Quick to production with the best of Spark and Tensorflow

Ronny Mathew
Manager, Data Science
Rue Gilt Groupe

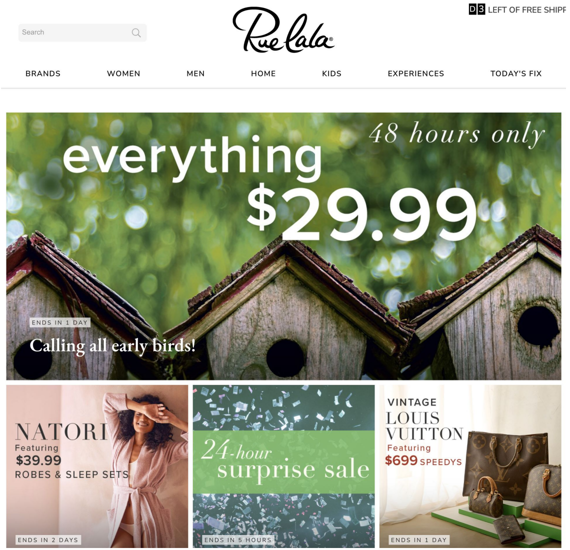
Who are we

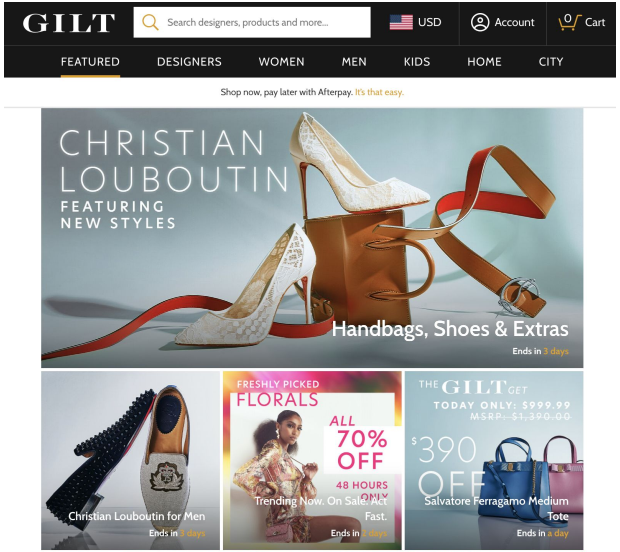
RUE GILT GROUPE

Rue Gilt Groupe is a fashion technology company based in Boston

Our **50M+ members** get exclusive access to **private sales** on **must have brands** from an ever changing catalog of **3M+ products**

ruelala.com





gilt.com

Data Science at Rue Gilt Groupe (RGG)

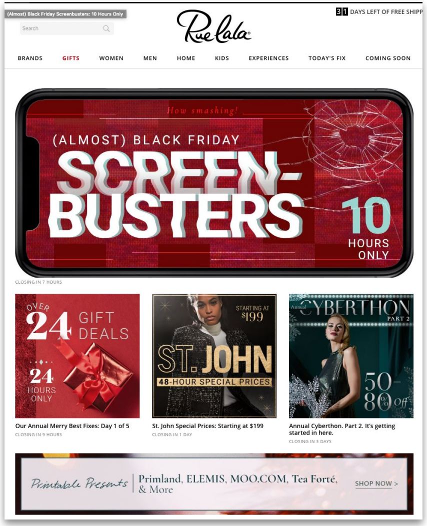
RUE GILT GROUPE

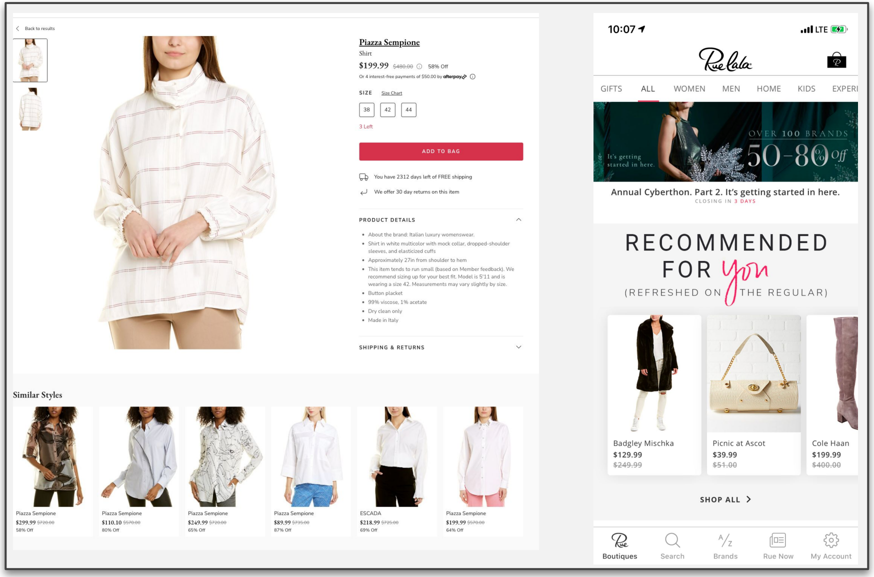
Close-knit group of passionate data scientists who thrive on collaboration and innovation

Working on various DS projects including **NLP, CV, time-series Forecasting, Recommenders**

We use ML to **personalize** the site and email experiences of our members among others projects

Sorting main pages





Product recs

Our 2020 ML stack

RUE GILT GROUPE

We leverage Spark and container based ML pipelines extensively

In addition to this, we use common ML packages like sklearn, xgBoost, openCV, prophet, ...

Developing next-gen Models

RUE GILT GROUPE

Developing the next generation of products pushed us to expand ***beyond Spark ML*** *and leverage state-of-the-art deep learning models in* ***Tensorflow***

- ➢ Solution to **quickly** move from a working prototype to a production model
- ➢ Did not want added complexity with more MLOps code
- ➢ **Minimal** approach perfect for a **small-medium** full stack Data science team
- ➢ Enable **rapid prototyping and experimentations**
- ➢ Lets Data Scientists and ML Engineers **focus more on research** and **actual ML tasks**

What we will cover

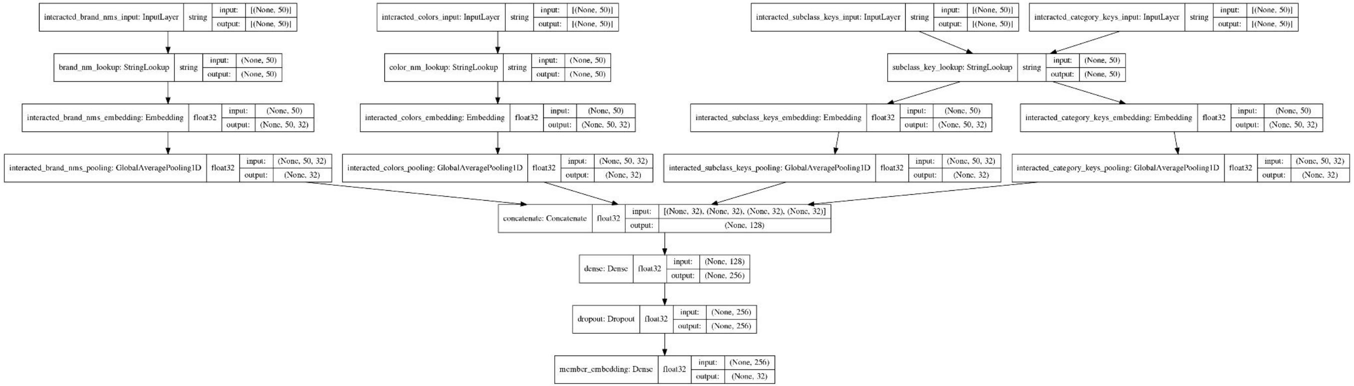
RUE GILT GROUPE

- This talk will focus on how we were able to achieve all of the above using some recent advancements in Tensorflow 2 and extension libraries for Spark by merging the **best of both Spark and Tensorflow worlds**
- **Enable Quick Distributed training** in tensorflow on GPU and CPU clusters
- MLFlow model registry as a **central model storage** ensuring consistency for all downstream batch and realtime inference jobs

How this helped us at RGG

RUE GILT
GROUPE

- Models using huge clickstream datasets are difficult to train on a single instance
- Developed a prototype for a **GRU based personalization model for new members**
- Converted our prototype to a production model quickly in **less than 2 weeks**
- A/B Test proved our new members loved their deep learning recs with **>25% lift in CTR**!

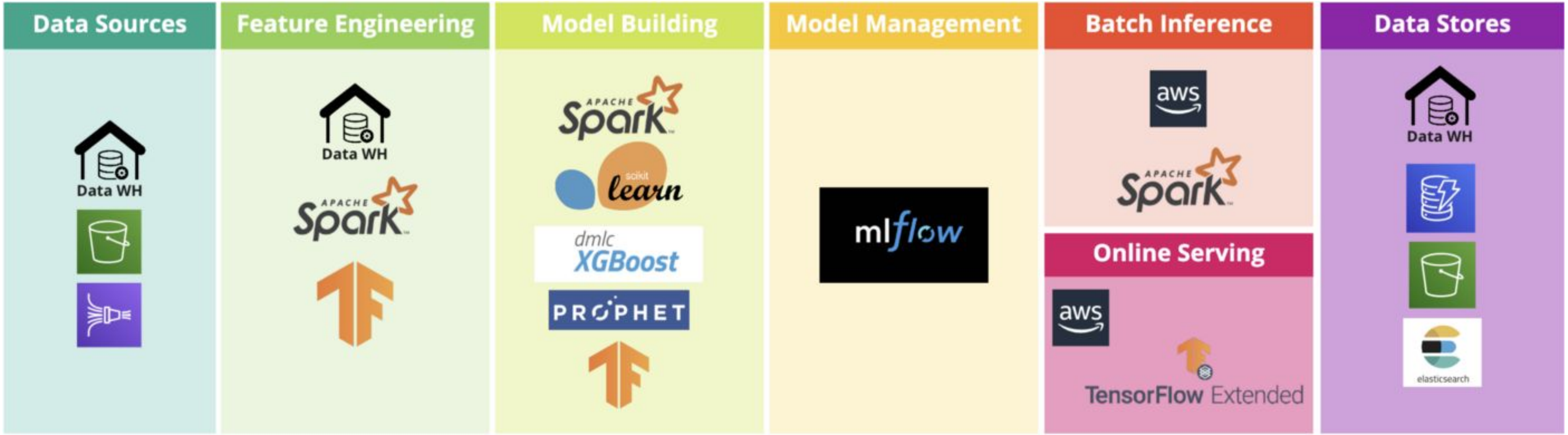


Our 2021 next-gen ML stack

RUE GILT GROUPE

Using the tools we will cover today, we successfully integrated Tensorflow into our spark world

Flexibility to choose from Spark or Tensorflow ecosystem at any ML stage



Next up for us

RUE GILT GROUPE

- Develop deep learning models for all of our personalization and recommenders
- Online session based recommendations
- Better text representation with cutting edge language models (BERT and more)
- Product Tagging and Catalog Management with NER, text and image models
- Leverage image resources better with Autoencoders and vision models
- Online recommendations using Reinforcement Learning
- Realtime intent detection for a user session

Let’s get started with some Spark features

Apache Spark

- Popular open source big data processing platform
- Write code in multiple programming languages
- Run pipelines on a single-node or clusters
- Work on structured and unstructured data
- Batch and streaming pipelines
- Distributed Machine learning with Spark ML

Data Pipelines in multiple languages

- Easy-to-use APIs in SQL, Python, Java, R, and/or Scala
- User-defined functions extend spark SQL functionalities

Data engineers

```
# in python
import pyspark.sql.functions as f
@f.udf("string")
def say_hello(name: str) -> str:
     return f"Hello {name}"
sqlContext.udf.register("say_hello",
say_hello)
```

Data Analysts/Data Scientists

```
-- uses the udf in sql
SELECT say_hello('Bob')
```

Pandas UDFs

Allows using a pandas UDF to manipulate spark dataframes

Manipulate spark dataframe as pandas DataFrames instead of a per row transformation

```
@f.pandas_udf("string")
def simple_udf(iterator: Iterator[pd.Series]) -> Iterator[pd.Series]:
     for x in iterator:
          yield pd.Series(list(map(lambda r: r + "1", x)))

spark.range(10).select(simple_udf("id")).show()
```

[Read more about them here]()

Spark ML

- Scalable implementations of many common ML algorithms, including
    - Traditional ML models like Regression, Trees, Multilayer Perceptron, Naive Bayes
    - Text models like Bag of words, Word2Vec, TF-IDF, LDA
    - ALS-based Collaborative Filtering and FP-Growth mining
    - Locality Sensitive Hashing and Random Projection for nearest neighbor lookups
- Libraries are available to easily integrate sklearn, XGBoost, tensorflow, pytorch and others

Let's look at some Tensorflow features

Tensorflow

- Machine learning framework maintained and used within Google
- V2 has adopted the core principle of **progressive disclosure of complexity** from Keras

  You should always be able to get into lower-level workflows in a gradual way. You shouldn't fall off a cliff if the high-level functionality doesn't exactly match your use case. You should be able to gain more control over the small details while retaining a commensurate amount of high-level convenience. source

- TFx extends tensorflow beyond model building and to end to end ML pipelines
- A lot of extension libraries that extends Tensorflow capabilities
- Great tutorials and guides on the official website for anyone to get started quickly
  - https://www.tensorflow.org/tutorials
  - https://www.tensorflow.org/guide

TensorFlow

TF Data

- TF APIs for data loading and manipulation using Tensorflow datasets
- Recommended way to load data into a tensorflow 2.x model
- **TFRecordDataset** is a tf dataset that can load data directly from **tfRecord** files
- Does a lot of great optimizations behind the scenes including prefetching and caching
    - While training on GPU, prefetch uses the CPU to load the next n batches
    - **tf.data.AUTOTUNE** can automatically adapt prefetching (and others) to your cluster

```
dataset = (tf.data.TFRecordDataset(filenames=[list of tfrecord files])
                      .batch(1000)
                      .shuffle(10000)
                      .prefetch(tf.data.AUTOTUNE))
```

TF Distribute

- Provides distributed training strategies in synchronous/asynchronous mode, on GPU/TPU, etc
- Automatically assigns **default strategy** to every training loop
- **A single-node development code can be converted to a distributed code by adding a few lines initially when we start development**

```
# we can also add more conditions here, eg check for num workers
# add more strategies as needed
if tf.config.list_physical_devices('GPU'):
   my_strategy = tf.distribute.MirroredStrategy()
else:   # Use the Default Strategy
   my_strategy = tf.distribute.get_strategy()

with my_strategy:
   # wrap all the training code within this scope
   ...
```

Distributing Datasets

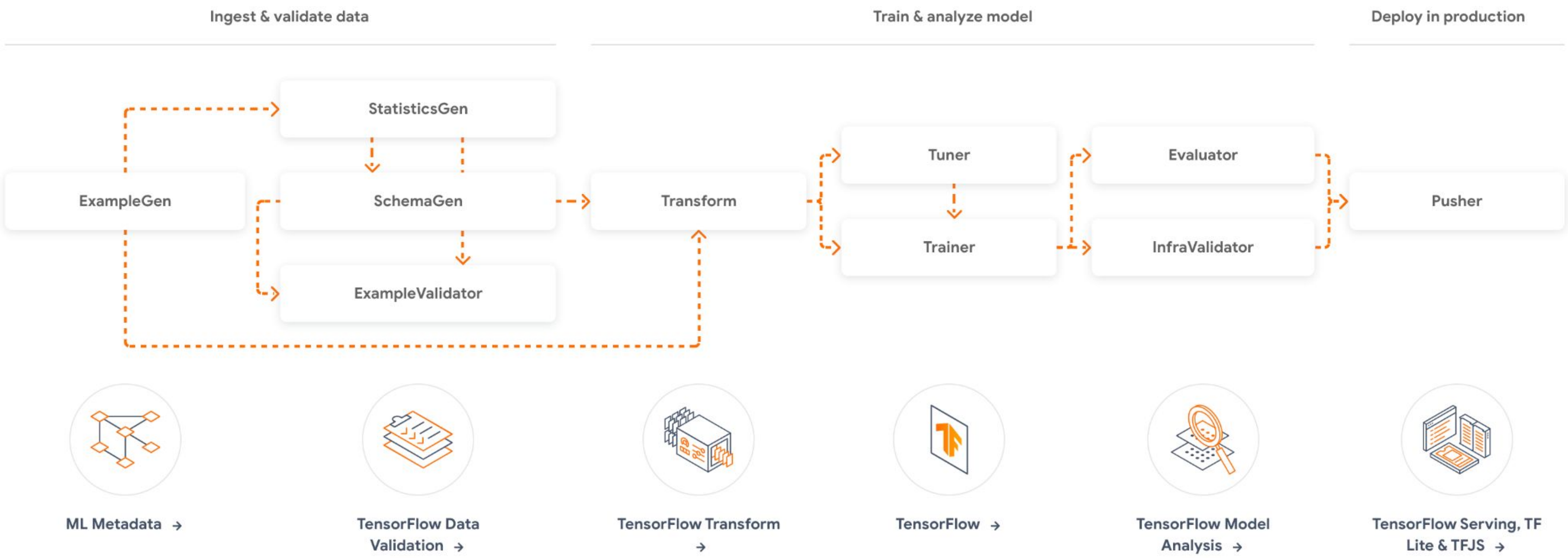
TensorFlow

Convert a tf dataset to a distributed dataset in a few lines of code

```
# There are two shard policies, DATA and FILE
options = tf.data.Options()
options.experimental_distribute.auto_shard_policy =
tf.data.experimental.AutoShardPolicy.DATA
dataset.with_options(options)
```

TF Extended (TFx)

- End-to-end machine learning platform for tensorflow
- Everything from data ingestion to model serving and monitoring



source

Some cool TFx components

TensorFlow

- **Tensorflow Serving**
  - Core Tensorflow component responsible for deploying and online serving
  - Loads and serves a model at a high performant REST endpoint
  - Docker containers provided by google with different flavors of tensorflow
  - Can be scaled as needed using dockerized containers for online recommendations
- **Tensorflow Transform**:
  - Performs feature engineering and transforms
  - Alternative to using Spark (as in our case)
- **Tensorflow Model Analysis (TFMA)**
  - Model validations
  - Monitoring issues and drifts in models

TF Recommenders

TensorFlow

- Extension library for building Deep learning Recommenders
- Provides easy to use wrappers for models, custom metrics and losses
- Exposes google's *scann* approximate nearest neighbor engine as a Layer
- Simplifies building models for
  - Retrieval (narrow down entire corpus to a few thousand candidates)
  - Ranking (rank top candidates to a handful with more complex architecture)
  - Multitask recommenders (learn from multiple objectives)
  - Deep Cross Networks (capture explicit feature interactions between items)
  - and more

More Extension Libraries and tools

TensorFlow

- TF Ranking
    - Build Learning-to-Rank (LTR) models
    - commonly used ranking loss functions like pointwise, pairwise, and listwise
    - ranking metrics like Mean Reciprocal Rank and NDCG
- TF Agents
    - Enables building contextual bandit and RL agents
- TF Hub
    - Collection of the latest pre-trained models for text, image, video, audio, and more
    - Easily pull to any existing model as a layer with simple code,

```
# sometimes we also need to load preprocessing layers
bert_path="https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/4"
bert_embeddings = hub.KerasLayer(bert_path, trainable=False)
```

MLFlow

MLFlow

- Open source platform for ML lifecycle
- Enables Experiment tracking, reproducibility, deployability and model management
- Works with most common ML libraries
- Seamlessly integrates with Spark and tensorflow

MLflow Tracking

Record and query experiments: code, data, config, and results

Read more

MLflow Projects

Package data science code in a format to reproduce runs on any platform

Read more

MLflow Models

Deploy machine learning models in diverse serving environments

Read more

Model Registry

Store, annotate, discover, and manage models in a central repository

Read more

source

MLFlow Tracking

- Track Hyperparameters, Metrics and Artifacts associated with each and every run
- **Compare metrics** from different runs and use that to **select** the best model
- **Autologging** capabilities for common frameworks

▾ Parameters

Name	Value
cross val horizon	14 days
cross val initial	730 days
cross val period	10 days
cross validation end	03/21/2021
cross validation start	03/22/2017
prophet changepoint_prior_scale	0.05
prophet mcmc_samples	0
prophet seasonality_mode	multiplicative
train months	25

▾ Metrics

Name	Value
coverage	0.581
cross val avg coverage	0.733
cross val avg mae	74229.3
cross val avg mape	0.086
cross val avg mse	9557433561.4
cross val avg rmse	97349.4
horizon 10 days coverage	0.82
horizon 10 days mae	67163.5
horizon 10 days mape	0.072

▾ Artifacts

- ▸ config
- ▸ data
- ▸ model_runs

MLFlow Model Registry

- **Manage models** using MLFlow models and Model Registry
- Transition model versions between Staging, Production and Archived stage
- Single source of truth for all models and its versions ensuring **consistency**

Registered Models

Name	Latest Version	Staging	Production
Forecast Test	Version 1	–	–
pricing_demand_model	Version 84	Version 33	Version 84
pricing_demand_model_backtest	Version 70	Version 70	Version 16
residual_pricing_demand_model	Version 182	–	Version 174
residual_pricing_demand_model_backtest	Version 182	Version 182	–

Linking them all

TF records with Spark

- TFRecord is a tf native file format that stores sequences of binary records as **Examples**
- Reading and writing TFRecords from Spark ensures that Spark and TFx uses same format
- Enables the flexibility of using Spark or TFx at any point of our ML lifecycle
- Other approaches like Petastorm achieves similar goals but loses the interoperability with TFx

```
# Write tfRecords as Example from a Spark Dataframe
some_df.write.format("tfrecords")
       .option("recordType", "Example")
       .mode("overwrite")
       .save(save_path))
```

Spark Tensorflow Distributor

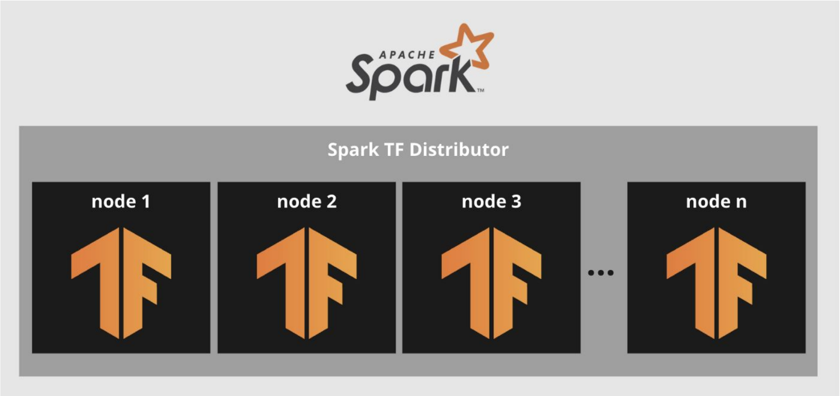
- Part of the TensorFlow ecosystem
- Enables running **tf.distribute strategies** from inside Spark jobs
- **Delegates** cluster management and communication to **Spark**
- Also runs on CPU clusters
- Straightforward usage,
  - Define both dataset loading and TF model building code in a **train** method
  - Pass the number of workers in the cluster and train method to the distributed runner

```
num_workers = 10
# add local_mode=True for local testing
# you can also specify another strategy in the train method and turn on
custom_strategy=True
MirroredStrategyRunner(num_slots=num_workers).run(train)
```

Spark Tensorflow Distributor

Under the hood this library automates a lot of infrastructure code

- Communication between nodes
- designation of chief and worker nodes
- Supports custom_strategy to use other tf.distribute strategies (TPUs, GPUs, etc)
    - Update train method with new strategy, add custom_strategy=True to the runner



Who is the chief here

- At times we need to know what the role of a node is, especially which one is chief
- Treat callbacks and other operations differently on chief
    - Don’t need all the workers (over)writing the model checkpoints to the same path!
- tf.distribute works by specifying a tf_config environment variable on each node
    - config also includes a node index where *node 0 is the chief node*
- We can find the role of a node using this code,

```
if 'TF_CONFIG' in os.environ:
    tf_config = json.loads(os.environ['TF_CONFIG'])
    node_index = tf_config['task']['index']
    is_chief = node_index == 0
```

Dont forget to save your models!

Since the training happens on the workers and not just a single instance,

we don't have access to the final trained model unless we handle it explicitly!

That means you might be running training for days *but there would be no way to retrieve your final model (yikes!)*.

We can use ModelCheckpoint or BackupAndRestore callbacks to handle this

```
save_best_callback = ModelCheckpoint(
                        filepath=s3://persist_path, # Somewhere on persistent
                        storage
                        monitor=monitor_metric,
                        save_best_only=True)
```

Batch Inference in Spark

- Pull the latest model from model registry
- Load it in a pandas UDF
- Define a schema for the UDF output
- Simply use the new UDF in your inference pipeline as you would use any spark function
- We can also use MLFlow function to create a UDF for a model with `mlflow.pyfunc.spark_udf`

Recap of our new pipeline

ML Stage	*How*
Data Ingestion	Use Spark to read from data sources
Feature Engineering	Create data pipelines in Spark Write data to persistent store as TFRecords
Model Building	Read TFRecords in distributed code from persistent store Train distributed tensorflow model on Spark Save trained model to MLFlow
Model Selection and Management	Pick the best model from MLFlow and add to model registry Use the same production model in all downstream applications
Batch Inference	Load model from MLFlow registry into a pandas UDF Add the UDF into the inference Spark pipeline

Unlocking Realtime Inference

- Realtime inference enables exciting use cases involving direct feedback from our members
- Leverage Spark Streaming pipelines to process the realtime data
- Build an online inference service using TF serving and the model from model registry
- Some use cases are,
  - In-session Recommendations
  - Intent and shopping pattern detection
  - Location based trending products
  - Hyper personalization using Reinforcement Learning

Pros of this Approach

- Great for smaller teams who want to focus on ML and less infrastructure
- Quickly transition from prototype to production models
- Flexibility to choose right framework for the use case at any stage of the ML lifecycle
- Scalable deep learning pipelines with minimal MLOps code
- Run distributed deep learning loads on Spark GPU/CPU clusters with fault tolerance
- Use all of tensorflow ecosystem and extension libraries
- Central storage, versioning, and management of models using MLflow

Open Challenges

- Distributed training doesn’t handle fault recovery automatically (in the case of a dead worker)
    - Parameter server strategy could be a better fit for this case
- Tuning and finding the right amount of nodes for the cluster still requires some manual effort
- Cryptic and messy Tensorflow error messages
- APIs between Keras and Tensorflow are in a state of flux

Q&A

Thank you!