AWS re:Invent

AIM 401-R2

Deep Learning Applications Using TensorFlow

Julien Simon
Principal Tech. Evangelist, AI/ML
Amazon Web Services
@julsimon

Kevin Clifford
Senior Product Manager
Advanced Microgrid Solutions

Andrew Martinez
Staff Research Scientist
Advanced Microgrid Solutions

AWS re:Invent

Agenda

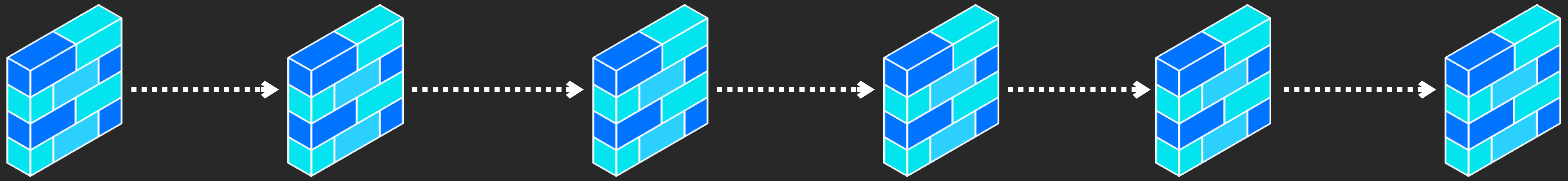
- Amazon SageMaker
- TensorFlow
- Case study: Advanced Microgrid Solutions
- Resources

Amazon SageMaker

AWS re:Invent

Amazon SageMaker

Easily build, train, and deploy Machine Learning models



Collect and prepare training data

Choose and optimize your ML algorithm

Set up and manage environments for training

Train and tune model (trial and error)

Deploy model in production

Scale and manage the production environment

FREE TIER

Amazon SageMaker

Notebook instances

Built-in, high-performance algorithms

ALGORITHMS	K-Means Clustering Principal Component Analysis Neural Topic Modelling Factorization Machines Linear Learner	XGBoost Latent Dirichlet Allocation Image Classification Seq2Seq, And more!
FRAMEWORKS	Apache MXNet, Chainer TensorFlow, PyTorch	Caffe2, CNTK, Torch

Set up and manage environments for training

Train and tune model (trial and error)

Deploy model in production

Scale and manage the production environment

Build

AWS re:Invent

Amazon SageMaker

Notebook instances

Built-in, high-performance algorithms

One-click training

Automatic Model Tuning

Deploy model in production

Scale and manage the production environment

Build

Train

Amazon SageMaker

Notebook instances

Built-in, high-performance algorithms

One-click training

Automatic Model Tuning

One-click deployment

Fully managed hosting with auto-scaling

Build

Train

Deploy

Selected Amazon SageMaker customers

GE Healthcare

intuit®

Hotels.com

DOW JONES

THOMSON REUTERS

edmunds.com®

DigitalGlobe™

Formula 1

cookpad

tinder™

grammarly

realtor.com®

TensorFlow

AWS re:Invent

TensorFlow

- Open source software library for Machine Learning
- Main API in Python, experimental support for other languages
- Built-in support for many network architectures: FC, CNN, LSTM, etc.
- Support for symbolic execution, as well as imperative execution since v1.7
  (aka eager execution)
- Complemented by the Keras high-level API

AWS re:Invent

AWS is the place of choice for TensorFlow workloads

"Of 388 projects, 80 percent using TensorFlow and other frameworks are running exclusively on AWS.

88% using only TensorFlow are running exclusively on AWS."

Nucleus Research report,
December 2017
https://aws.amazon.com/tensorflow

TensorFlow on Amazon SageMaker: a first-class citizen

- Built-in TensorFlow containers for training and prediction
  - Code available on Github: https://github.com/aws/sagemaker-tensorflow-containers
  - Build it, run it on your own machine, customize it, etc.
  - Supported versions: 1.4.1, 1.5.0, 1.6.0, 1.7.0, 1.8.0, 1.9.0, 1.10.0, 1.11.0

- Advanced features
  - Optimized both for GPUs and CPUs (Intel MKL-DNN library)
  - Distributed training
  - Pipe mode
  - TensorBoard
  - Keras
  - Automatic Model Tuning

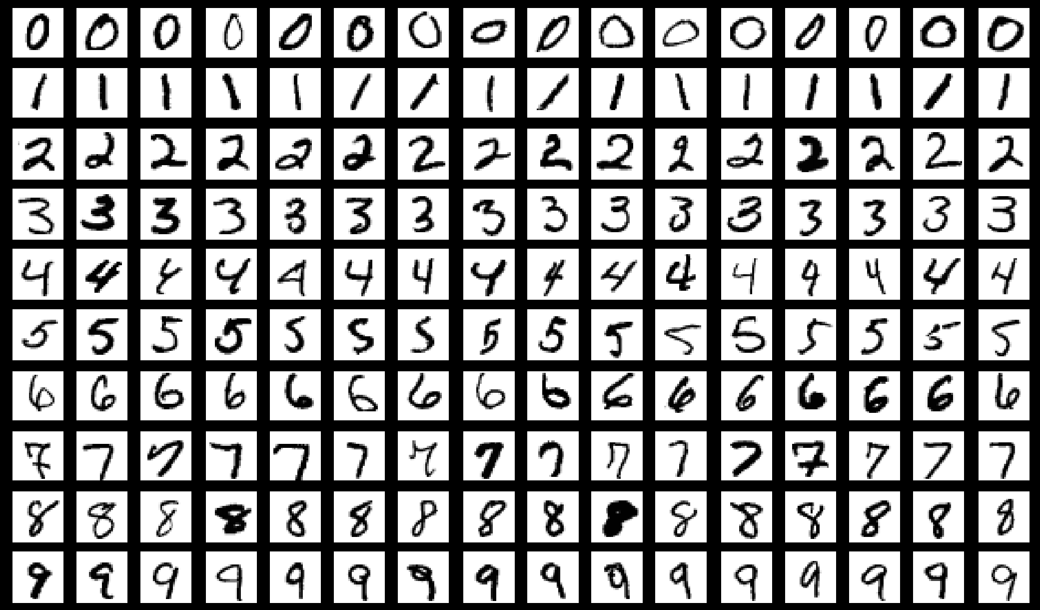
Using Keras on Amazon SageMaker

- Keras is a popular API running on top of TF, Theano and Apache MXNet.
- The *tf.keras* API is natively supported in Amazon SageMaker
- To use Keras itself (*keras.**), you need to build a custom container.
- This is not difficult!
  - Write a Dockerfile.
  - Build the container.
  - Push it to Amazon ECR.
  - Use it with *sagemaker.estimator.Estimator*.

- Full instructions and demo in this AWS Innovate talk: https://www.youtube.com/watch?v=c8Nhwr9VmfM

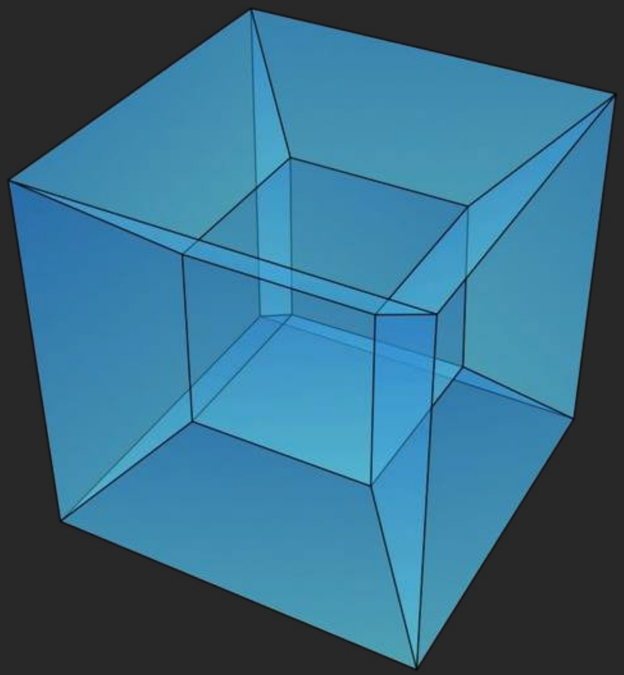
Example: MNIST with a Fully Connected network

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=10)
model.evaluate(x_test, y_test)
```



Automatic Model Tuning

Finding the optimal set of hyper parameters



1. Manual Search ("I know what I'm doing")
2. Random Search ("Spray and pray")
3. Grid Search ("X marks the spot")
   - Typically training hundreds of models
   - Slow and expensive
4. Hyper Parameter Optimization: use Machine Learning
   - Training fewer models
   - Gaussian Process Regression and Bayesian Optimization, https://docs.aws.amazon.com/sagemaker/latest/dg/automatic-model-tuning-how-it-works.html

Case study: Advanced Microgrid Solutions

Kevin Clifford
Senior Product Manager
Advanced Microgrid Solutions

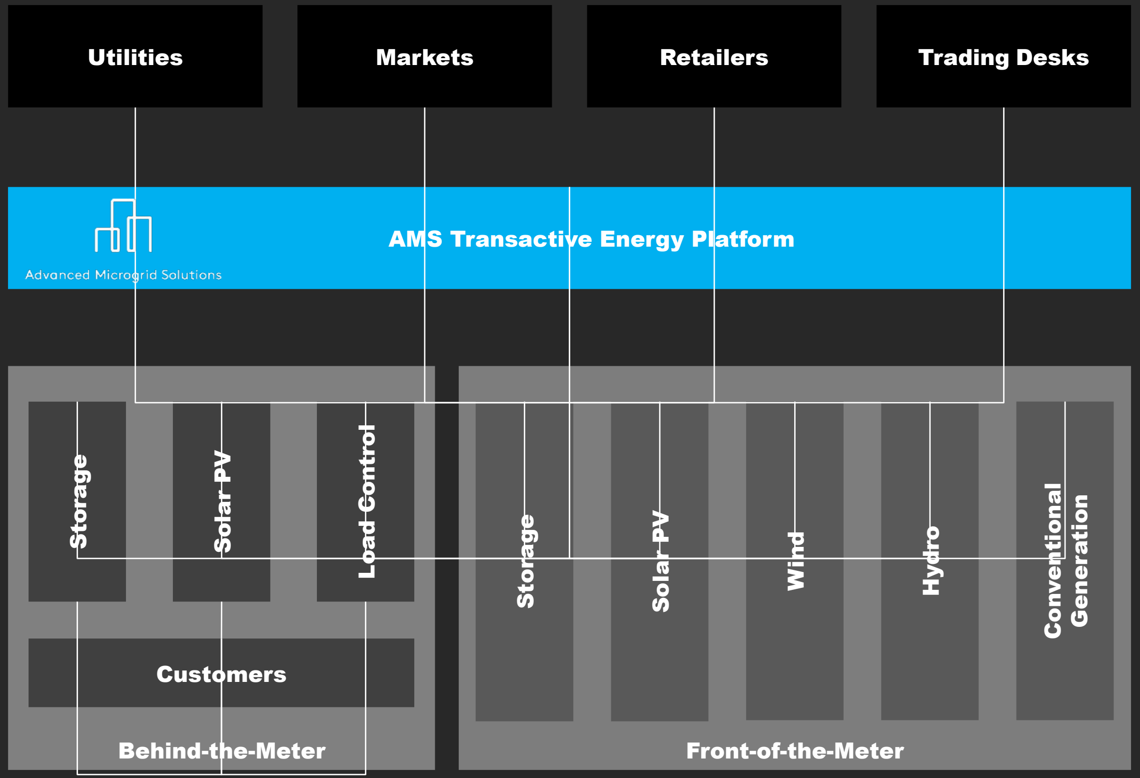
Andrew Martinez
Staff Research Scientist
Advanced Microgrid Solutions

AWS re:Invent

Advanced Microgrid Solutions (AMS)

Founded in 2013 in San Francisco, CA

Technology-agnostic energy platform and services company that maximizes wholesale energy market revenues for both behind-the-meter and front-of-the-meter assets



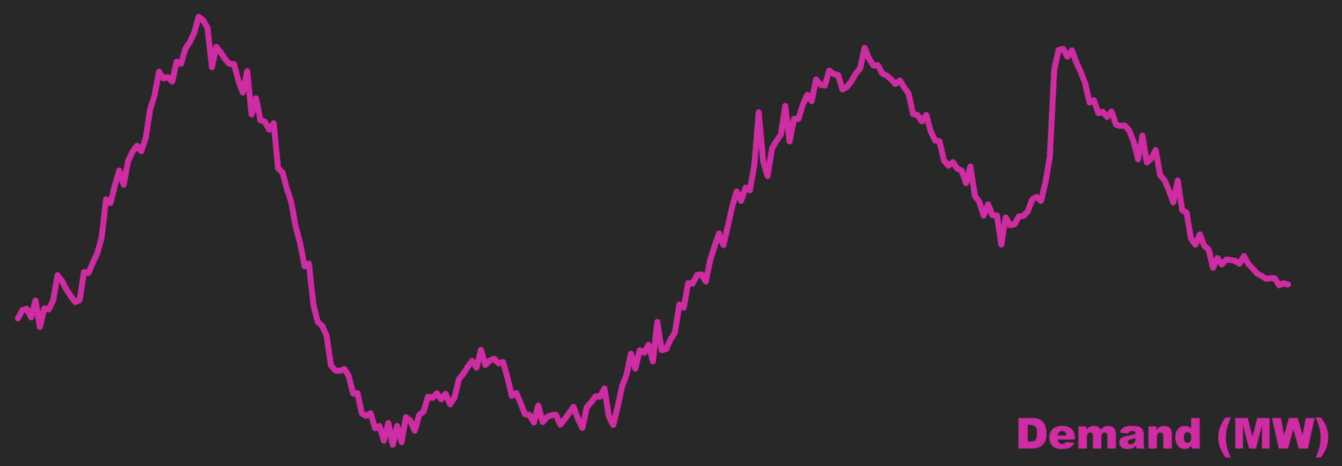
Situational and Business Intelligence → **Control and Aggregation** → **Economic Optimization** → **Grid and Market Participation** → **Asset Health Management**

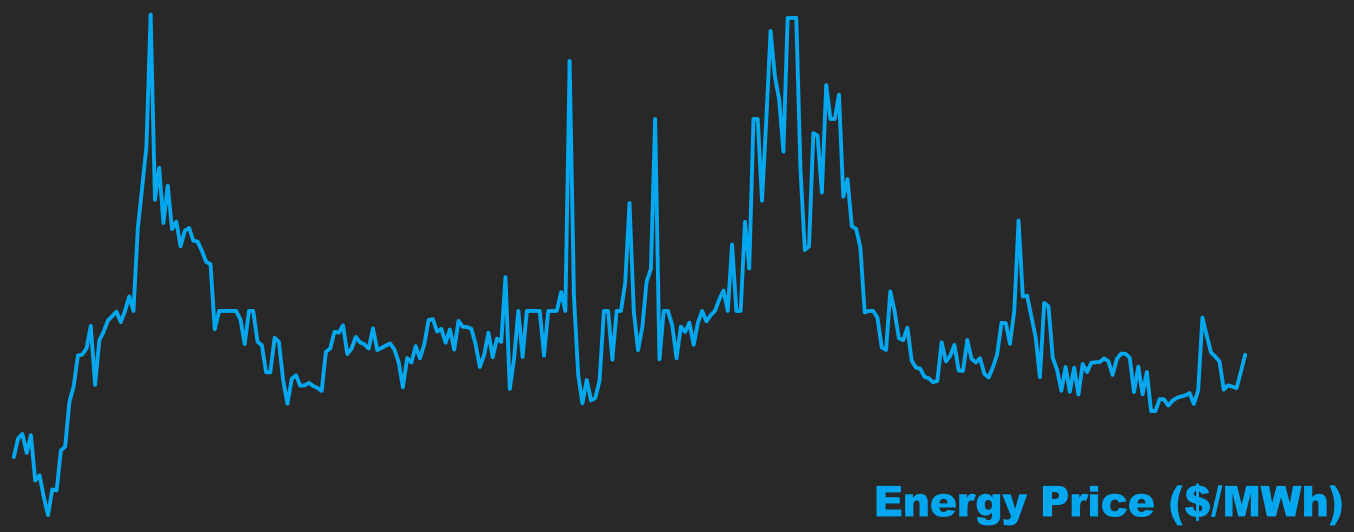
Mission Statement:

To lead a worldwide transformation to a clean energy economy by facilitating the deployment and optimization of clean energy assets

Energy Markets 101

- Electricity is traded in regional “wholesale energy markets”
- Supply = Demand at all times... or grid will fail
- Demand varies due to weather and behavioral factors
- Market operator must procure correct amount of supply to meet demand
- Suppliers must decide price and quantity to bid for every trading interval
- Market Price is set at the most expensive supplier needed to meet demand





Energy Technologies 101

Less complex ⟷ *More complex*

Thermal (coal, gas, oil)	**Renewables (solar, wind)**	**Hydroelectric**	**Batteries**
Bidding at marginal cost	Bidding at zero marginal cost + REC value	Use-limited resource bidding at opportunity cost	Use-limited resource bidding at opportunity cost across multiple market products





Use Case

Optimize market participation of clean energy assets in Australia's National Energy Market

Australia's National Energy Market (NEM) facilitates energy production for the 5 east Australian states

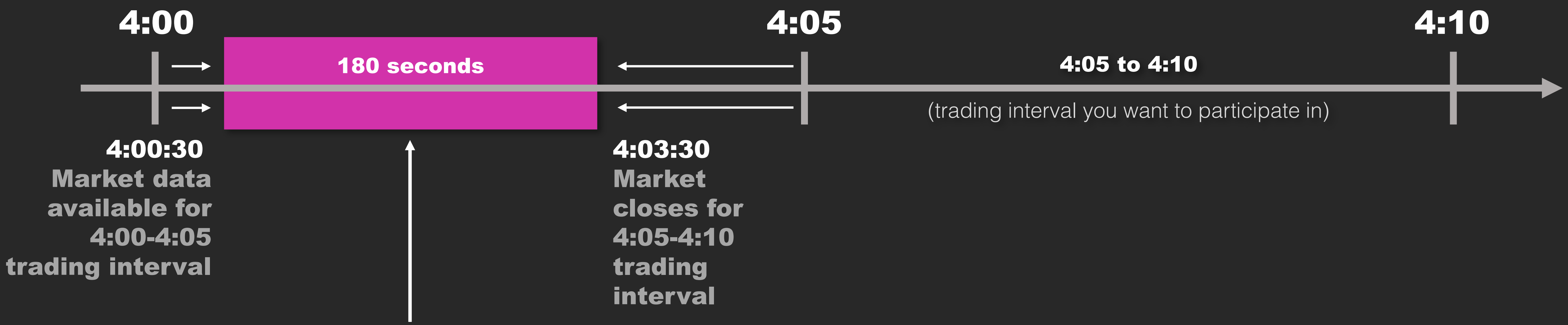
- Serves 9M customers
- $16.6B / 200 TWh traded annually

NEM is a "spot" market

- All parties bid to consume or generate energy during upcoming 5-minute time window
- 9 unique market products for energy and ancillary services

AWS re:Invent

Use Case Challenges



Left with only 180 seconds to:

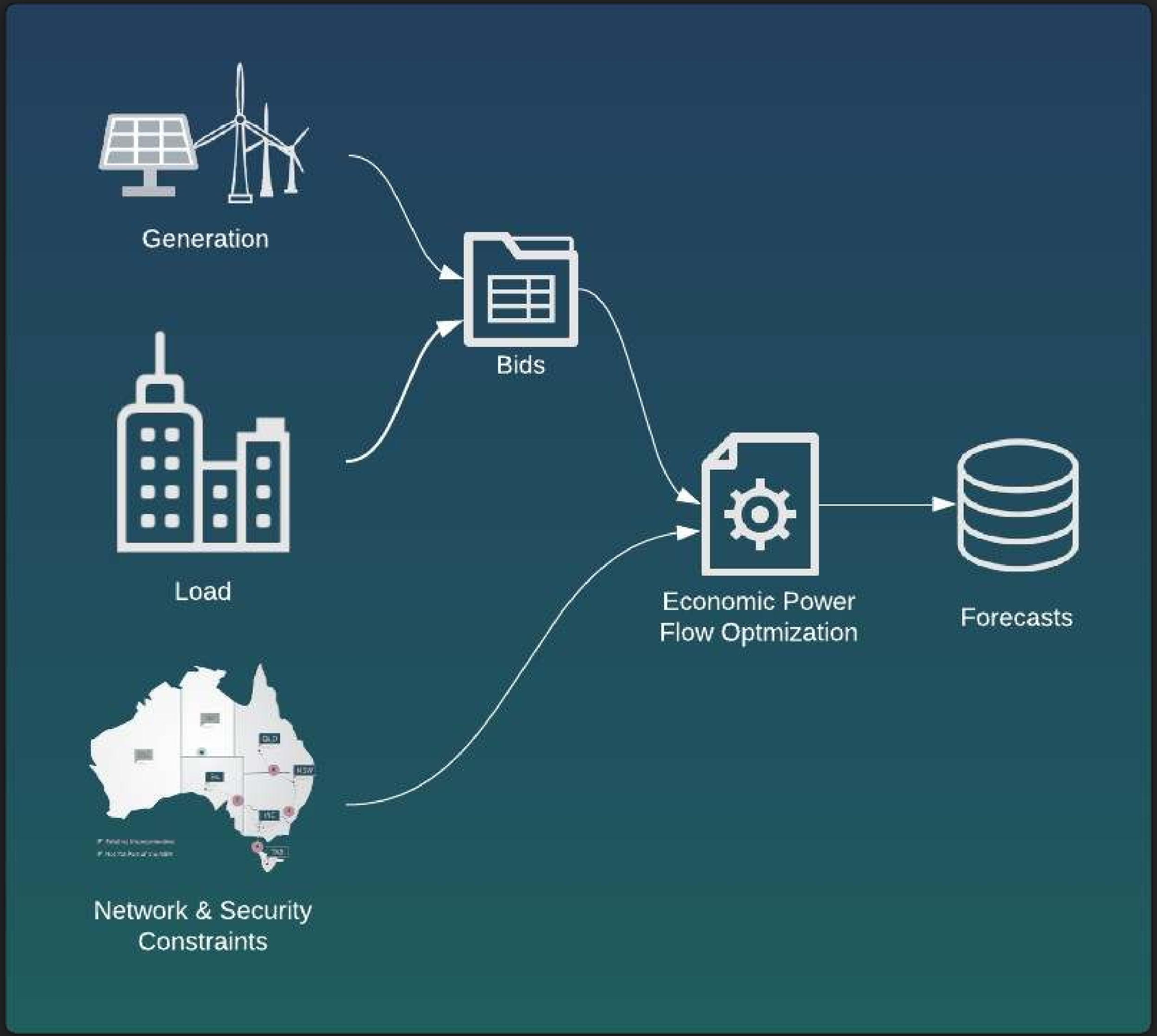
- 0-50s Forecast prices for upcoming trading intervals
- 90s Determine optimal asset dispatch
- 10s Construct competitive market bids
- 20s Present to user for final confirmation
- 10s Deliver to Market Operator

…which we have to repeat every 5 minutes

- Multiple market products
- Considerations across time
- Volatility (timing & magnitude)
- Drift (changing market composition)
- Rapidity (< 20 sec)
- Frequency (5 min)
- Accuracy

Existing Market Forecast, why Machine Learning?

- Spot prices are forecasted by balancing generation and load bids
- Power flow optimization model
- Bids must be supplied through end of day, but can be updated at every market interval (5 minutes)
- Market forecast accuracy is subject to the bidding behavior of market participants



Why Neural Networks?

Complex market dynamics

- Seasonality and common exogenous factors, such as weather
- Network outages & neighboring market conditions

Increasing volatility

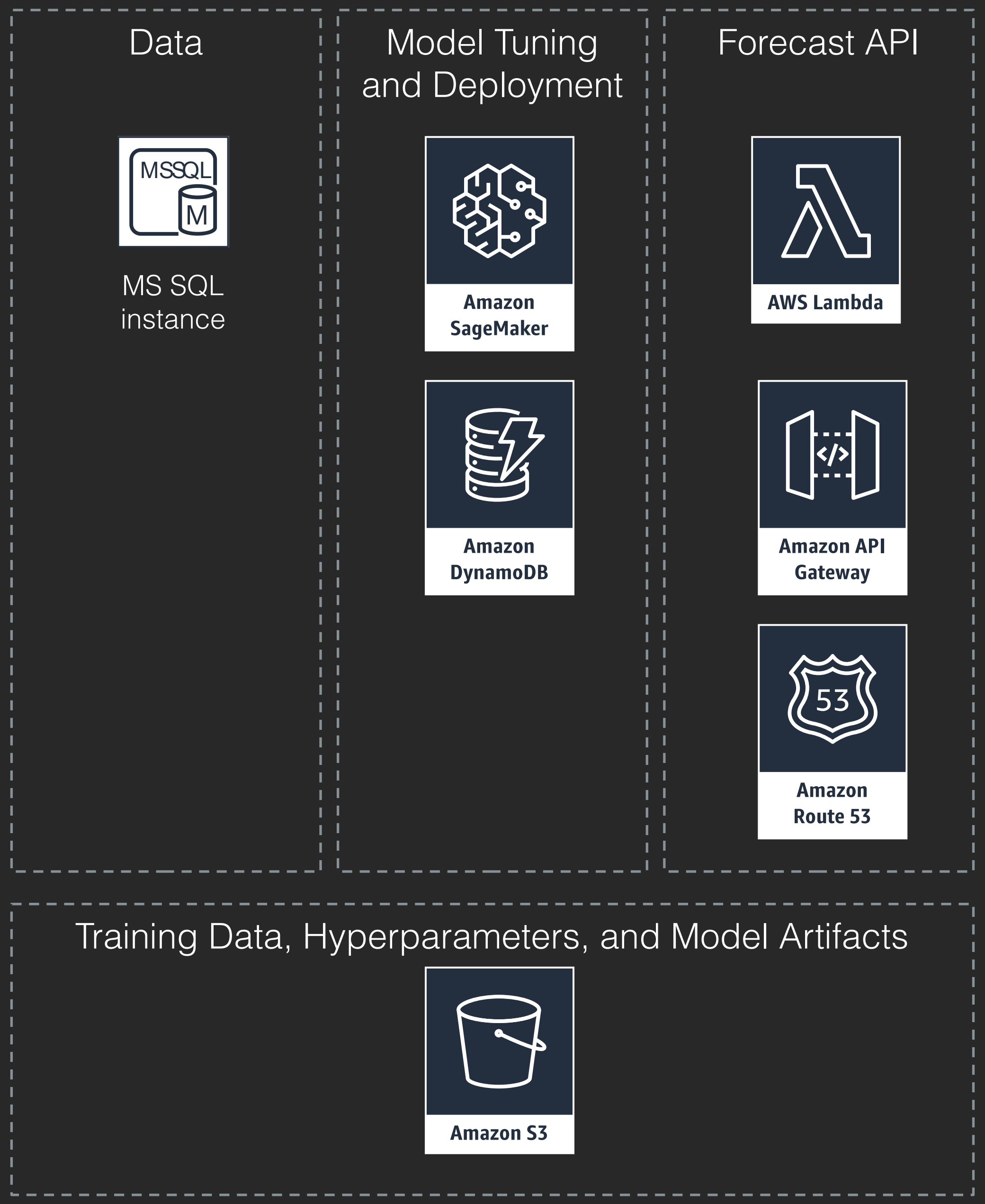
- Changing generation portfolio with increasing penetration of (intermittent) renewable

Lago, Jesus, et al. "Forecasting Spot Electricity Prices: Deep Learning Approaches and Empirical Comparison of Traditional Algorithms." *Applied Energy*, vol. 221, 2018, pp. 386–405., doi:10.1016/j.apenergy.2018.02.069.

Green, Richard, and Nicholas Vasilakos. "Market Behaviour with Large Amounts of Intermittent Generation." *Energy Policy*, vol. 38, no. 7, 2010, pp. 3211–3220., doi:10.1016/j.enpol.2009.07.038.

Architecture Overview

- Data ingestion
- Pre-processing
- Model tuning and deployment via Amazon Sagemaker + TensorFlow + Keras
- Post-processing
- API wraps individual product models used for inference and scenario generation
- Deployed via AWS Chalice



AMS Forecast Machine Learning Model

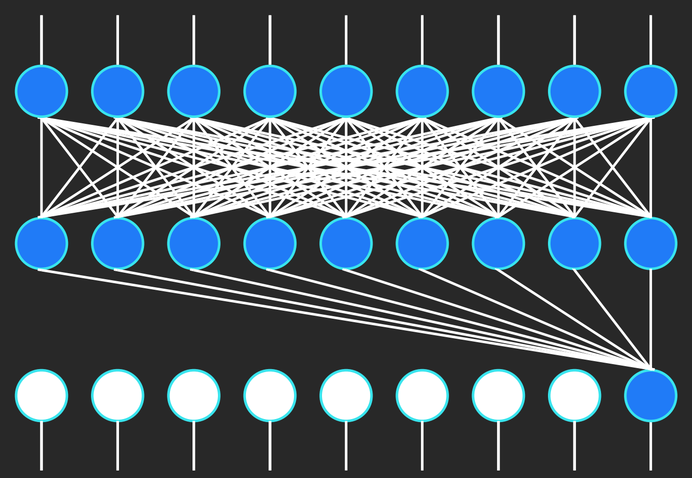
Model design considerations

1. Learn the deviation between market forecast and historical prices
2. Multi-period forecasts of both point estimates and prediction intervals
3. Develop a framework for efficient simulation of price scenarios for stochastic optimization

AWS re:Invent

Benchmark Model

- Learn the deviation between market forecast and historical prices
- Input: market forecast
- Output: multi-step ahead cleared market prices




```
layers = [
    tf.keras.layers.Dense(
        units=units,
        activation=activation,
    ),
    tf.keras.layers.Dropout(
        rate=dropout_rate,
    ),
    tf.keras.layers.Dense(
        units=n_forecast_intervals,
    ),
]
```

Uncertainty Estimation

- Predict both point estimates as well as uncertainty
- Add an output dimension to represent quantiles

```
layers = [
    tf.keras.layers.Dense(
        units=units,
        activation=activation,
    ),
    tf.keras.layers.Dropout(
        rate=dropout_rate,
    ),
    tf.keras.layers.Dense(
        units=n_forecast_intervals,
    ),
]
```

```
layers = [
    tf.keras.layers.Dense(
        units=units,
        activation=activation,
    ),
    tf.keras.layers.Dropout(
        rate=dropout_rate,
    ),
    tf.keras.layers.Dense(
        units=n_forecast_intervals * n_quantiles,
    ),
    tf.keras.layers.Reshape(
        target_shape=(n_forecast_intervals, n_quantiles),
    ),
]
```

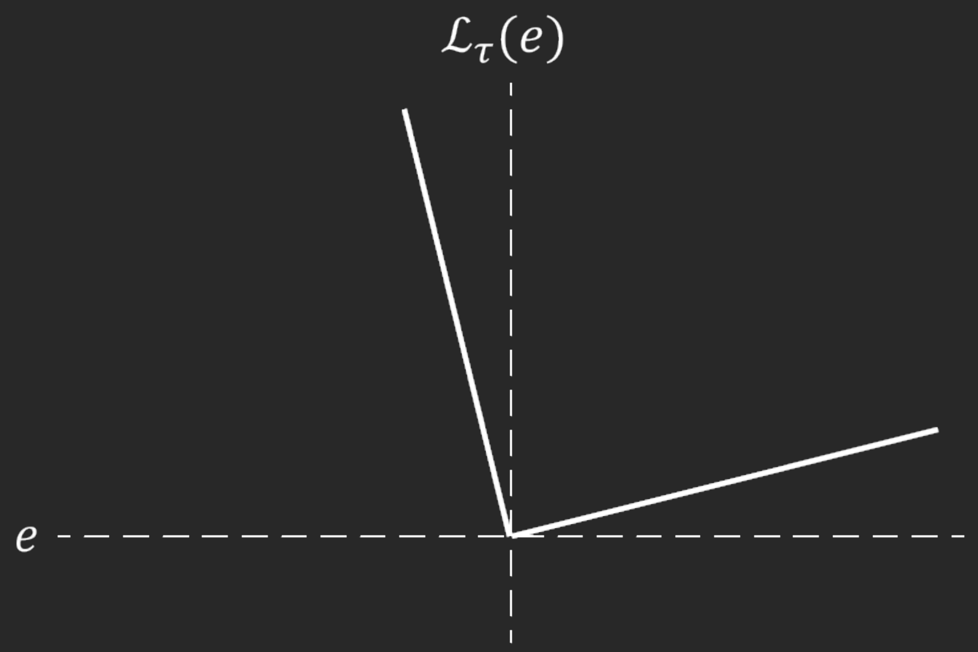
AWS re:Invent

Uncertainty Estimation

- Quantile Regression
- Asymmetric (pinball) loss function conditional to quantile, $\tau \in [0, 1]$

$$e = f(x) - y$$

$$\mathcal{L}_\tau(e) = \begin{cases} -\tau \cdot e, & \text{if } e < 0 \\ (1 - \tau) \cdot e, & \text{else} \end{cases}$$



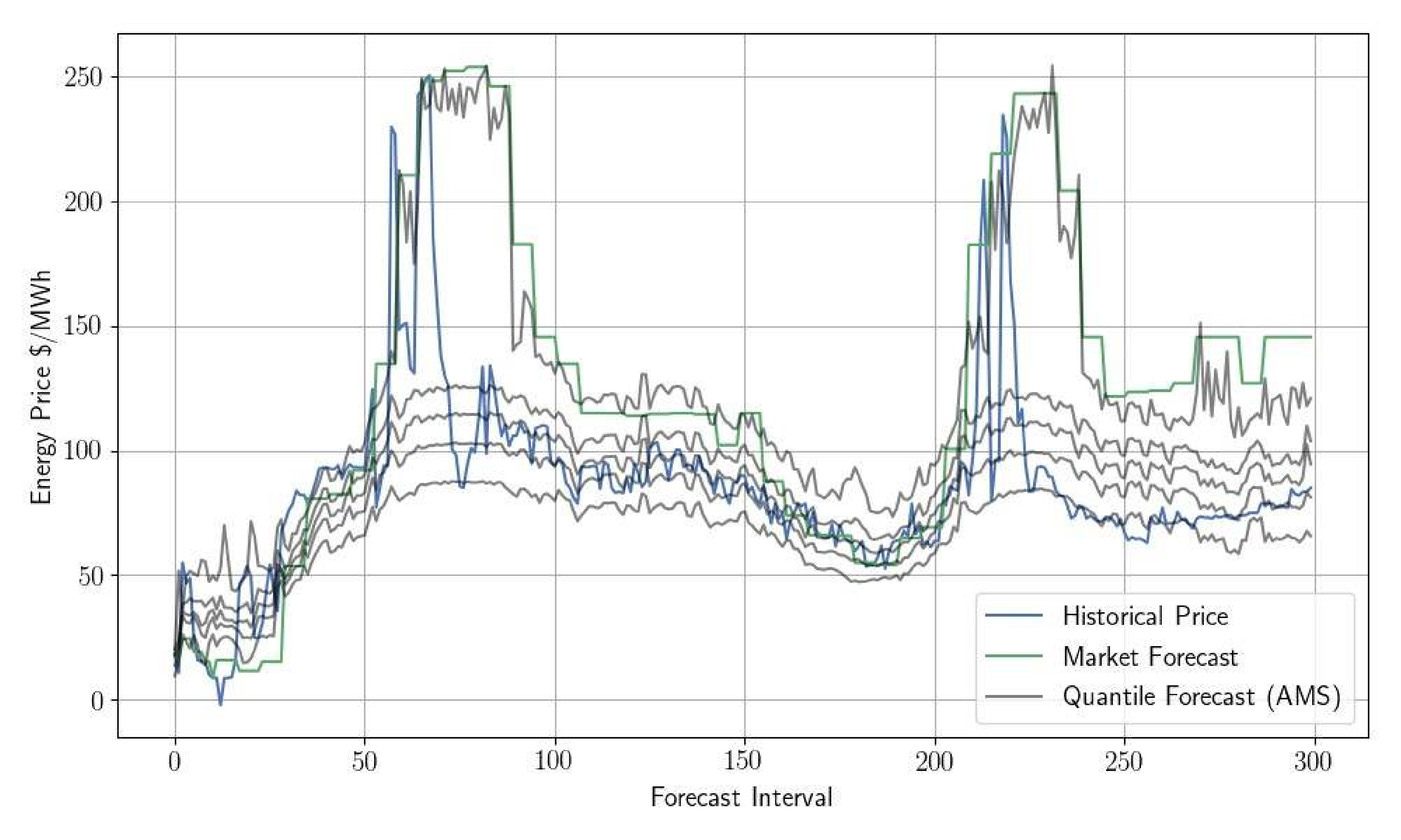
Example:
Under-prediction, $\mathcal{L}_{0.9}(-10) = 9$
Over-prediction, $\mathcal{L}_{0.9}(10) = 1$

```
tf.losses.compute_weighted_loss(
    losses=tf.maximum(
        -quantile * error,
        (1- quantile) * error
    ),
    weights=weights,
    reduction=Reduction.MEAN,
)
```

AWS re:Invent

Example Forecast

- Single inference
- 300 forecast intervals
- 5 quantiles:
  10, 30, 50, 70, 90%

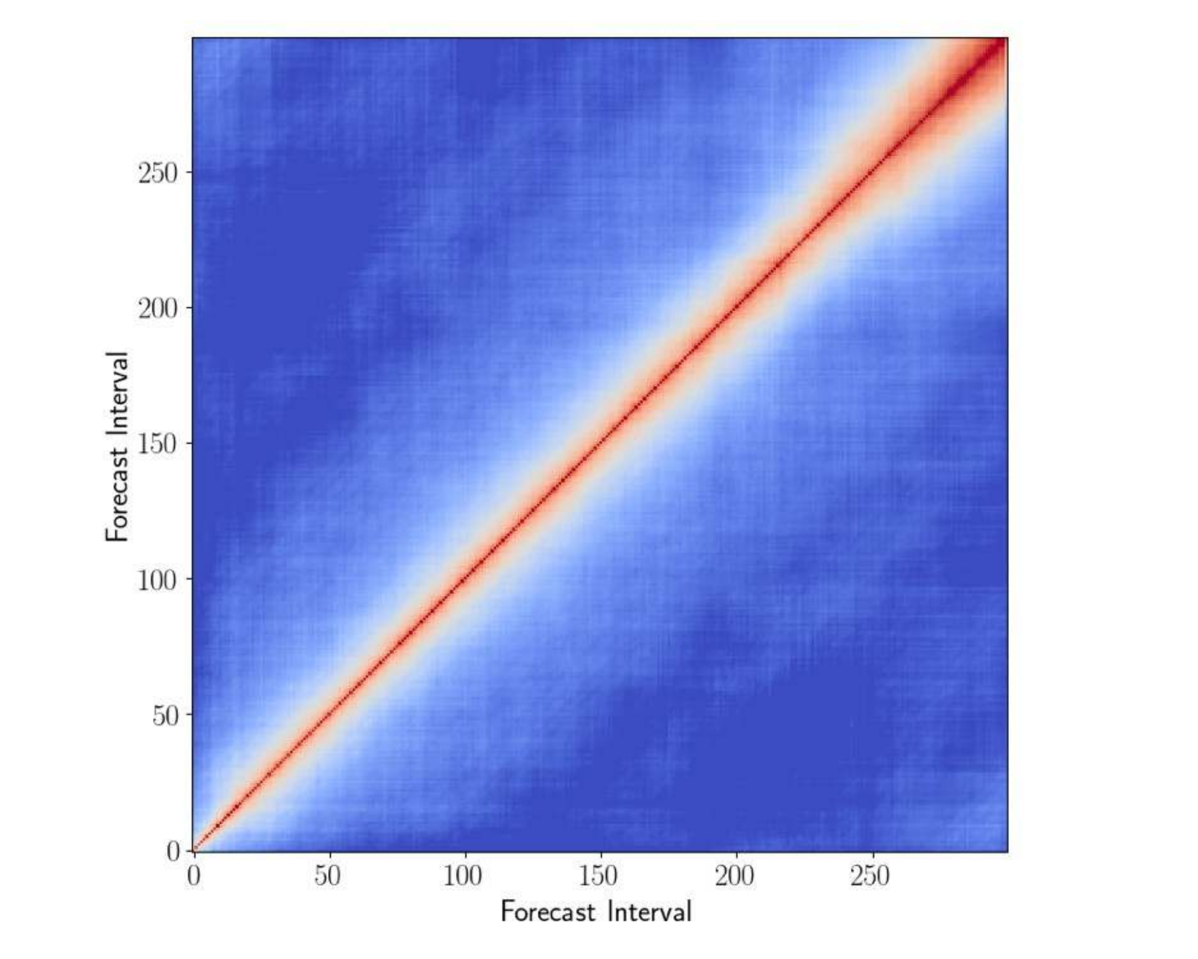


AWS re:Invent

Stochastic Scenario Generation

- Develop a framework for efficient simulation of price scenarios for stochastic optimization
- Use test set to derive temporal covariance
- Now sample from a known distribution to generate realistic price scenarios!

```
np.random.multivariate_normal(
    mean=np.zeros(n_forecast_intervals),
    cov=covariance,
    size=n_scenarios
)
```



AWS re:Invent

Benchmark Model Results

Meets requirements

- Improvement on market forecasts
- Single model, capable of quick quantile estimation and scenario generation

Limitations

- Densely connected model prone to overfitting when additional features are included

Next steps

- Develop a more robust model, less dependent on regularization methods
- Use intuition for feature & target dependencies to reduce model connectivity

Convolutional Network

Dilated convolutional neural network

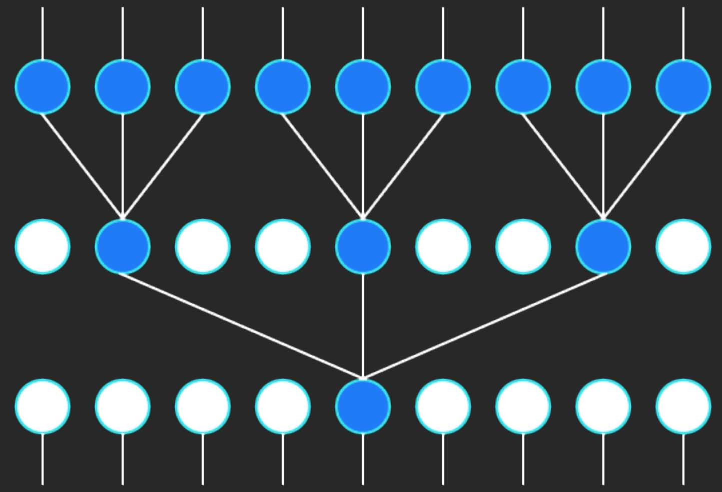
- Example

    number of layers = 3

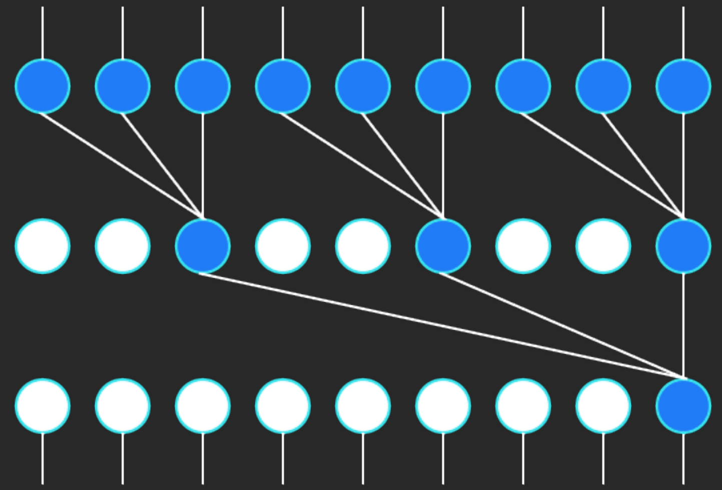
    kernel size = 3

    dilation rate = 3

- Receptive field grows exponentially
- Captures both short and long-term dependencies
- Typically stacked with residual connections



Acausal



Causal

Hyperparameter and Model Tuning

Data parameters

- Missing value imputation method
- Sample weight exponential decay rate

Model architecture

- Filter size
- Kernel size
- Dropout rate
- Dilation rate
- Number of layers (and stacks)
- Skip connections
- Causal vs acausal convolution

Solver parameters

- Learning rate

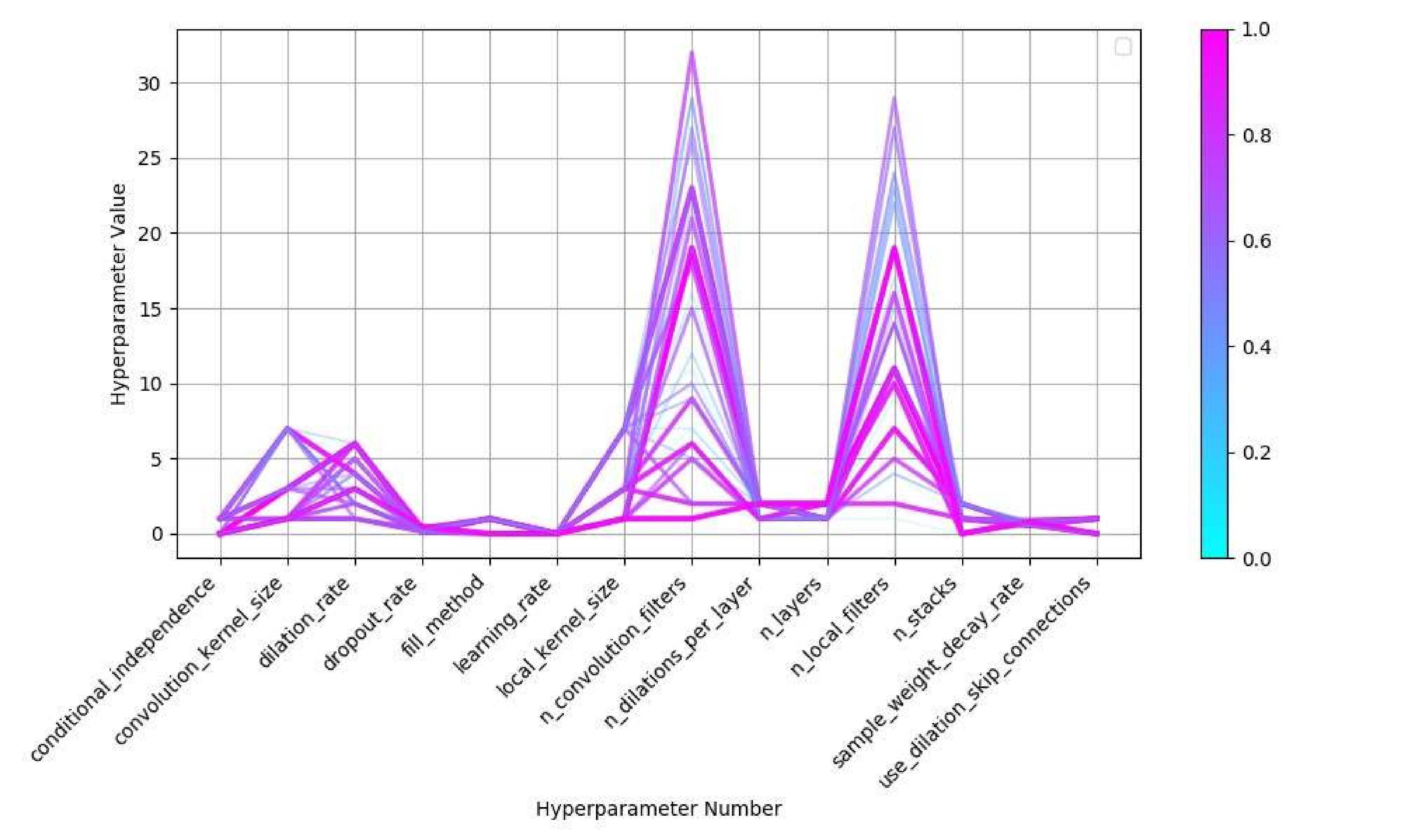
```
tuner = HyperparameterTuner(
    estimator=self.estimator,
    objective_metric_name='validation-loss',

hyperparameter_ranges=hyperparameter_ranges,
    metric_definitions=[{
        'Name': 'validation-loss',
        'Regex': ', loss = ([+-]?\d+\.\d+)',
    }],
    strategy='Bayesian',
    objective_type='Minimize',
    max_jobs=max_jobs,
    max_parallel_jobs=max_parallel_jobs,
    tags=tags,
    base_tuning_job_name=self.job_name,
)
```

AWS re:Invent

Hyperparameter Tuning

Tuning example

- 36 jobs (3 in parallel)
- 2 hours per training job
- 72 training hours
- ml.p2.xlarge, Tesla K80, $1.26/hr
- 20% difference in loss among jobs



AWS re:Invent

Accuracy

- We follow common model validation practices, splitting the data into three chronologically separated groups

  train – model training

  validate – hyperparameter tuning

  test – final comparison metric, empirical error used to derive covariance matrix, and estimation of future performance

- Case Study: South Australia energy prices, 24-hour point estimates

  68% reduction of mean absolute error (against market forecast)

  24% reduction of median absolute error

AWS re:Invent

Takeaways

Without prior experience using Deep Learning tools

- Deployed benchmark TensorFlow model on AWS in weeks
- Learned behavioral market patterns improves upon market-generated forecast
- Extended the model to state-of-the-art temporal dilated convolutional network
- Single forecast API provides quantile forecasts and realistic product-temporal-correlated price scenarios for all queried products

Recommendations

- Start simple, extending the examples provided by AWS
  https://github.com/awslabs/amazon-sagemaker-examples

- Keras allows for quick prototyping

- Parameterize the “art” of Deep Learning architecture and let tuning discover best design

Presenters

Kevin Clifford

Andrew Martinez

Corresponding author

Corey Noone (coreyn@advmicrogrid.com)

Shameless plug

We're hiring data scientists and software developers

Advanced Microgrid Solutions

986 Mission St, 4th Floor

San Francisco, CA 94103

https://advmicrogrid.com

Resources

AWS re:Invent

Resources

https://ml.aws

https://tensorflow.org/
https://keras.io/

https://aws.amazon.com/sagemaker
https://github.com/awslabs/amazon-sagemaker-examples
https://github.com/aws/sagemaker-python-sdk

https://medium.com/@julsimon

AWS re:Invent

Please complete the session survey in the mobile app.

Thank you!

Julien Simon
Principal Tech. Evangelist, AI/ML
Amazon Web Services
@julsimon

Kevin Clifford
Senior Product Manager
Advanced Microgrid Solutions

Andrew Martinez
Staff Research Scientist
Advanced Microgrid Solutions

AWS re:Invent