Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 171 (2020) 188–196

Procedia Computer Science

www.elsevier.com/locate/procedia

Third International Conference on Computing and Network Communications (CoCoNet'19)

Project repositories for machine learning with TensorFlow

PS Janardhanan *

SunTec Business Solutions Pvt Ltd, TechnoPark, Trivandrum 695581, India

Abstract

In machine learning, models capture intelligence from data using algorithms implemented on frameworks like TensorFlow. Models learn during the training phase; an iterative process in which parameters are tuned to improve the prediction accuracy. Software repositories are used to save the artifacts of model development so that they can be modified in subsequent releases and shared between development teams. Issues in saving the state of machine learning projects is different from standard software practices. Machine learning models are equivalent to binary executable programs and ideally one should be able to recreate the model from the information preserved in project repositories. Recreation of the model becomes necessary when there is a change in the development team as part of the product transition. Retraining of models also becomes necessary when the inaccuracy in the predictions made on new data increases beyond the acceptable limit. When traditional source code management systems are used for maintaining repositories of machine learning projects, we face challenges in keeping complete information required for model development starting from saved states. This paper presents the results of the studies conducted to identify the challenges in maintaining TensorFlow machine learning projects in repositories. Some of the existing tools are compared and recommendations are made to improve the ease of recreation of machine learning models by saving complete information in project repositories maintained in normal source code control systems.

Keywords: Knowledge Store; Machine Learning; Project Repository; TensorFlow; Model Versioning;

* Tel.: +91-984-526-1540.
E-mail address: janardhananps@suntecgroup.com

10.1016/j.procs.2020.04.020

1. Introduction

Machine learning (ML) development practices differ from traditional software engineering since machines automatically learn problem solving skills from data [1]. In traditional software development, humans develop code based on algorithms. In machine learning, models learn problem solving skills and there is no human readable code in them. This introduces certain challenges when conventional source code control systems are used to maintain repositories of machine learning projects. The challenges are in experiment management, reproducibility, and production deployment [6]. TensorFlow is an open source machine learning platform for developing models from training data in a short time making use of heterogeneous computing resources like CPUs, GPUs and TPUs. TensorFlow provides an extensive suite of functions and classes that allow users to build complex models from scratch. The latest version is TensorFlow 2.0 with substantial improvements from the previous version. TensorFlow has received wide acceptance among the machine learning community in a short span of time because of its ease of use, python interface and capability to deploy models on web browsers and mobile devices. Although TensorFlow is primarily used for machine learning, it can also be used for developing non-ML tasks that require numerical computation using data flow graphs. When the models are packaged for distribution, the contents vary significantly from executable binaries. This paper addresses the issues faced when conventional source code control systems are used as repositories for storage and versioning of ML projects.

The remaining sections of this paper are organized as follows: Section 2 defines the problem solved in this work. Section 3 gives an overview of machine learning models with specific information on TensorFlow models. Section 4 describes the model development in detail with the parameters learned by the model and details of the functions available for building the model. This section also describes the hyper-parameters used in the training phase to improve the accuracy of the models. Section 5 justifies the need for version control in ML projects. Section 6 describes existing tools and services that are exclusively used for versioning of ML projects. Section 7 presents the proposed solution and the paper is concluded in section 8. The last section is on references used in this study.

2. Problem statement

Machine learning is a process that builds predictive models from large number of examples. The development methodologies used in ML is different from traditional software development. The complexity of traditional computer programming is in the code and in machine learning, the complexity is in model development. The ML development methods are common for different problems and these methods are getting standardized with the evolution of frameworks such as TensorFlow. These frameworks provide a platform for model building using algorithms coded in programming languages like python. During the training phase, the models learn from data and the accuracy of the model depends on the quality and volume of training data.

The training phase of model development is iterative in nature in which the configuration parameters are tuned continuously to improve the prediction accuracy [1]. It is required to archive these parameters in each iteration along with the test results indicating the accuracy. This information will be used when the models are retrained. To achieve exact reproduction of models, it is required to store the training data and test data in repositories with proper versioning. Since the training data will be of massive volume, we may store only a snapshot of it with pointers to the location of the total training data. All these requirements impose challenges when traditional source code control systems are used for storing the artifacts required for model development.

Often ML capabilities are added to existing software systems for enhancing them with automated problem-solving skills. It is preferred to use the same source code repository for all components related to one project. This necessitates the need for storing artifacts of traditional software systems and ML systems together in one repository. At present, no standards are available for maintaining the artifacts of ML projects to be maintained in conventional source code control systems. There exist source code control systems exclusively designed for ML development projects. For software developers migrating to ML world, it will be a challenge to use dedicated ML tools together with traditional source code control systems.

3. Machine learning models in TensorFlow

ML is the study and construction of systems that can learn from data and apply the learned knowledge for automated problem solving. The AI systems use the model to make useful predictions from never seen data drawn from the same distribution as the one used to train the model. A model is a representation of what a machine learning system has learned from the training data. Within TensorFlow, model is a graph that expresses the structure of how a prediction will be computed. The weights and biases of TensorFlow graph are determined during the training phase. After the training is completed, all the variables and network graph are to be saved to a file for future use. TensorFlow uses its own format for saving and recovering the artifacts of models. The model artifacts are saved in a language-neutral, recoverable serialization format, which enables higher-level systems and tools to produce, consume, and transform TensorFlow models.

TensorFlow models are executables of the machine learning paradigm of computing. What machine learning practitioners term models are expressed as programs that TensorFlow executes. The model's parameters are often stored separately in checkpoints. TensorFlow's run-time system interprets and executes the computational graphs using the parameters learned during training phase. Note that the behavior of the computation graph may change depending on these parameters. When executing the computation graph, TensorFlow may read and write files, send and receive data over the network, and even spawn additional processes. All these tasks are performed with the permissions of the TensorFlow process. Allowing for this flexibility makes TensorFlow a powerful platform for training and deployment of models on wide variety of platforms. The computation graph may also accept inputs. These inputs are the data you supply to TensorFlow to run inference on real world data.

The *tf.train.Saver* class provides methods to save and restore models. Model saving is done by creating an instance of *tf.train.Saver()* class. If we don't specify any argument in *tf.train.Saver()*, it saves all the variables [3]. A single saved model can represent multiple graphs. In this case, all the graphs in the saved model share a single set of checkpoints and assets. A saved TensorFlow model has two main files; Meta graph and Checkpoint file. The meta graph is a protocol buffer which saves the complete TensorFow graph consisting of variables, operations, collections etc. This file has an extension of *.meta*. A MetaGraph is a dataflow graph, plus its associated variables, assets, and signatures. A signature is the set of inputs to and outputs from a graph. The checkpoint file is a binary file which contains all the values of the weights, biases, gradients and all the other variables representing the intelligence captured in the model. This file has an extension of .ckpt.

4. Model development

During the training phase, the time taken for capturing knowledge in the form of models depends on the choice of TensorFlow API functions and the parameters selected in each experiment. This section is on high level API functions that can be used for model development and various user defined parameters that can be used to refine the accuracy of the models. Models learn and store knowledge in the form of model parameters. A model parameter is internal to the model and its value is estimated or learned from the training data. They define the skill of the model in solving a problem and are used for making predictions. They are saved as part of the learned model. Model parameters are estimated using optimization algorithms exposed by TensorFlow in the form of API functions, which is a type of efficient search through possible parameter values. Some examples of model parameters include weights and biases in artificial neural network, support vectors in a support vector machines, and coefficients in linear or polynomial regressions. Users of the model do not have direct access to these parameters. They are used by the models to make inferences on real world data when used in production.

4.1. API Functions

Python is the preferred programming language for developing models in TensorFlow and these programs make a series of calls to high level TensorFlow APIs. These APIs are for selecting the Loss functions, Optimizers and Activation functions. TensorFlow provides APIs for all these functions and they are shown in Table.1.

Table 1. Loss, Optimization, Activation functions.

Type	Function
Regression Loss	L1 (Mean absolute error)
Regression Loss	L2 (Mean square error)
Regression Loss	Mean bias error
Regression Loss	Huber
Classification Loss	Cross Entropy / Negative Log Likelihood
Classification Loss	Hinge / Multi class SVM
Classification Loss	Squared Hinge
Neural Network Loss	MLE – Maximum likelihood estimation
Optimizer	Stochastic Gradient Descent
Optimizer	Mini-Batch Gradient Descent
Optimizer	Adagrad (Adaptive Gradient Descent)
Optimizer	RMSProp
Optimizer	Adam (Adaptive Moment Estimation)
Neural Network Activation	Sigmoid (Logistic)
Neural Network Activation	Softmax
Neural Network Activation	Softplus
Neural Network Activation	Tanh (Hyperbolic tangent)
Neural Network Activation	Elu
Neural Network Activation	Relu (Rectified Linear Units)
Neural Network Activation	Leaky Relu

Appropriate selection of these functions is required to train a model with highly accurate prediction capabilities. Less experienced ML developers spend lot of time in function selection by trial and error. Highly experienced ML experts will have lot of intuition for selecting these functions in a short time

4.2. Hyper-parameters

Machine learning development is an iterative process in which the accuracy of the model is continuously improved by repeating the training and evaluation phases. In each of these iterations, certain parameters are tweaked continuously by the developers. Any parameter manually selected based on learning from previous experiments qualify to be called a model hyper-parameter [4]. These parameters represent intuitive decisions whose value cannot be estimated from data or from ML theory. The hyper-parameters are knobs that you tweak during each iteration of training a model to improve the accuracy in the predictions made by the model. The hyper-parameters are variables that govern the training process itself. They are often specified by practitioners experienced in machine learning development. They are often tuned independently for a given predictive modeling problem.

Building an ML model is a long process that requires knowledge, experience and intuition. In ML, hyper-parameter optimization or tuning is the problem of choosing a set of optimal hyper-parameters for a learning algorithm. We may not know the best combination of values for hyper-parameters in advance for a given problem. We may use rules of thumb, copy values used on other problems, or search for the best value by trial and error. When a machine learning algorithm is tuned for specific problems by changing the TensorFlow higher level APIs, we need to tune the hyper-parameters also to discover optimum combination of parameters that results in a model with higher accuracy in prediction. Hyper-parameter tuning is often referred to as searching the parameter space for

optimum values. With Deep Learning models, the search space is usually very large, and a single model might take days to train. Table.2 shows the list of hyper-parameters used in ML development work using TensorFlow framework. Typical range of values that may be used in each iteration is also shown in the last 3 columns of Table.2.

Table 2. Hyper parameters

Parameter	Description	Typical Values		
		Start	End	Step Size
Epochs	A full training pass over the entire dataset such that each example has been seen once. An epoch represents N/batch size training iterations, where N is the total number of examples in training data.	100	1000	50
Learning rate	A scalar used to train a model via gradient descent. During each iteration, the gradient descent algorithm multiplies learning rate by the gradient. Resulting product is called the gradient step.	0.001	3.0	0.01
Momentum in Stochastic Gradient Descent	The coefficient of friction controlling the rate at which the descent happens, when it goes towards the bottom.	0.1	0.9	0.1
Regularization method	Regularization is used to prevent overfitting by the model. Different kinds of regularization include:	L1	L2	
	• L1 regularization - Lasso			
	• L2 regularization – Ridge			
	• Dropout regularization (Probability)	0.1	0.99	0.1
Regularization Rate - λ	The penalty on a model's complexity. The scalar value λ specifies the importance of the regularization function relative to the loss function. Raising the value of λ reduces over-fitting at the cost of model accuracy.	0	1000	10
Early Stopping - Patience	Regularization by early stopping callback function tests a training condition for every epoch and if a set number of epochs elapses without showing any improvement, then it automatically stops the training. The patience parameter is the number of epochs to check for improvement.	10	100	10
K in k-means clustering	Number of clusters to be discovered	5	200	10
Percentage of Validation data	Percentage of training data set being used as the validation data set.	5	25	5
C and Sigma	For Support Vector Machines			
Number of hidden layers	For Neural Networks	1	100	10
Number of units per layer	For Neural Networks	10	100	10
max_depth	Maximum depth of a tree in Random Forest method	10	50	2
n_estimators	Number of trees in the Random Forest. More number of trees gives better performance	10	1000	10
Hyper parameter optimization	Grid search, Random search or Beyesian optimization	Grid		

Model optimization using hyper-parameters is a search problem to identify the ideal combination of these parameters. The commonly used methods for optimization using hyper-parameters are; Grid search, Random search and Beyesian optimization. In Grid search, a list of all possible values for each hyper-parameter in a specified range is constructed and all possible combinations of these values are tried sequentially. In grid search, the number of experiments to be carried out increases drastically with the number of hyper-parameters. Rather than training on all possible configurations, in Random search method the network is trained only on a subset of the configurations. Choice of the configurations to be trained is randomly picked up and only the best configuration is trained in each iteration. In Beyesian optimization, we are using ML techniques to figure out the hyper-parameters. It predicts regions of the hyper-parameter space that might give better results. Gaussian process is the technique used and it finds out the optimal hyper parameters from the results of the previously conducted experiments with various types

of parameter configurations. The list of hyper-parameters shown in Table.2 is not complete. There can be additional parameters depending on the optimization function used in model training.

5. Version control

Versioning is the creation and management of multiple releases of a software product or solution, all of which have the same general function but are improved, upgraded or customized. Version control is the practice of ensuring collaborative data sharing and modification among users of systems that employ different versions of a product. Version control is a major component of software configuration management. It is also known as revision control or source control and it manages changes to documents, source codes, and other collections of information. Baselines created on versions helps in managing the deployment, current product engineering and update releases of software solutions. It allows software companies to upgrade and roll back releases depending on performance issues of software systems with change in usage scenarios.

Machine learning is an iterative trial and error process and information on parameters used in each iteration is to be preserved to recreate the model in a short time by elimination of unwanted trials [1]. Versioning lets you keep track of all the models developed in each iteration and what parameters you used to get there. The prediction accuracy of the model depends on the optimum combination of how the data is divided into training and validation sets, parameters used in training phase, algorithm choice, API functions used, and finally creativity based on domain expertise.

6. Existing tools and services

Many companies are starting to develop ML project platforms and tools for automating the development process of machine learning models. These tools make it easy to track, reproduce, manage and deploy models in production [6]. They cover each phase of ML development life cycle. The commonly used tools are MLflow, Polyaxon, Comet.ml and modified Git.

6.1. MLflow

MLflow is an open source platform to manage the ML life cycle, including experimentation, reproducibility and deployment [6] [7]. It currently offers three components named Tracking, Projects and Models. The MLflow Tracking component is an API and UI for logging parameters, code versions, metrics, and output files when running your machine learning code and for later visualizing the results. MLflow tracking lets you log and query experiments using Python, REST, R API, and Java APIs. MLflow Project is a format for packaging data science code in a reusable and reproducible way, based primarily on conventions. In addition, the Projects component includes an API and command-line tools for running projects, making it possible to chain together projects into workflows. MLflow model is a standard format for packaging machine learning models that can be used in a variety of downstream tools. The format defines a convention that lets you save a model in different “flavors” that can be understood by different downstream tools. These flavors can be specific to libraries like TensorFlow or it can be of generic flavors like Python function. MLflow is focused on scikit-learn library [9] and it allows to serve models on Microsoft Azure Machine Learning and Amazon SageMaker. Support for TensorFlow was introduced in MLflow 0.2 version.

6.2. Polyaxon

Polyaxon is an open source ML life cycle management tool [10]. It is a platform for reproducible and scalable Machine Learning and Deep Learning applications by providing an interactive workspace with notebooks, Tensorboards, visualizations, and dashboards. It helps in collaborating with rest of the team, share and compare experimental results. It has built-in versioning for code and experiments. Polyaxon has a community Edition, an open source platform to manage the complete ML life cyle on Kubernetes, Docker, Or Docker Compose, or any

container management platform. It supports Hyperparameters search and optimization and runs on any infrastructure, Cloud, on-premises or in hybrid environments, including single laptop, container management platform, heroku, on docker-compose, or on Kubernetes.

6.3. Commercial solution

Comet.ml [8] is an automatic versioning solution that tracks and organizes modeling efforts in all stages. It is a commercial solution providing automatic hyper-parameter optimization. The model and training data never leave the development machine. Comet.ml optimization service only requires a few lines of code with no additional dependencies or server setup. Comet.ml organizes all runs as experiments and workspaces contain projects which house the experiments. We need to download the comet library and add the experiment's tracking code in the file. Then run the experiment as normal, and Comet will automatically save the data. To store additional parameters, we can explicitly add them using the experiment API. Once the experiment is logged in Comet, we can review the hyper-parameters, code, and results.

6.4. Git

Git is the versioning protocol used widely to monitor and version software development and deployment. GitHub is web-based commercial implementation of this open-source tool. In ML solutions, results without context information is not useful since it is essential for recreating the model from the training data. Git has missing functionality needed for ML projects in maintaining full context information. Git itself does not allow tracking of changes in data, hyper-parameters, model artifacts and model dependencies. For ML projects, keeping all the folders of the development branch synced in Git is a hard task. The model checkpoint and meta data occupies storage space depending on the model complexity. So, one way is to store all the training and validation data set in the cloud storage and all the reproducible codes in the Git and generate the models on the fly.

Git Data Version Control (DVC) is a Git extension that adds functionality for managing code and data together. It works directly with cloud storage (AWS S3 or Google GCP) to push the changes [5]. According to their tutorial, "DVC streamlines large data files and binary models into a single Git environment and this approach will not require storing binary files in your Git repository." DVC is a streamlined version of combining Git with machine learning specific functionality.

7. Proposed solution

In machine learning projects, we need to store data sets used to train and test the model. For each of the experiments, context information shown in Table.3 are to be checked into the Git repository. All dependent parameters for model building are captured in this table. A unique ID for each entry should be constructed with the project name and experiment number. Typical values for each of the context variables are also shown in the table. Numerical accuracy varies between CPUs, GPUs and TPUs. Hence saving the processor context becomes useful in interpreting the results. Maximum difference between training and validation loss is a measure of overfitting.

Table 3. Contexts and artifacts saved in project repository

Description	Typical values
Experiment number (Run ID)	58
Additional Processors used	GPU, TPU
Number of additional processors used	2 , 3
Time taken for training (in mins)	1200
Number of instances of training data	45000000000
Snapshot (1%) of training data	TrainingSample.csv

Number of instances of validation data	45000
Snapshot (1%) of validation data	ValidationSample.csv
TensorFlow version	2.0
Python version	3.7
Packages / Libraries used	Numpy, Keras, Pandas
ML program source code	Logistics.py
Loss, Optimization, Activation functions used - Listed in Table.1	Functions.txt
Range / Selection of Hyper parameters used - Listed in Table.2	Hyper-parameters.txt
Minimum value of validation loss observed	0.21
Average difference between training and validation loss	0.1
Maximum difference between training and validation loss	0.15
Model Prediction Accuracy in percentage	5
Model Meta graph file	Logistics.meta
Model Check point file	Logistics.ckpt
Name of the data scientist	DataScientist 1

8. Conclusion

The work presented in this paper identified the issues in maintaining repositories for TensorFlow machine learning projects. It is seen that ML specific platforms like MLflow, Polyaxon and Comet.ml are good for maintaining repositories of machine learning projects and controlling iterative refinement of the model building process. They do not allow user selection of context information to be stored in the repository. These tools do not provide support for maintaining repositories of non-ML projects and hence they cannot be used for projects in which ML capabilities are gradually added to traditional software solutions.

The Git source code repositories are traditionally used for version control of software projects. It has been enhanced to support ML projects with some limitations. Git does not support automated refinement of the iterative process of model building. It is recommended to keep the parameters indicated in section 7 on proposed solution also in the Git repository. For developers new to ML work, Git may be a better option because of the familiarity with the Git commands and usage. Git can be a better choice for evolutionary projects in which traditional software is getting enhanced with ML capabilities. The proposed enhancements help in overcoming the limitations of Git being used for ML projects and in hybrid projects consisting of traditional and ML methods.

References

[1] Saleema Amershi et al, “Software Engineering for Machine Learning: A Case Study”, ICSE-SEIP '10 Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice, Montreal, Quebec, Canada — May 27 - 27, 2019, Pages 291-300.

[2] Shuai Zhao et.al, "Packaging and Sharing Machine Learning Models via the Acumos AI Open Platform" , ICMLA 2018: 17th IEEE International Conference on Machine Learning and Applications. December 2018, Orlando.

[3] Ankit Sacha, "A quick complete tutorial to save and restore Tensorflow models" https://cv-tricks.com/tensorflow-tutorial/save-restore-tensorflow-models-quick-complete-tutorial/

[4] Jason Brownlee, "What is the Difference Between a Parameter and a hyper-parameter?", https://machinelearningmastery.com/difference-between-a-parameter-and-a-hyper-parameter/

[5] Justin Gage, "How to version control your production Machine Learning models", https://blog.algorithmia.com/how-to-version-control-your-production-machine-learning-models/

[6] Matei Zaharia, “MLflow: A platform for managing the machine learning lifecycle”, https://www.oreilly.com/ideas/mlflow-a-platform-for-managing-the-machine-learning-lifecycle

[7] Mlflow – www.mlflow.org

[8] Comet.ml - https://www.comet.ml
[9] scikit-learn library- https://scikit-learn.org
[10] Polyaxon - https://polyaxon.com