# **RESEARCH PLAN**

## **á ML3 ñ á TensorFlow Extended ñ**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Group name: Group 4**  **List of members:**   |  |  | | --- | --- | | **Names** | **IDs** | | Nguyễn Đức Trường | 20127374 | | Lê Ngọc Tường | 20127383 | | Nguyễn Tư Duy | 20127484 | | Nguyễn Tấn Phát | 20127588 | |

|  |
| --- |
| **Keywords:** Tensor, platform, machine learning, training model, end to end  **Description:** TensorFlow Extended (TFX) is a platform for building scalable and production-ready machine learning pipelines using TensorFlow. It is an end-to-end platform that helps developers to transform data, train models, and deploy them to production environments.  **List of references:** ámaterials that have been used in this researchñ   * Google. Tensorflow homepage (https://www.tensorflow.org) * Google. Tensorflow Extended tutorials (<https://www.tensorflow.org/tfx/tutorials>) * Nguyen Van Hieu. Tensorflow tutorials (<https://blog.luyencode.net/khoa-hoc-tensorflow/>) * NandaKishore Joshi. What is TensorFlow Extended (TFX). 2022 (<https://medium.com/mlearning-ai/what-is-tensorflow-extended-tfx-e5a07b209f0a>) |
|  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Presentation outline**   1. **Introduction:**   Deploying production-ready Machine Learning (ML) pipelines is not quite a straightforward operation, in contrast to traditional software engineering. Data ingestion, validation, pre-processing, model training, and post-training tasks are just a few examples of the many jobs that can be included in a ML pipeline. For the most part, data scientists or ML engineers eventually develop a lot of boilerplate code and glue code to execute these jobs. Since these glue codes are frequently fragile, pipeline problems result.  ML pipelines can get incredibly complex over time and require a lot of overhead to maintain job relationships. There are drawbacks to writing one's own boiler and glue codes. To adapt to any modifications in the input data format, many manual interventions are required. Platform changes may necessitate a complete restructuring of glue codes because they are platform dependent. The capabilities of distributed systems may not be fully utilized by specialized glue routines when working on massive datasets for computer vision or NLP projects.  TensorFlow Extended (TFX) is a Google-production-scale ML platform based on TensorFlow. TFX assists in reducing boilerplate code required for each task and in streamlining pipeline definitions. For orchestration technologies like Apache Airflow, Apache Beam, and Kubeflow Pipelines, TFX provides libraries to carry out various pipeline activities as well as glue code to connect the various components.    Figure 1: ML pipeline architecture  Over the years, Google has built TFX with assistance from a sizable developer and research community. Since its initial release in 2017, the platform has developed to include more elements and functionality. To enable the deployment of ML models at scale in production contexts and to offer a complete set of tools for organizing the end-to-end ML lifecycle have been two of the main driving forces behind the development of TFX.  Organizations from several sectors, including technology, banking, and the healthcare industry, have adopted TFX significantly in recent years. The platform is well-suited for a number of use cases, from developing recommendation systems to detecting fraud, thanks to its flexibility and scalability.   1. **A deeper insight to the selected solution**    1. Major components and main functionalities:   libraries_components  Figure 2: Components and main functionalities   * + ExampleGen: Ingests data into TFX pipelines and optionally splits the input dataset. * StatisticsGen: Computes statistics over the input data. * SchemaGen: Generates a schema based on the computed statistics. * ExampleValidator: Validates the input data against the schema. * Transform: Transforms the input data. * Trainer: Trains a model using TensorFlow. * Evaluator: Evaluates a trained model. * Pusher: Deploys a trained model to a serving system   1. Applications:   TFX has a wide range of applications in both academic and industry activities. In the academic field, TFX has been used to facilitate research in various areas such as natural language processing, computer vision, and speech recognition. For example, Google used TFX to build a system that could transcribe speech in real-time, which was used to improve accessibility for individuals who are deaf or hard of hearing. TFX has also been used by researchers to automate the process of data cleaning and preparation for machine learning models, which can save significant time and effort.  In the industry, TFX has been used by various companies to streamline their machine learning pipelines and improve the efficiency of their data processing. For example, Airbnb used TFX to build a machine learning platform that could automatically generate descriptions for their property listings, which improved the quality and consistency of their listings. TFX has also been used by Uber to improve their fraud detection system by automating the process of data preparation and feature engineering.   * 1. Popularity   There are no specific statistics on the number of users of TFX. However, according to a [survey by Statista](https://www.statista.com/statistics/793840/worldwide-developer-survey-most-used-frameworks/), TensorFlow is one of the most used frameworks among developers globally in 2022 with 12.95% users, outperforms other machine learning frameworks. That TFX is one of the important and prominent platforms of Tensorflow, so it can be said that the number of users of TFX is also very large.   * 1. Other solutions   There are several alternatives to TFX and pros, cons for each platform:   |  |  |  |  | | --- | --- | --- | --- | | **Platform** | **Description** | **Pros** | **Cons** | | TFX | Google-production-scale machine learning (ML) platform based on TensorFlow | Provides a configuration framework and shared libraries to integrate common components needed to define, launch, and monitor your machine learning system | Limited support for non-TensorFlow models | | Kubeflow Pipelines | Platform for building and deploying portable, scalable machine learning workflows based on Docker containers | Provides a set of pre-built components for building machine learning pipelines, as well as a framework for building custom components | Steep learning curve | | Apache Airflow | Open-source platform to programmatically author, schedule, and monitor workflows | Provides a way to create, schedule, and monitor complex workflows, as well as a way to define dependencies between tasks | Limited support for machine learning-specific tasks | | MLflow | Open-source platform for the complete machine learning lifecycle | Provides a way to track experiments, package code into reproducible runs, and share and deploy models | Limited support for model serving | | Pachyderm | Open-source platform for building scalable, reproducible machine learning workflows | Provides a way to version data and code, as well as a way to build, test, and deploy machine learning pipelines | Limited support for non-Docker workflows |  1. **Demonstration**   Anything that could be used to highlight the solution’s main functionalities, rather than being just an installation guide.   * 1. ExampleGen: * Custom input/output split     Notice how the hash\_buckets were set in this example  For an input source which has already been split, set the input\_config for ExampleGen component:     * 1. StatisticsGen: * Using StatisticsGen with a schema: For the first run of a pipeline, the output of StatisticsGen will be used to infer a schema. However, on subsequent runs you may have a manually curated schema that contains additional information about your data set. By providing this schema to StatisticsGen, TFDV can provide more useful statistics based on declared properties of your data set. * Invoking StatisticsGen with a curated schema that has been imported by an ImporterNode:      * 1. SchemaGen: * Using SchemaGen for the initial schema generation:      * Using SchemaGen for the reviewed schema import:      * 1. ExampleValidator * Computing descriptive data statistics: TensorFlow Data Validation (TFDV) can compute descriptive statistics that provide a quick overview of the data in terms of the features that are present and the shapes of their value distributions. Tools such as Facets Overview can provide a succinct visualization of these statistics for easy browsing.          * Inferring a schema over the data   To mark that the feature should be populated in at least 50% of the examples:       * Checking the data for errors   To check for errors in the aggregate, TFDV matches the statistics of the dataset against the schema and marks any discrepancies. For example:     * Checking data skew and drift   TFDV performs this check by comparing the statistics of different datasets based on the drift/skew comparators specified in the schema. For example, to check if there is any skew between 'payment\_type' feature within training and serving dataset:     * 1. Transform * Transform makes extensive use of TensorFlow Transform for performing feature engineering on your dataset. TensorFlow Transform is a great tool for transforming feature data before it goes to your model and as a part of the training process. * You can transform your data however you like prior to running TFX. But if you do it within TensorFlow Transform, transforms become part of the TensorFlow graph. This approach helps avoid training/serving skew. * Transformations inside your modeling code use FeatureColumns. Using FeatureColumns, you can define bucketizations, integerizations that use predefined vocabularies, or any other transformations that can be defined without looking at the data. * By contrast, TensorFlow Transform is designed for transformations that require a full pass over the data to compute values that are not known in advance. For example, vocabulary generation requires a full pass over the data.   1. Trainer * Trainer takes:   tf.Examples used for training and eval.  A user provided module file that defines the trainer logic.  Protobuf definition of train args and eval args.  Other optional parameters…   * Trainer emits: At least one model for inference/serving (typically in SavedModelFormat) and optionally another model for eval (typically an EvalSavedModel). * Typical pipeline DSL code for the generic Trainer would look like this:      * 1. Evaluator * The Evaluator TFX pipeline component performs deep analysis on the training results for your models, to help you understand how your model performs on subsets of your data. The Evaluator also helps you validate your exported models, ensuring that they are "good enough" to be pushed to production. * When validation is enabled, the Evaluator compares new models against a baseline (such as the currently serving model) to determine if they're "good enough" relative to the baseline. It does so by evaluating both models on an eval dataset and computing their performance on metrics (e.g. AUC, loss). If the new model's metrics meet developer-specified criteria relative to the baseline model (e.g. AUC is not lower), the model is "blessed" (marked as good), indicating to the Pusher that it is ok to push the model to production. * Consumes:   An eval split from ExampleGen.  A trained model from Trainer.  A previously blessed model (if validation to be performed).   * Emits:   Analysis results to ML Metadata.  Validation results to ML Metadata (if validation to be performed).   * 1. Pusher * The Pusher component is used to push a validated model to a deployment target during model training or re-training. Before the deployment, Pusher relies on one or more blessings from other validation components to decide whether to push the model or not. * A Pusher component consumes a trained model in SavedModel format, and produces the same SavedModel, along with versioning metadata. * A Pusher pipeline component is typically very easy to deploy and requires little customization, since all of the work is done by the Pusher TFX component. Typical code looks like this:      1. **Discussions and Conclusion:**  * In conclusion, TFX is a powerful framework that offers a wide range of functionalities for building and deploying machine learning pipelines at scale. It provides a unified platform for managing and automating the entire ML workflow, from data preparation to model training and deployment. TFX allows data scientists and engineers to focus on the high-level aspects of machine learning, such as feature engineering and model selection, while abstracting away the low-level details of data preprocessing and model serving. * TFX has gained significant popularity in both academia and industry, and is being widely used by many organizations around the world (as Google, Intel,…). While there are other solutions available that offer similar functionalities, TFX stands out in its ability to integrate seamlessly with other TensorFlow-based tools and technologies, making it a preferred choice for many users. The rich set of features, ease of use, and scalability of TFX make it a compelling option for anyone looking   to build and deploy ML pipelines efficiently and effectively. |
|  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Weekly schedule:**   |  |  |  |  | | --- | --- | --- | --- | |  | Week 4-5  (27/2 - 11/3) | Week 6-8  (13/3 -1/4) | Week 9-11  (3/4 - 22/4) | | 20127588 | - Overview research - Install Tensorflow on laptop - Build basic model ML - Plan for the week | - Plan for the week - Assign work to each member - Report: Transform com- ponent - Report: Trainer compo- nent | - Plan for the week - Report: Model Analysis Libraries - Slide for the first sub- mission - Code: TFX Component - Code: Model Analysis - Synthesize code, slides, reports to edit and sub- mit | | 20127383 | - Overview research. - Install TF, TFX on Co- lab. | - TFX pipeline Basics. - Research, report: Evalu- ator, Pusher components.. | - Research, report: Serv- ing library and deploy. - Code: demo for serving, TFX Pipeline. - Slides design: Compo- nents. | | 20127484 | - Overview research. - Import Tensorflow and TFX on Google Colab.. | - Watch some basic tuto- rials. - Research and write re- port about: ExampleGen component and Statistic- sGen component. | - Research and write report about: TFX pipeline and TensorFlow Data Validation (TFDV) library. - Code a simple demo us- ing some main functions of TFDV. - Make presentation slides (PowerPoint): 2. TFX pipeline | | 20127374 | - Overview research - Try TF, TFX on Google Colab | - Watch some tutorials - Research and write re- port about: SchemaGen and ExampleValidator | Research and write report about: Trans- form(TFT) library - Code a simple demo us- ing TFT library - Slides design: Conclu- sion. | |

|  |
| --- |
| **Group self-evaluation**   * **Advantages:** * Research TFX help we know about how to implement machine learning pipeline, enhancing our Machine Learning skills. * Improving your productivity: Instead of coding yourself a pipeline (it’s a waste of time and effort), if we know about TFX, we can use simplified pipeline definitions and minimize the codes to write for every task. * Keeping up with the latest developments. * Building more reliable and scalable ML systems. * Advancing our career. * **Disadvantages:** * Vietnamese documents are few. English document has some unknown terms. * TFX is a powerful and comprehensive platform for building machine learning pipelines, but it can also be quite complex. * One of the key challenges in building machine learning models is preparing and cleaning the data used for training. |
|  |
|  |