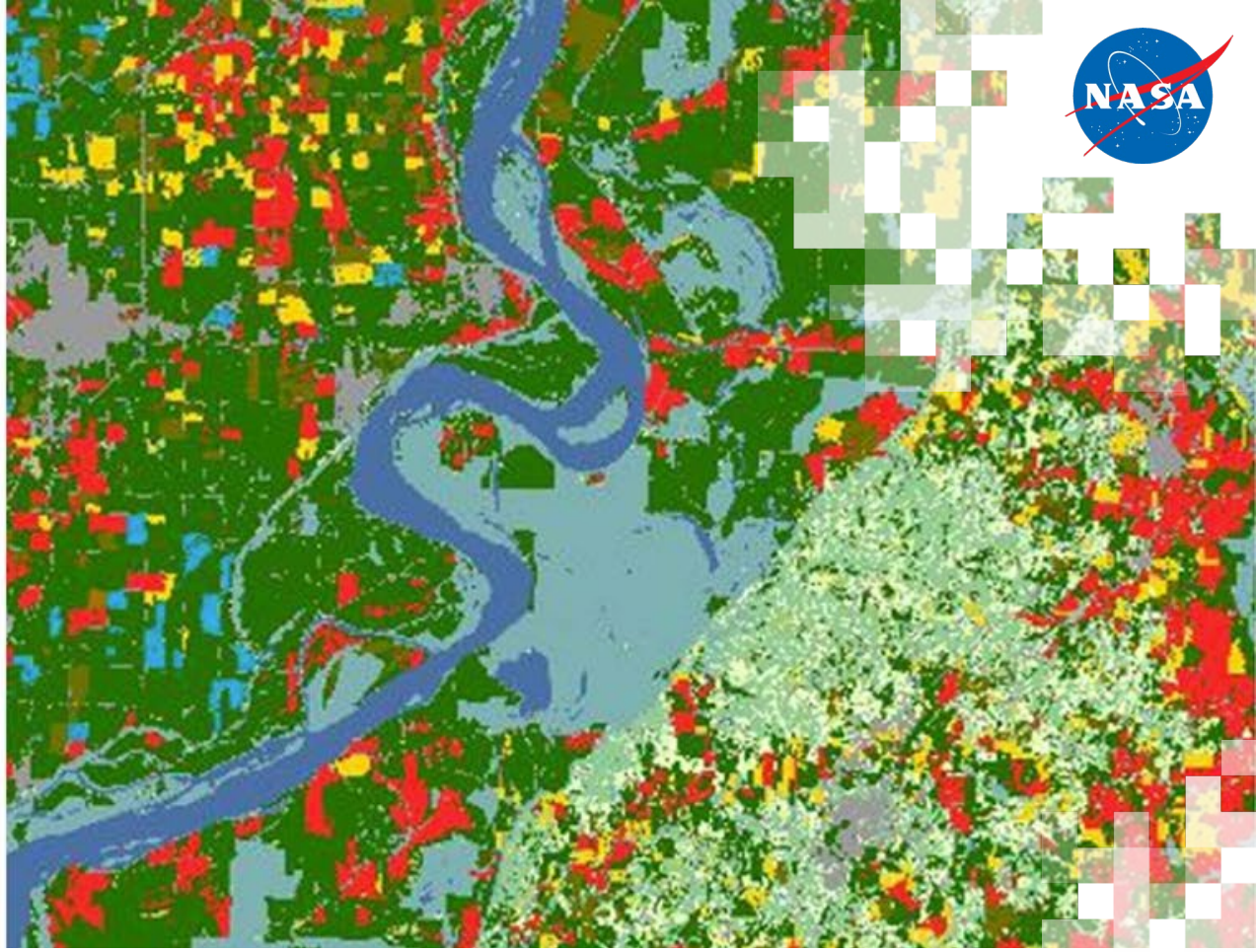
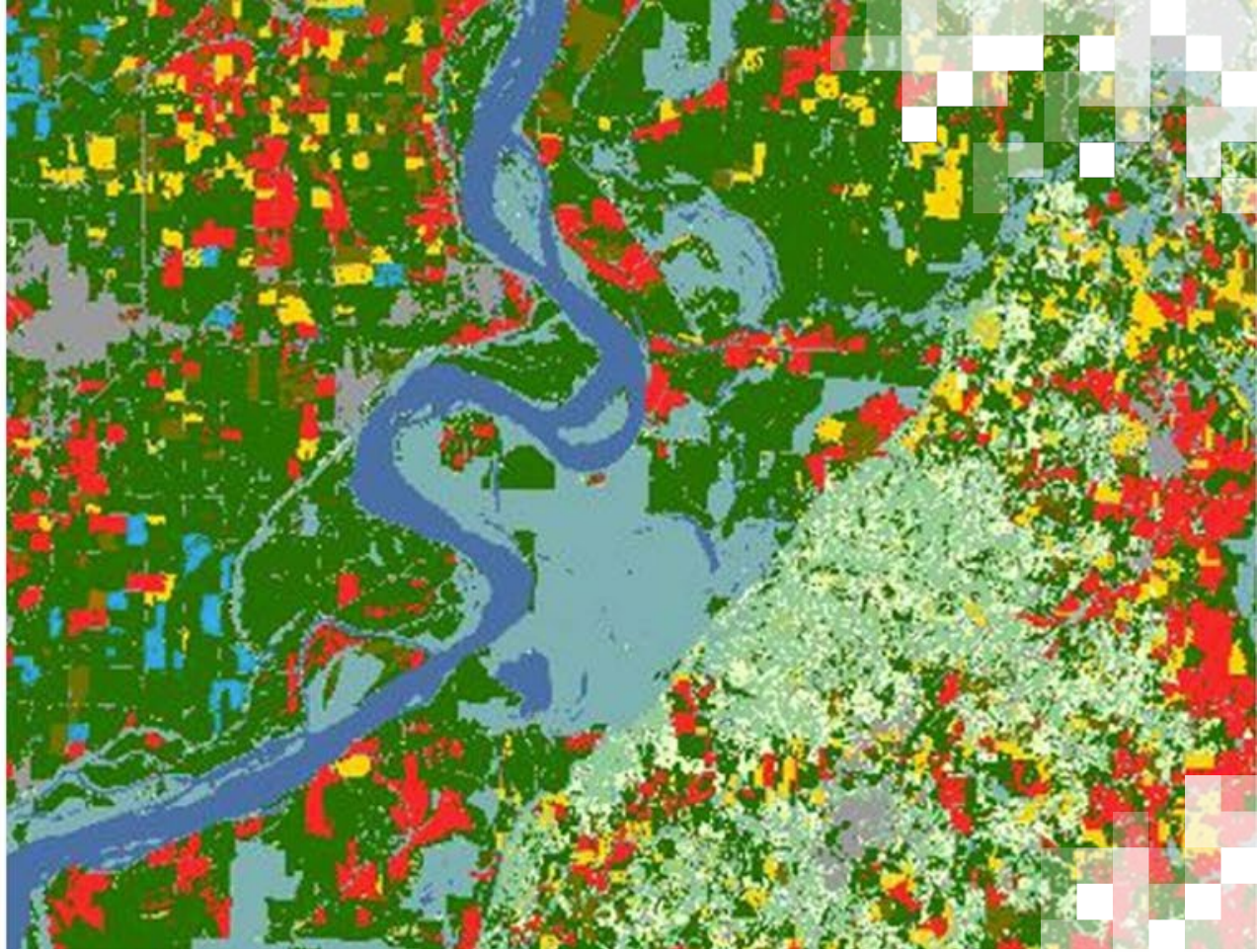


Large Scale Applications of Machine Learning using Remote Sensing for Building Agriculture Solutions

Part 3: Training & Testing ML Models for Irregularly-Spaced Time Series of Imagery

John Just (Deere & Co., Iowa State University), Erik Sorensen (Deere & Co.)

March 19, 2024

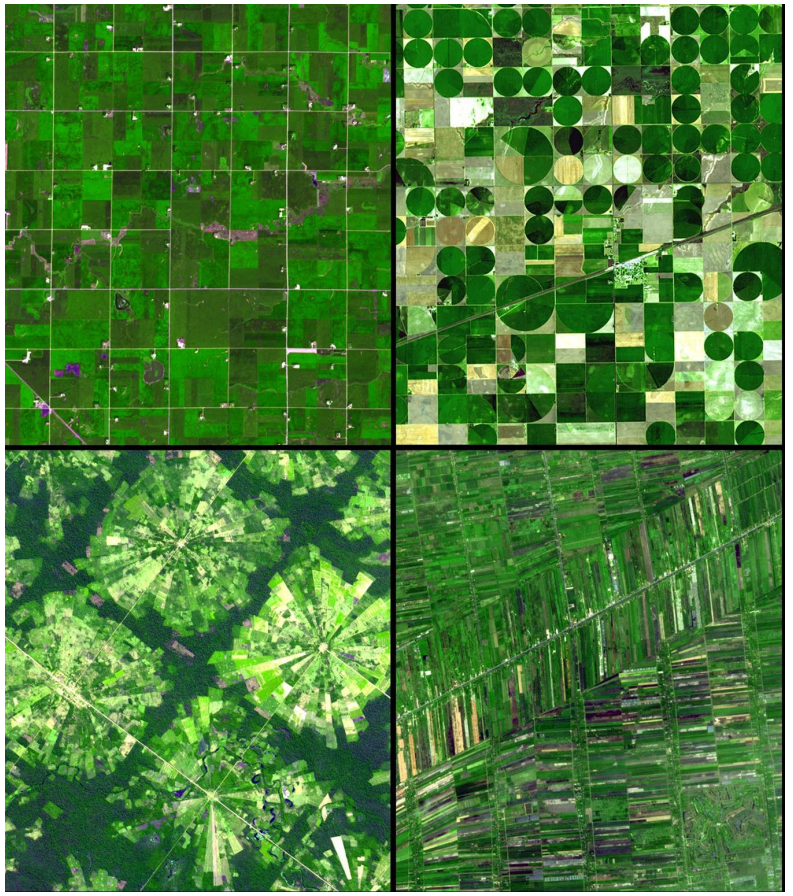


Large Scale Applications of Machine Learning using Remote Sensing for Building Agriculture Solutions

Overview

Motivation for Training

- Timely and accurate in-season crop maps at local to regional scales is crucial for agricultural decision-making and management.
- Irregularly-spaced time-series are common with optical satellite images.
- Training robust models on remote sensing data often requires very large data, but processing and training is complex.
- The Cropland Data Layer (CDL, USDA–NASS) only gives estimates of the types of crops released to the public a few months after the end of the growing season, and not their sequence or timing (e.g., for double crops).



Montage of images shows differences in field geometry and size in different parts of the world. Image credit: NASA (Instrument: Terra – ASTER)

Training Learning Objectives

By the end of this training series, participants will be able to:

- Use recommended techniques to download and process remote sensing data from Sentinel-2 and the Cropland Data Layer (CDL) at large scale (> 5GB) with cloud tools (Amazon Web Services [AWS] Simple Storage Service [S3], Databricks, Spark/Pyspark, Parquet).
- Produce interactive plots of maps, tables, time series, etc. for investigation & verification of data and models.
- Filter data from both the measured (satellite images) and target (CDL) domains to serve modeling objectives based on quality factors, land classification, area of interest (AOI) overlap, and geographical location.
- Build training pipelines in TensorFlow to train machine learning algorithms on large scale remote sensing/geospatial datasets for agricultural monitoring.
- Utilize random sampling techniques to build robustness into a predictive algorithm while avoiding information leakage across training/validation/testing splits.

Prerequisites

- Fundamentals of Remote Sensing
- Crop Classification with Time Series, Part 2
- Sign up for and access Databricks Community Edition

Training Outline

Part 1

Data Preparation of Imagery for Large-Scale ML Modeling

March 05, 2024

Time

Part 2

Data Loaders for Training ML Models on Irregularly-Spaced Time-Series of Imagery

March 12, 2024

Time

Part 3

Training & Testing ML Models for Irregularly-Spaced Time Series of Imagery

March 19, 2024

Time

Homework

Opens March 19 – **Due April 1** – Posted on Training Webpage

A certificate of completion will be awarded to those who attend all live sessions and complete the homework assignment(s) before the given due date.

How to Ask Questions

- Please put your questions in the Questions box and we will address them at the end of the webinar.
- Feel free to enter your questions as we go. We will try to get to all the questions during the Q&A session after the webinar.
- The remainder of the questions will be answered in the Q&A document, which will be posted to the training website about a week after the training.

Part 1 – Trainers

John Just

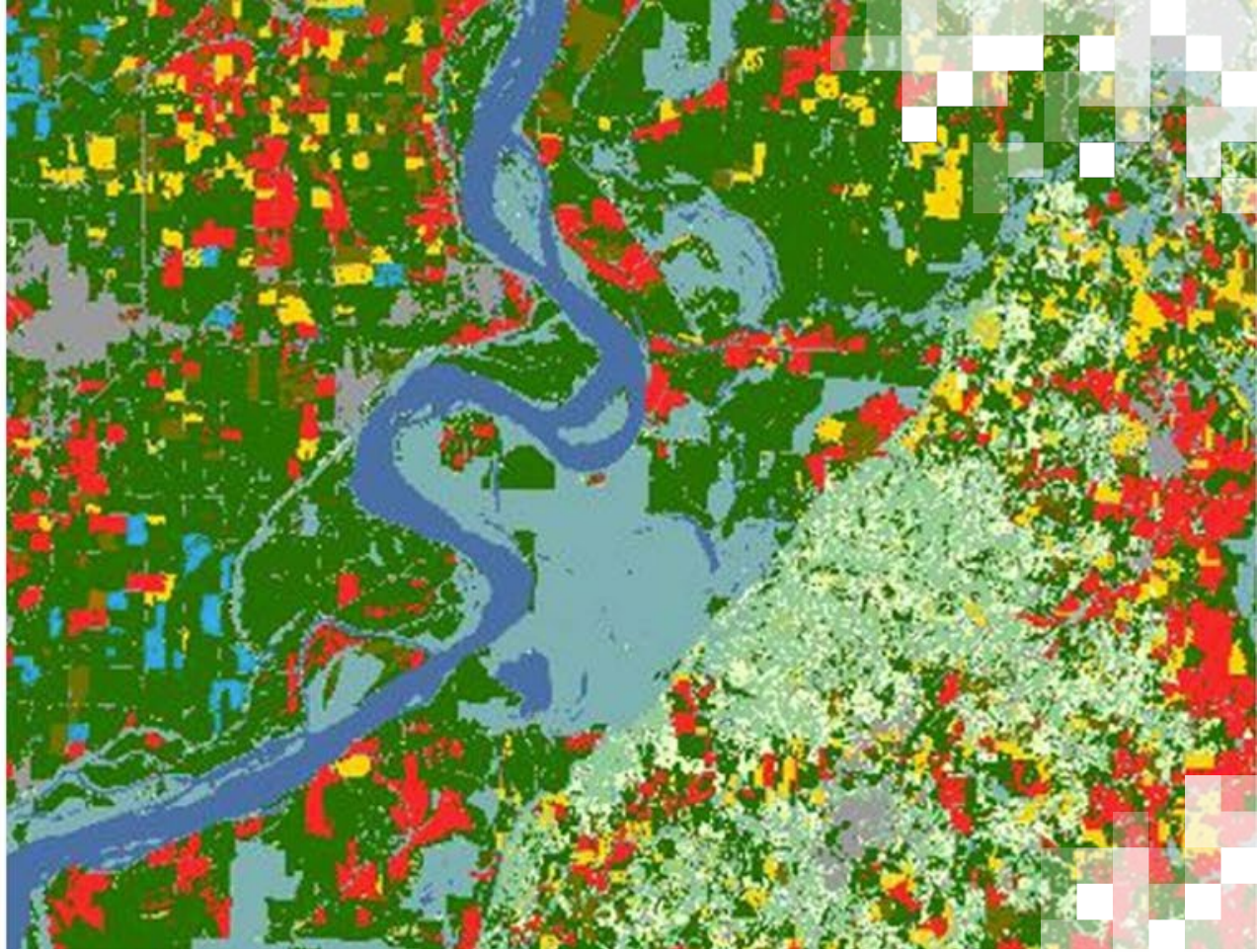
Principal Data Scientist

John Deere

Erik Sorensen

Senior Data Scientist

John Deere



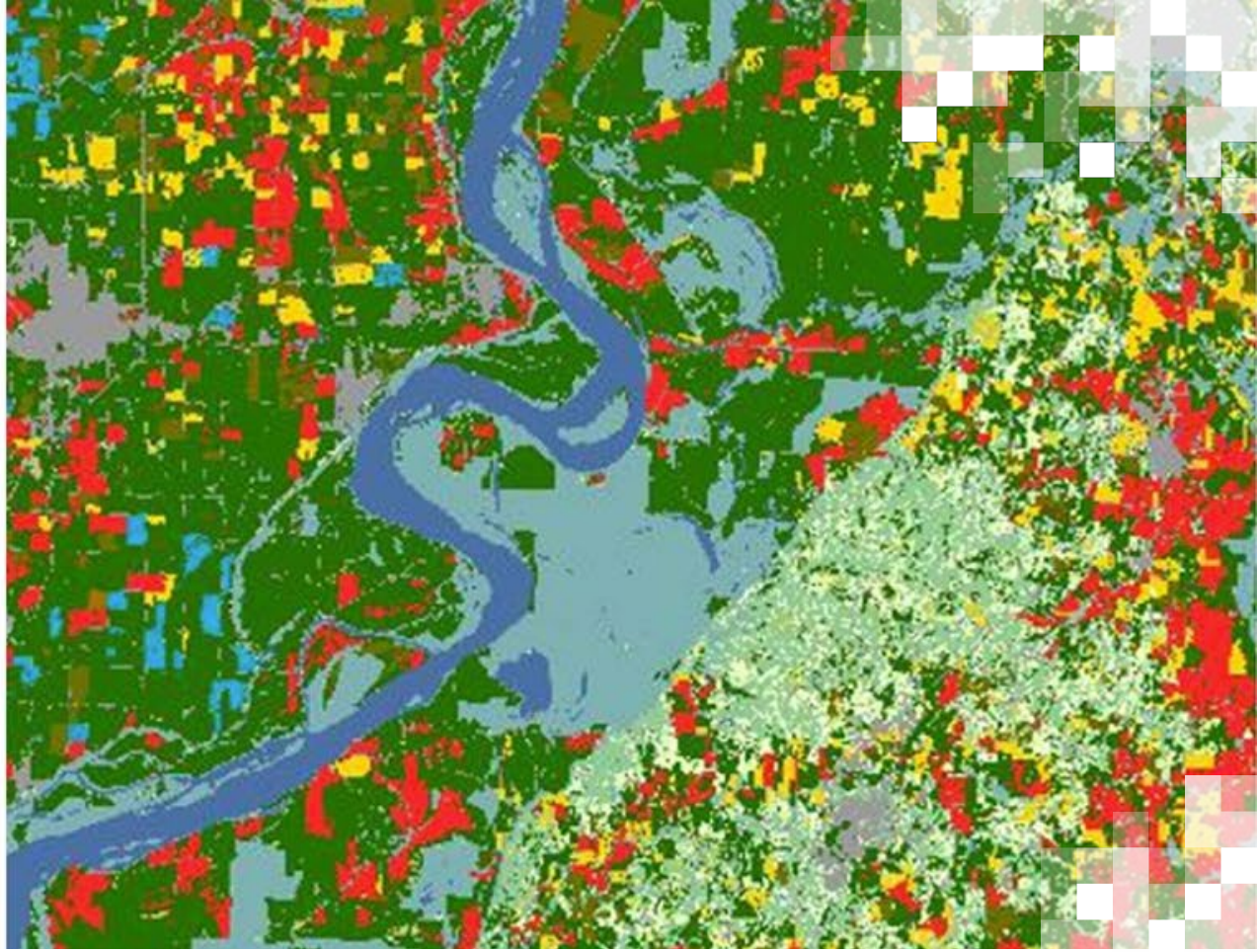
Large Scale Applications of Machine Learning using Remote Sensing for Building Agriculture Solutions

Part 3: Training & Testing ML Models for Irregularly-Spaced Time Series of Imagery

Part 3 Objectives

By the end of Part 3, participants will be able to use Python in Databricks Community Edition (or any other Python environment) to:

- Set up and train a 1-D convolutional neural network (CNN) model that learns to detect crop-type from a satellite image.
- Monitor model performance during training and how to choose appropriate hyperparameter adjustments.
- Visualize the model output in various ways to validate performance after training.

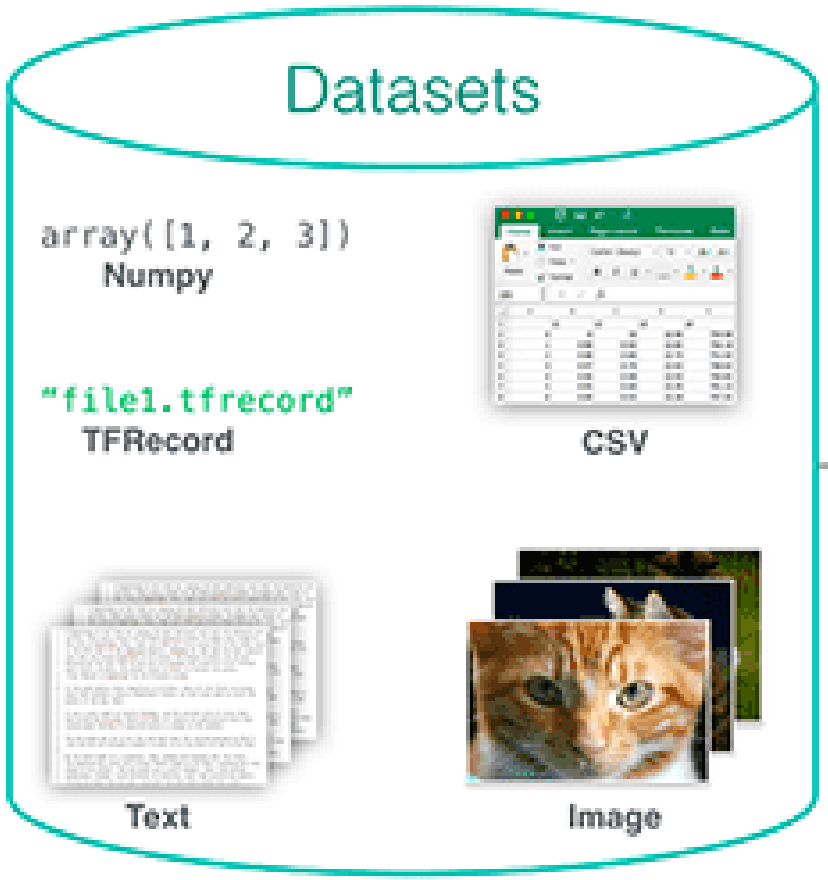


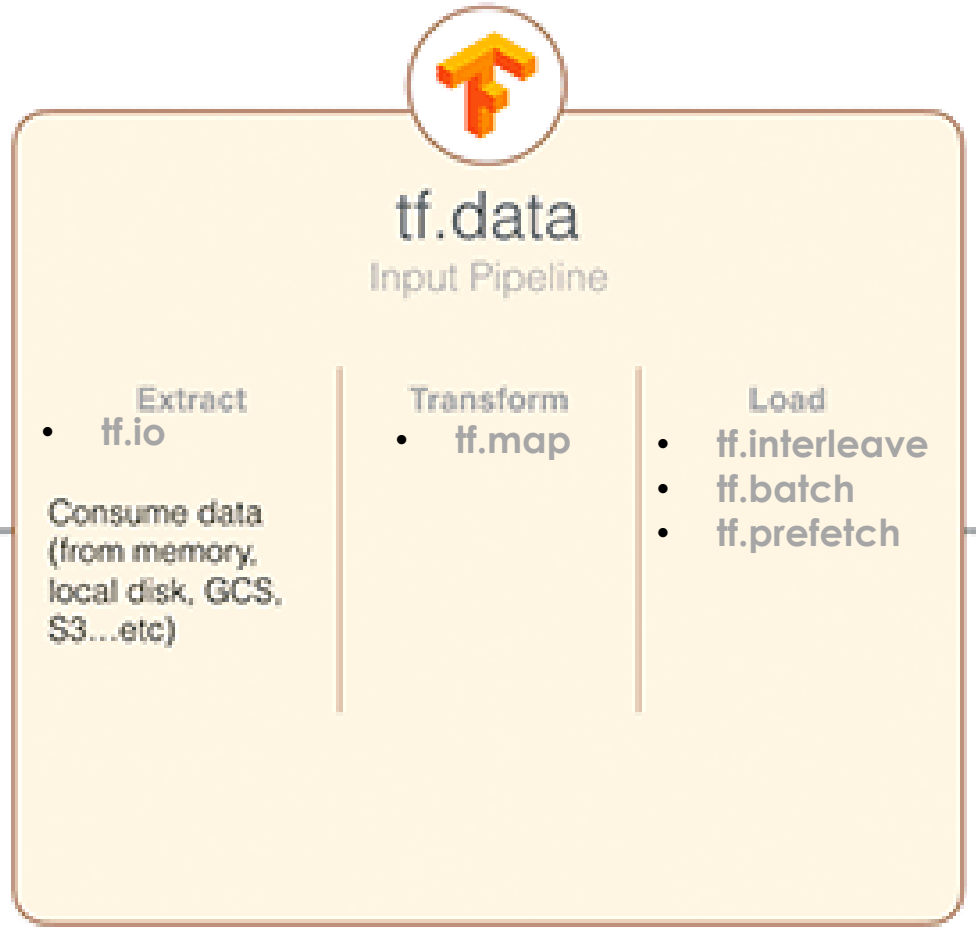
Part 3 Section 1:

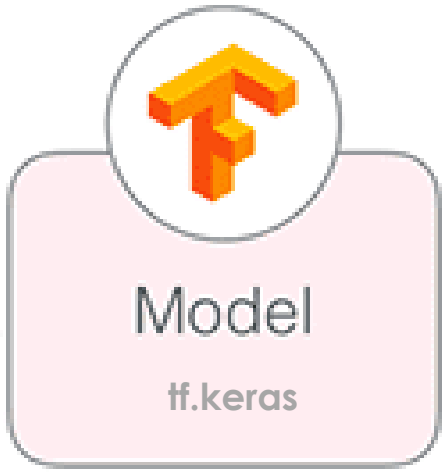
Build One-Dimensional (temporal) Convolutional Neural Network Model with Keras

Data Flow

We are here







Part 1 (built a dataset)

Part 2 (build a dataloader)

Part 3
(model training and inference)

Keras Modules

Models API

- The Sequential class
- Model training APIs
- Saving & serialization

Losses

- Probabilistic losses
- Regression losses

Layers API

- Layer activations
- Layer weight initializers
- Convolution layers
- Pooling layers
- Normalization layers
- Regularization layers
- Reshaping layers
- Activation layers

Callbacks API

- ModelCheckpoint
- BackupAndRestore
- TensorBoard
- EarlyStopping
- LearningRateScheduler
- LambdaCallback

Optimizers

- SGD
- RMSprop
- Adam

Metrics

- Accuracy metrics
- Probabilistic metrics
- Regression metrics
- Classification metrics based on True/False positives & negatives
- Image segmentation metrics

Keras Simple 1D CNN Model

- model = Sequential()
- model.add(Conv1D(filters=3, kernel_size=5, input_shape=(20, 12)))
- model.add(MaxPooling1D(pool_size=5))
- model.add(Flatten())
- model.add(Dense(10, activation='relu'))
- model.add(Dense(num_classes, activation='softmax'))

model.summary()

Caution: Keras will make some assumptions for you about your model layers. E.g., here it assumed you don't want any padding.

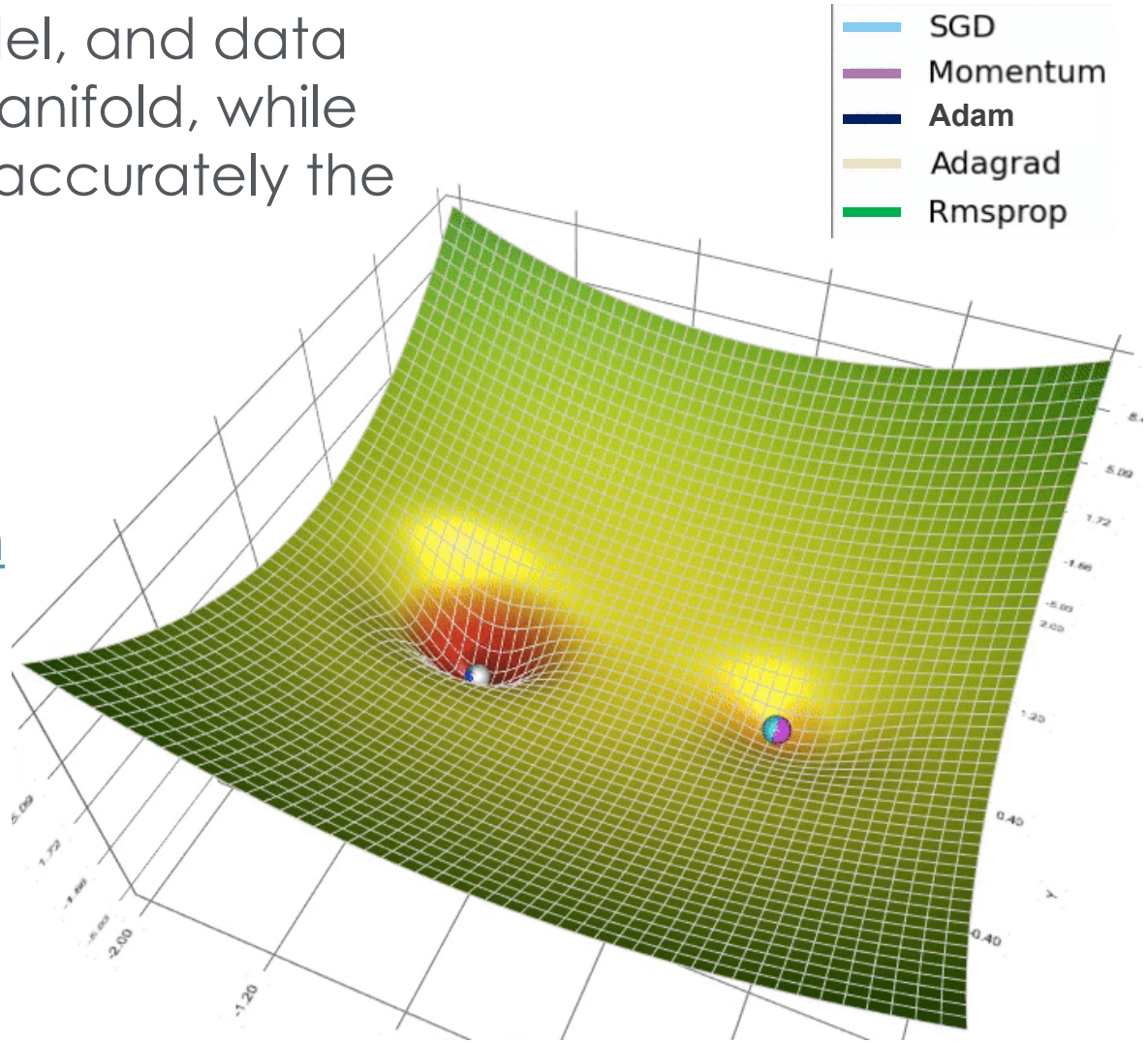
Layer (type)	Output Shape	Param #
conv1d_1 (Conv1D)	(None, 16, 3)	183
max_pooling1d_1 (MaxPooling1D)	(None, 3, 3)	0
flatten_1 (Flatten)	(None, 9)	0
dense_1 (Dense)	(None, 10)	100
dense_2 (Dense)	(None, 6)	66

Total params: 349
Trainable params: 349
Non-trainable params: 0

Losses & Optimizers

In mathematical terms, the loss function, model, and data determine the structure of the optimization manifold, while the optimizer determines how efficiently and accurately the optimal point is found (marble paths).

For this demo we use a common probabilistic multi-class classification loss referred to as categorical cross-entropy loss, and the Adam optimizer.



Credit: Medium

Training & Testing

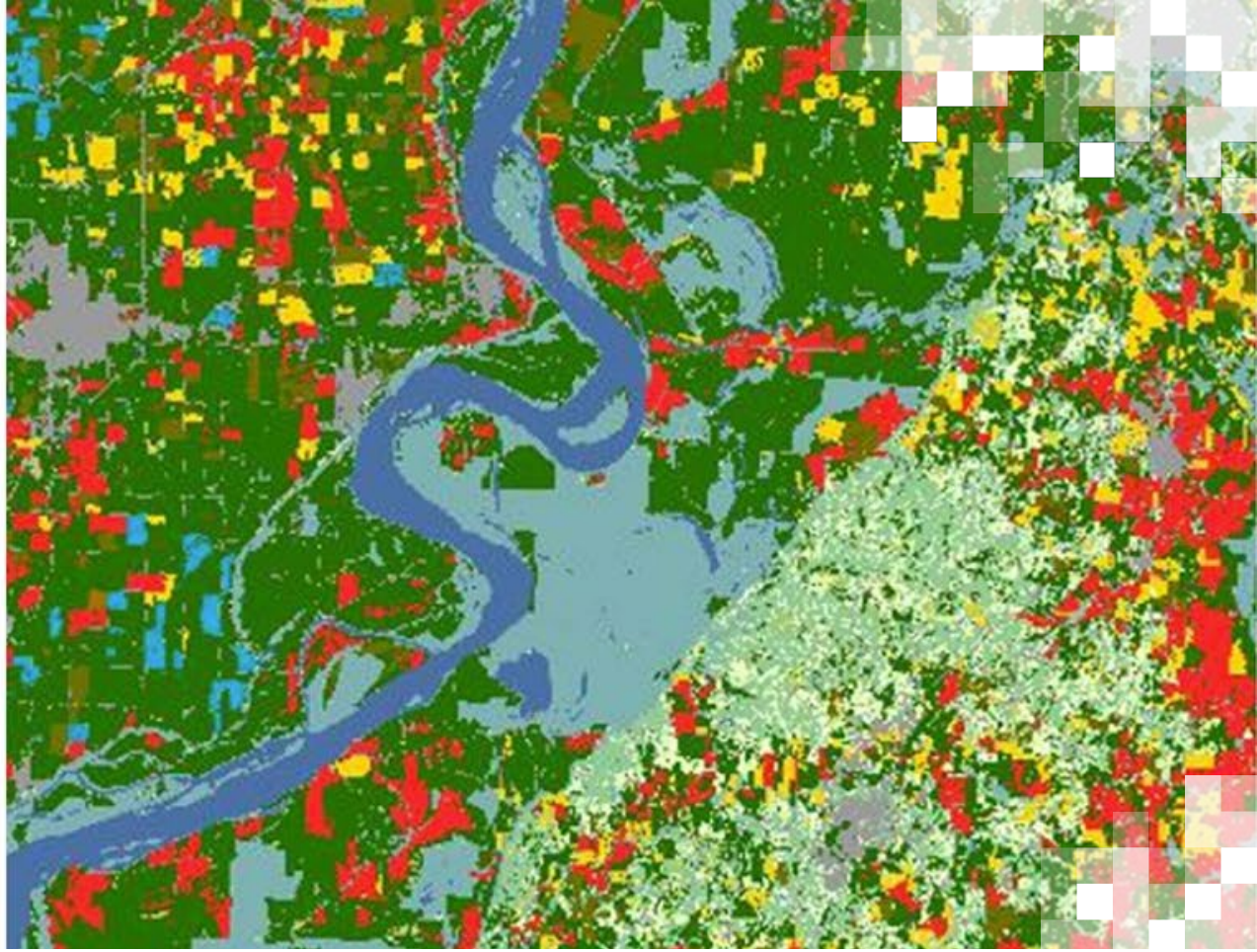
Training is quite easy with Keras since much of the functionality we need is already built-in and we just call "model.fit". We need to specify a few things though:

- The training and validation datasets (what we built in "Part-2")
- Loss function, optimizer, and metrics
- Callbacks (in this case TensorBoard and EarlyStopping)

```
#Callbacks
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=ES_PATIENCE, mode='min')
tb_callback = tf.keras.callbacks.TensorBoard('/tmp/tensorboard/', update_freq=1)

#Assign loss, optimizer, and metrics to the model
model.compile(loss=tf.keras.losses.CategoricalCrossentropy(),
              optimizer=tf.keras.optimizers.Adam(),
              metrics=[tf.keras.metrics.CategoricalAccuracy()])

#Run training/optimization routine to fit model to data
history = model.fit(train_ds,
                    epochs=MAX_EPOCHS,
                    validation_data=val_ds,
                    callbacks=[early_stopping, tb_callback],
                    verbose=1)
```



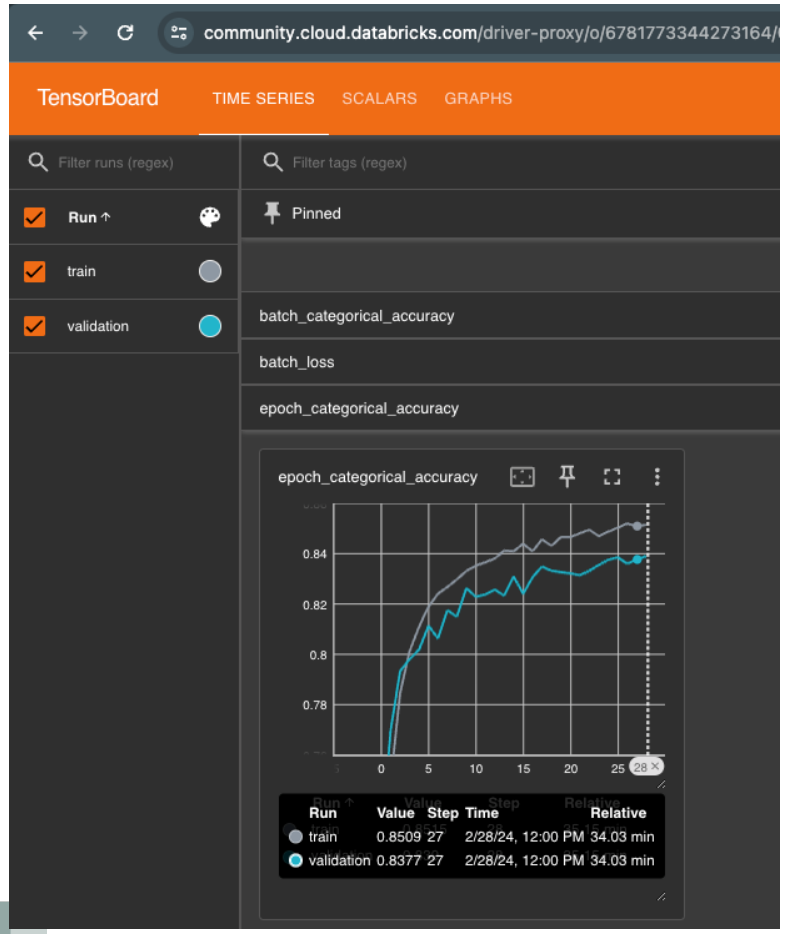
Part 3 Section 2:
TensorBoard

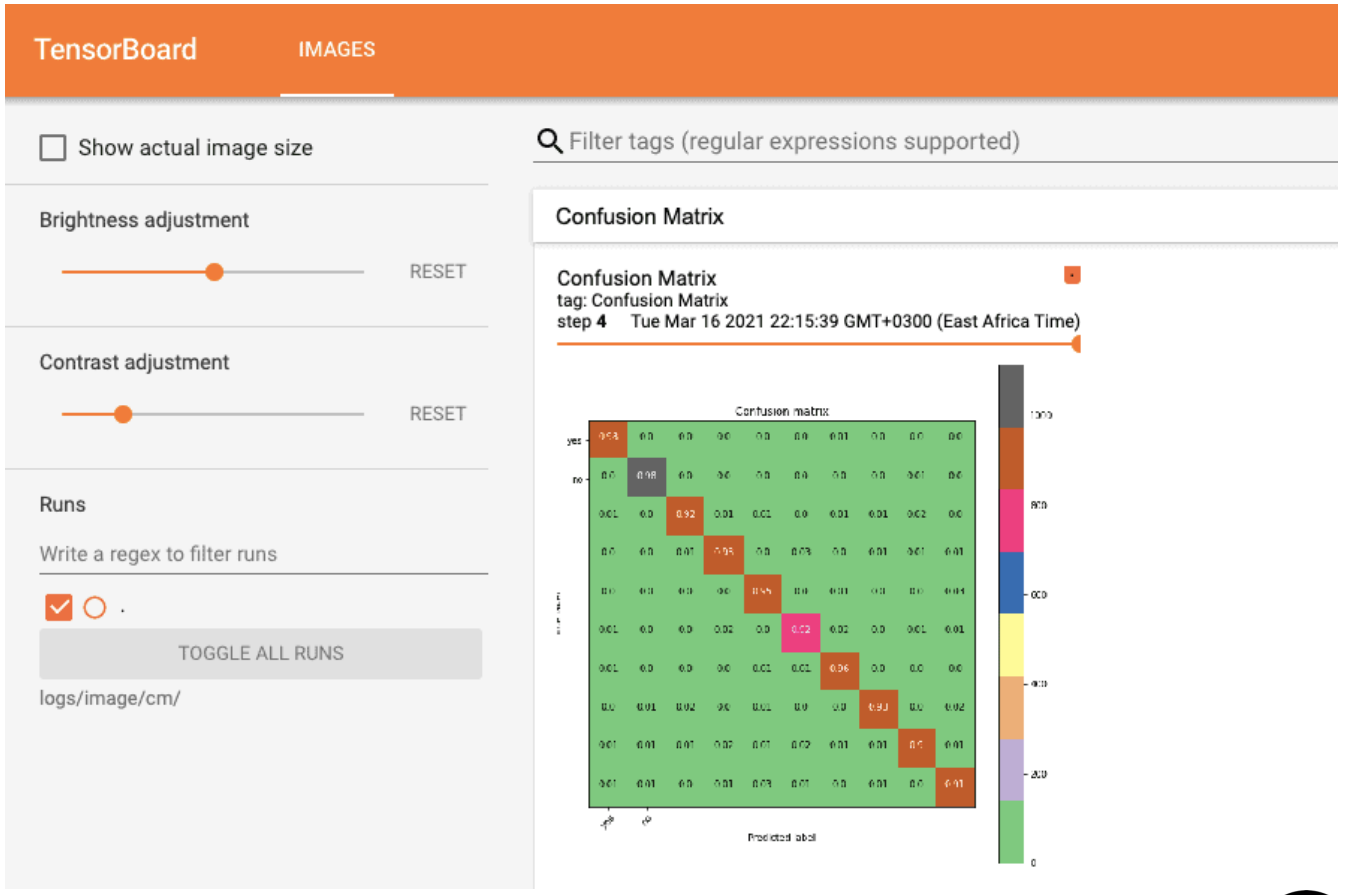
TensorBoard Overview

- TensorBoard is a tool that is part of TensorFlow and is used for providing the measurements and visualizations needed during the machine learning workflow.
- It's a unique library, and even other competing ML tools like PyTorch use TensorBoard.
- Some features include:
  - Tracking and visualizing metrics such as loss and accuracy
  - Visualizing the model graph (operations and layers)
  - Viewing histograms of weights, biases, or other tensors as they change over time
  - Projecting embeddings to a lower dimensional space
  - Displaying images, text, and audio data
  - Profiling TensorFlow programs

TensorBoard: Model Performance Visuals

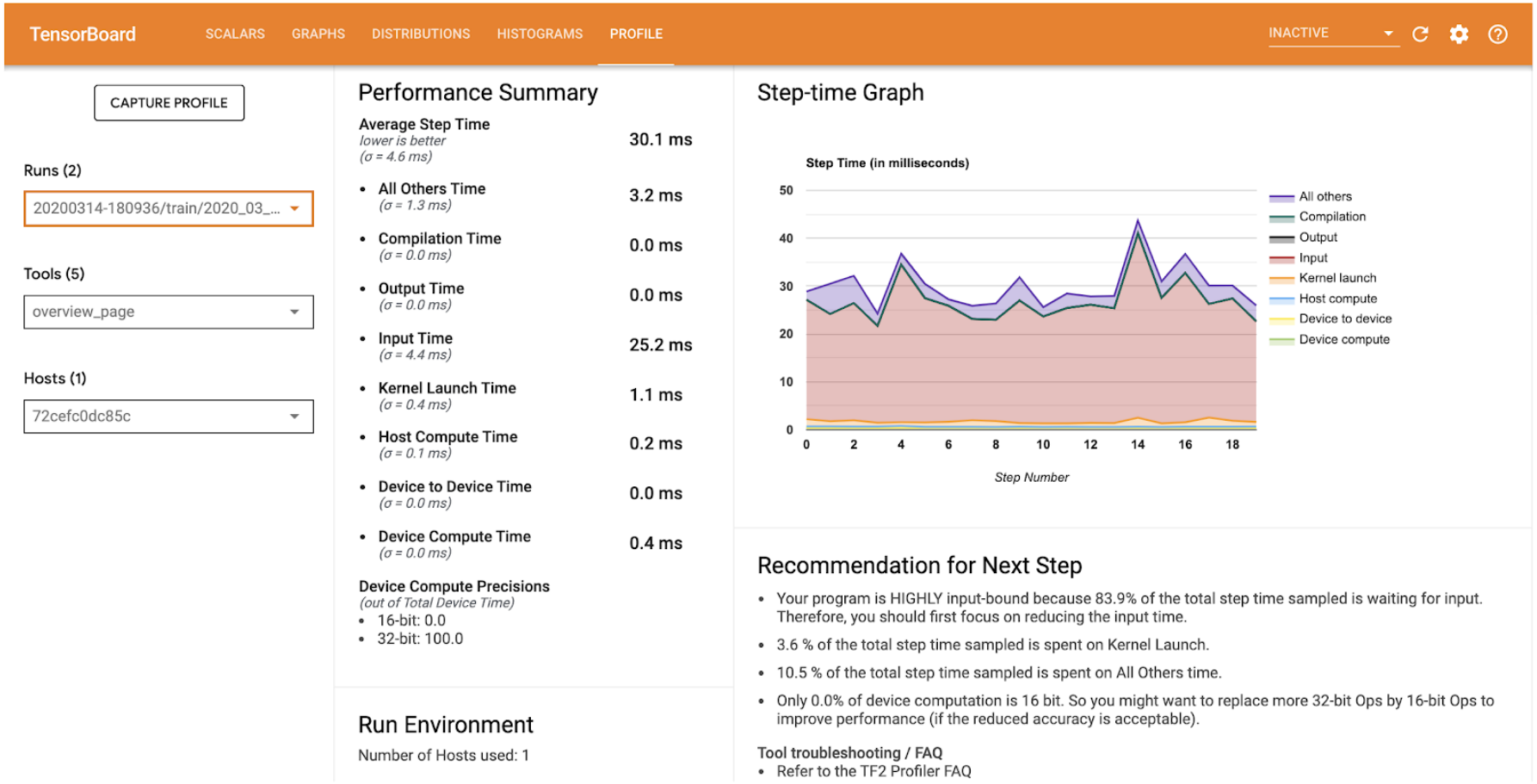
- Time series and scalars plots are the most typical visualizations used for monitoring the training process (loss & accuracy metrics show up here).
- Other custom visualizations like images can be viewed as well.



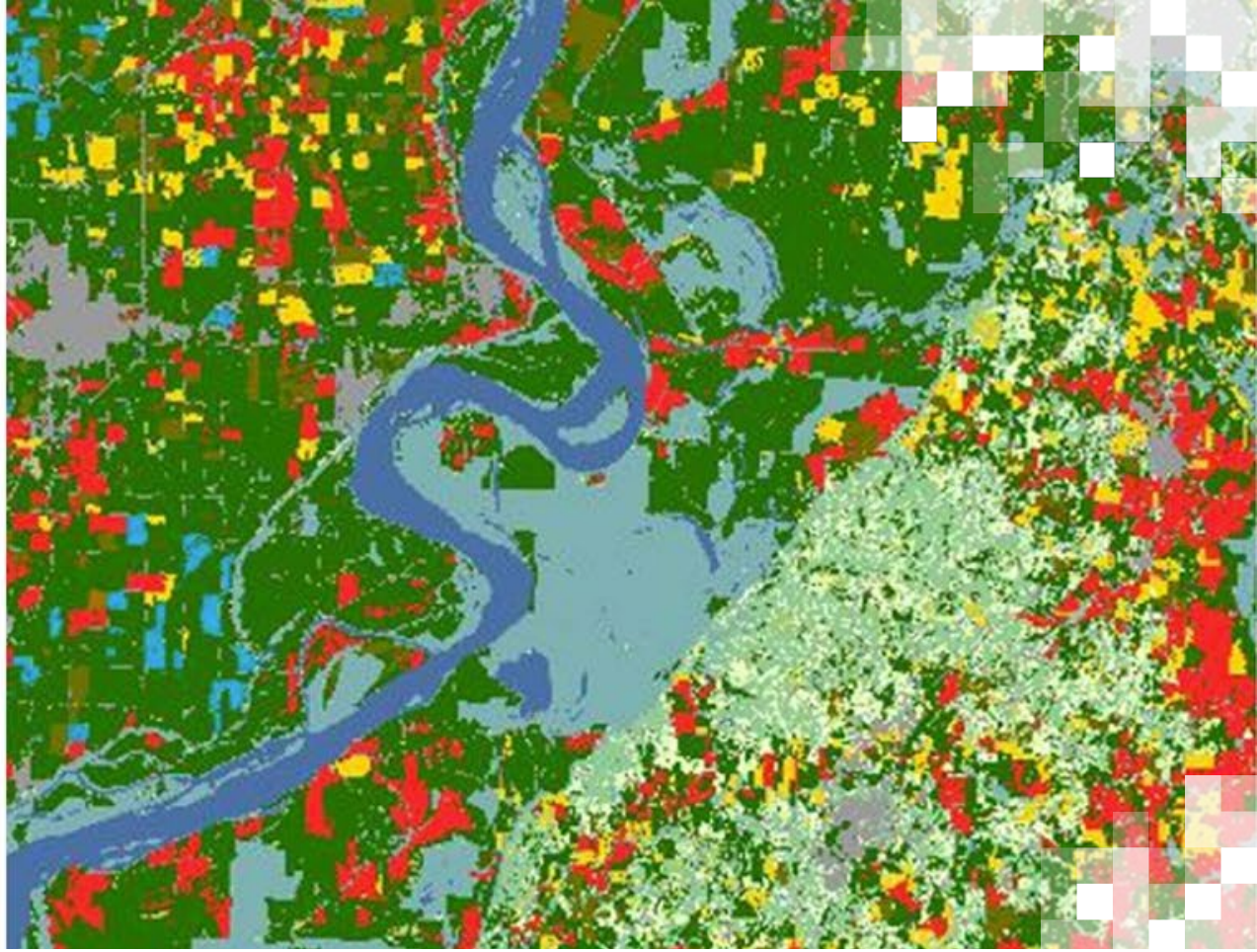


TensorBoard: TF Profiler

The TF Profiler helps you understand the hardware resource consumption (time and memory) of the various TensorFlow operations (ops) in your model and resolve performance bottlenecks to ultimately make the process execute faster.



Note this may not work within Databricks due to the nature of the database runtime.



Part 3 Section 3:

Databricks Procedural Demo (Run the code)

Databricks Community Edition Overview

Link to instructions to signup for Databricks Community Edition

- Jupyter Notebook style coding
- Databricks Community Edition allows up to 10GB persistent storage on the "FileStore."
  - Can store generic files, tables, and code.
  - Notebooks are stored in the "workspace" area.
- Can spin up small instances with 2 CPUs, 15GB RAM, 130GB local storage, Spark enabled out of the box.
- Anything stored on the local machine is lost when the instance shuts down.
- Running notebook code longer than ~60 minutes will cause the node shutdown. However, as long as you are interacting with the notebook (writing and running code manually) it usually will stay up longer.

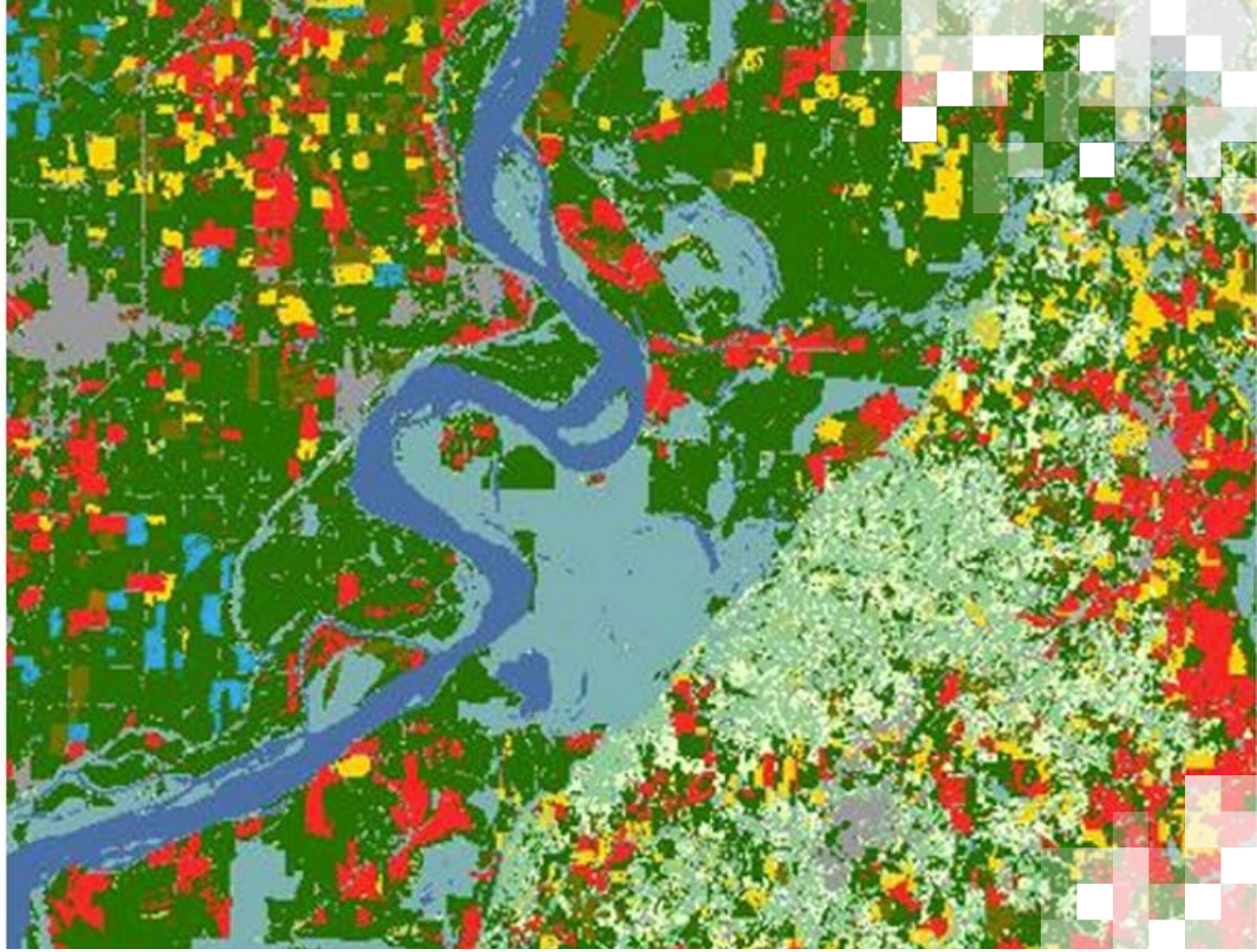
Demo Information and Notes

Available materials for this demo:

- One script for model training & evaluation for Part-3. Scripts from previous parts can also be obtained from the website (with other training materials).
- To get data for training and the model as it would exist after running the scripts in this training demonstration, download the zip files located with the other training materials.
- Note: The model and training process have not been exhaustively tuned via hyperparameter searches.

How to download the resulting files from your Databricks account FileStore:

- This is somewhat not intuitive. To download, you should navigate to the path of **YOUR** file using the below format.
  - https://community.cloud.databricks.com/files/path/to/folder/filename.extension
    - 'path/to-folder' is the directory path where your file exists on the file store.



Training Summary

Part 1 & 2 Summary

- APIs allow us to automate and scale very large data processing pipelines in preparation for analysis and model building.
- A convenient form for modeling time-series imagery data involves storing in Parquet table format
- Storing data in Parquet format and using Spark/Databricks to query or manipulate the data enables rapid investigation and transformation
- Creating a customized and highly efficient queue to load data via TensorFlow datasets using Parquet files enables efficient extraction of subsets of data.
- Using map, shuffle, batch, and prefetch one can optimize the performance of the TensorFlow dataset with parallelization.

Part 3 Summary

- **Keras** is a module within TensorFlow that allows for building quick training pipelines of Neural Network models.
- TensorBoard allows monitoring what the model pipeline is doing during training including **loss curves**, **performance metrics graphs**, and **custom images**.
- When working with satellite data models, it is important to visualize your results across both **space** and **time**, as results can change drastically depending on the time of year the predictions are made.
- **Randomness** is important in the **training** dataloader but should be **removed** from the **test** dataloader (for/when analyzing results).
- Label definition has a large impact on model results. E.g., a sparsely represented class in the training set will likely result in poor performance for that class at test time (e.g., "Cultivated").
- We can successfully train a model to predict crop type in real-time during the season using this process!

Homework and Certificates

- **Homework:**
  - One homework assignment
  - Opens on March 19
  - Access from the training webpage
  - Answers must be submitted via Google Forms
  - **Due by April 1**

- **Certificate of Completion:**
  - Attend all three live webinars (attendance is recorded automatically)
  - Complete the homework assignment by the deadline
  - You will receive a certificate via email approximately two months after completion of the course.

Contact Information

Trainers:

- John Just (John Deere)
  - JustJohnP@JohnDeere.com
- Erik Sorensen
  - SorensenErik@JohnDeere.com
- Sean McCartney
  - Sean.McCartney@nasa.gov

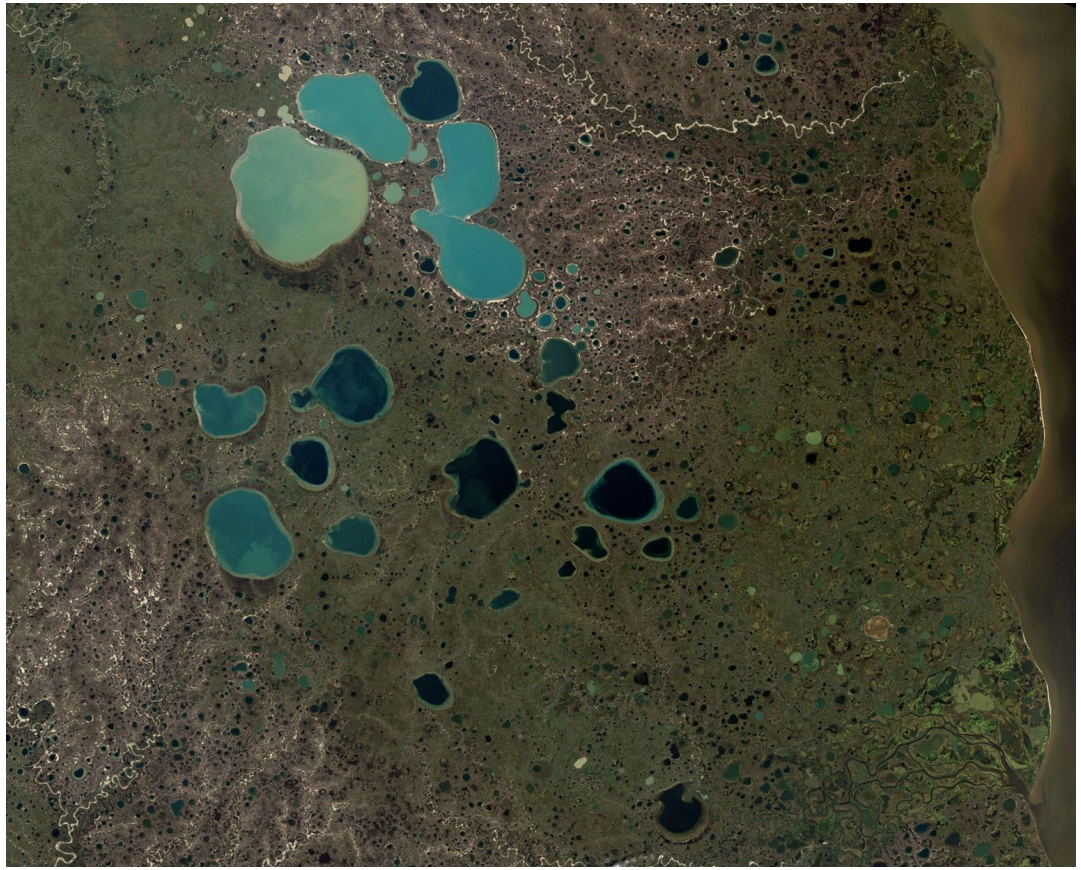
- ARSET Website
- Follow us on X (formerly Twitter)!
  - @NASAARSET
- ARSET YouTube

Visit our Sister Programs:

- DEVELOP
- SERVIR

Questions?

- Please enter your questions in the Q&A box. We will answer them in the order they were received.
- We will post the Q&A to the training website following the conclusion of the webinar.



Credit: NASA Earth Observatory


APPLIED REMOTE SENSING TRAINING PROGRAM
TRAIN - EMPOWER - ADVANCE

Thank You!