BenchML
INSO4151 - Capstone Project

Team Roles

- Gabriel Rosa - Backend Developer
    - In charge of the database and back-end development. Will develop the database in PostgreSQL and the backend in Python.
- Enrique Viera - Middleware Developer
    - In charge of routing between front and back end. Will help with development of some front-end features and back-end changes.
- Fernando Davis - Machine Learning Scientist
    - In charge of optimizing and benchmarking code for machine learning models in Python.
    - Aid in backend, middleware, and frontend development.
- Carolina Santiago Pérez - Frontend Developer
    - Will lead the front-end features. Develops the different pages of the application using React.

Problem Statement

How can owners of machine learning models optimize their models and make them more secure?

Project Objectives

Improve a machine learning model's training and validation percentage and ensure a 5% difference

Develop a Gradient-based adversarial attack and measure robustness of object detection classifiers

Develop and expandable and maintainable API to provide future development with more optimization techniques and adversarial attacks.

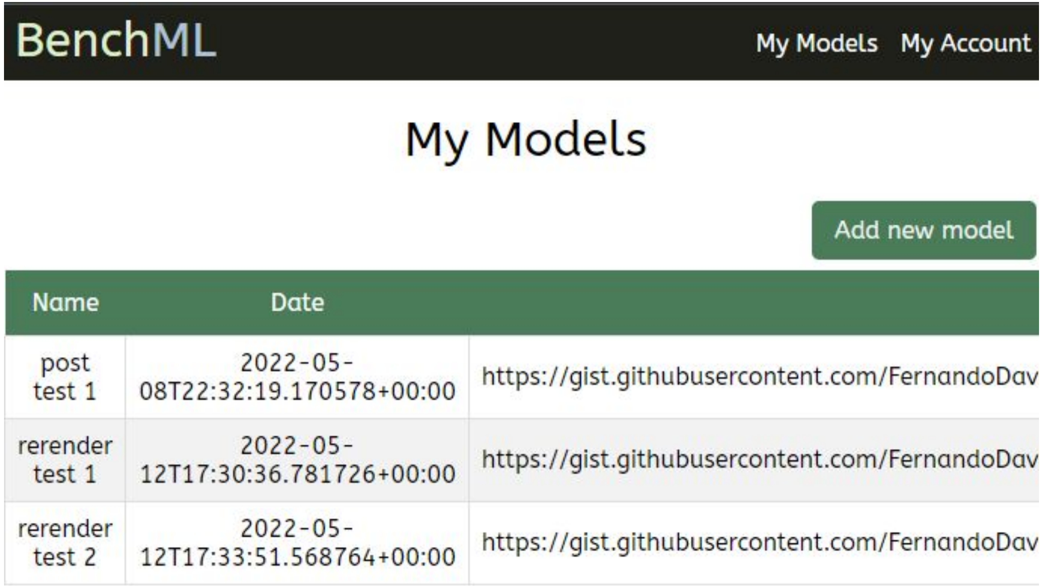
Solution Approach

Web-based framework to benchmark or attack a machine learning model, that can optimize or detect security issues within the model.

Frontend

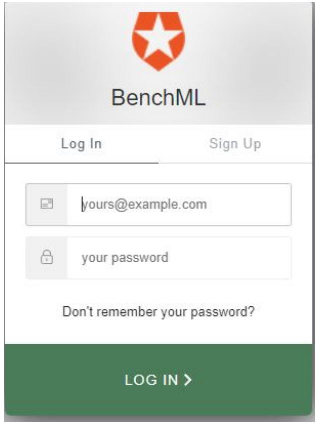
- ReactJS
  - Component based
  - Easy creation of dynamic applications
  - Unidirectional data flow

Frontend

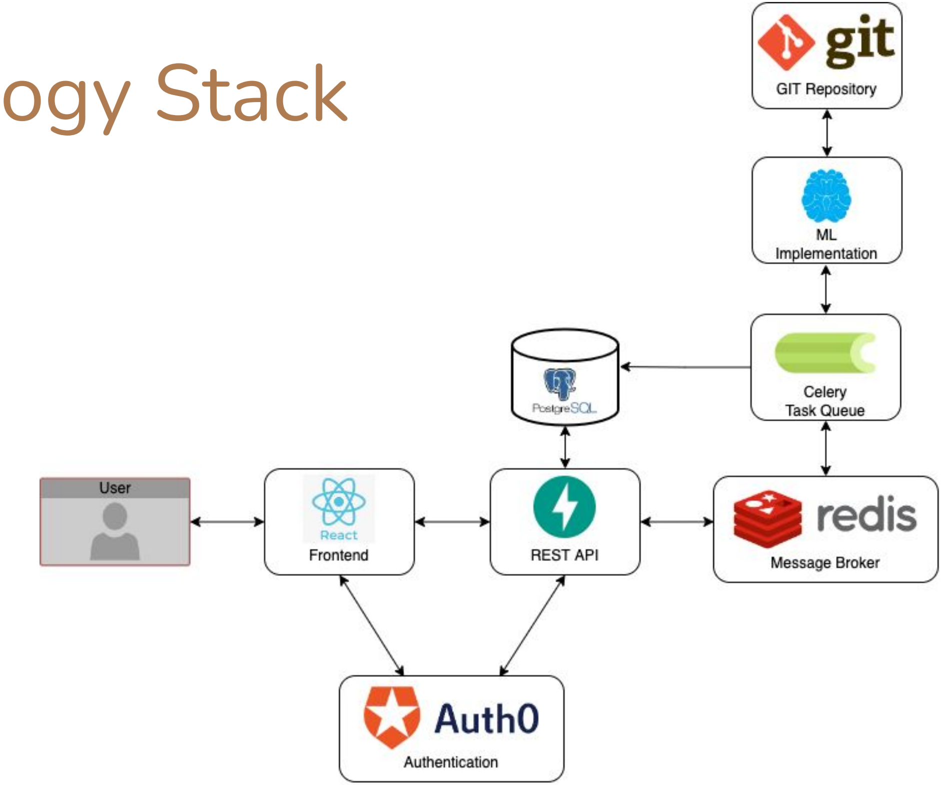


My Models

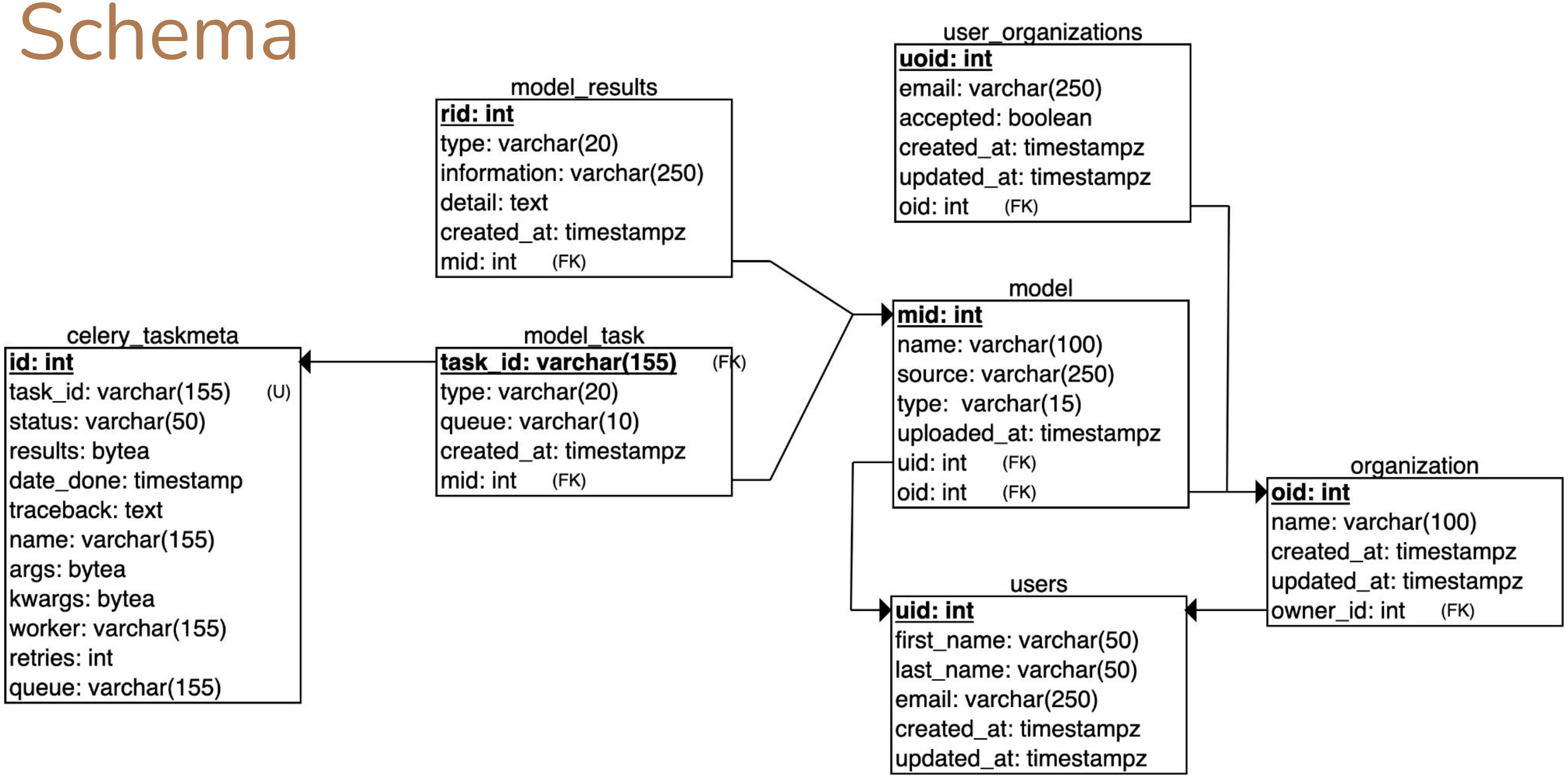
Name	Date	
post test 1	2022-05-08T22:32:19.170578+00:00	https://gist.githubusercontent.com/FernandoDav
rerender test 1	2022-05-12T17:30:36.781726+00:00	https://gist.githubusercontent.com/FernandoDav
rerender test 2	2022-05-12T17:33:51.568764+00:00	https://gist.githubusercontent.com/FernandoDav



Technology Stack



Schema



model_results

rid: int
type: varchar(20)
information: varchar(250)
detail: text
created_at: timestampz
mid: int (FK)

user_organizations

uoid: int
email: varchar(250)
accepted: boolean
created_at: timestampz
updated_at: timestampz
oid: int (FK)

celery_taskmeta

id: int
task_id: varchar(155) (U)
status: varchar(50)
results: bytea
date_done: timestamp
traceback: text
name: varchar(155)
args: bytea
kwargs: bytea
worker: varchar(155)
retries: int
queue: varchar(155)

model_task

task_id: varchar(155) (FK)
type: varchar(20)
queue: varchar(10)
created_at: timestampz
mid: int (FK)

model

mid: int
name: varchar(100)
source: varchar(250)
type: varchar(15)
uploaded_at: timestampz
uid: int (FK)
oid: int (FK)

organization

oid: int
name: varchar(100)
created_at: timestampz
updated_at: timestampz
owner_id: int (FK)

users

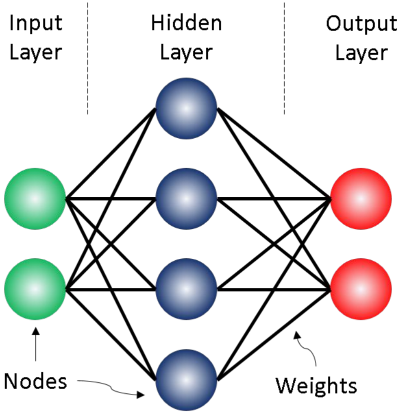
uid: int
first_name: varchar(50)
last_name: varchar(50)
email: varchar(250)
created_at: timestampz
updated_at: timestampz

Machine Learning

- What is machine learning?
    - GPS system
    - Suggested friends system in social media
- Why use machine learning?
    - Pros and cons

Machine Learning - Optimizer

- Hyperparameters determine the network structure and its effectiveness
- Include but are not limited to:
  - Dropout
  - Number of hidden units
  - Batch size
  - Learning rate
  - Number of layers



https://mlnotebook.github.io/post/neuralnetwork/

Machine Learning - Optimizer

- Machine learning models, specifically neural networks, are time and computationally expensive
- Improving model accuracy and results
- Reduce need for hand-tuning of hyperparameters and individual testing

Machine Learning - Optimizer

To achieve good and fast hyperparameter optimization you need:

- Customizable, reliable, and fast optimization algorithms such as:
  - Population Based Training (PBT)
  - Asynchronous Successive Halving Algorithm (ASHA)
- Multi-gpu and multi-node support for the optimizer

Machine Learning - Optimizer

Ray Tune:

- Great optimization algorithms
- Multi-gpu and multi-node support
- Library agnostic modules
- Support for other libraries such as HyperOpt.

Machine Learning - Optimizer

Machine learning model types supported:

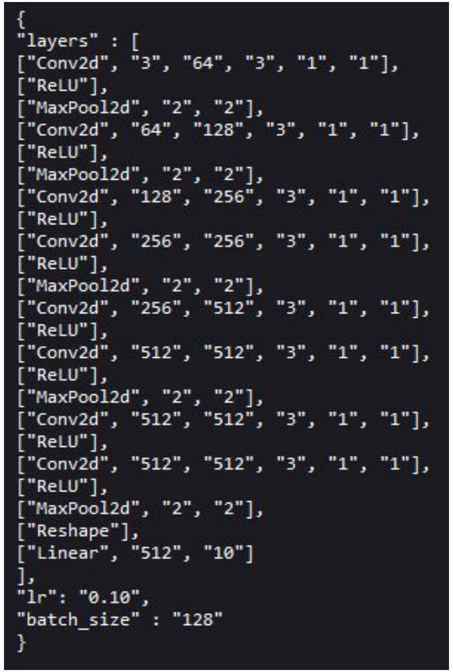
- Tensorflow/keras
- PyTorch

All available modules and layers can be adapted into the optimizer making it entirely modularized with opportunity for extra model integrations and optimization algorithms in the future.

Machine Learning - Adversarial

- layers
- lr (learning rate)
- batch size
- Other blank values can be mapped

to layers directly

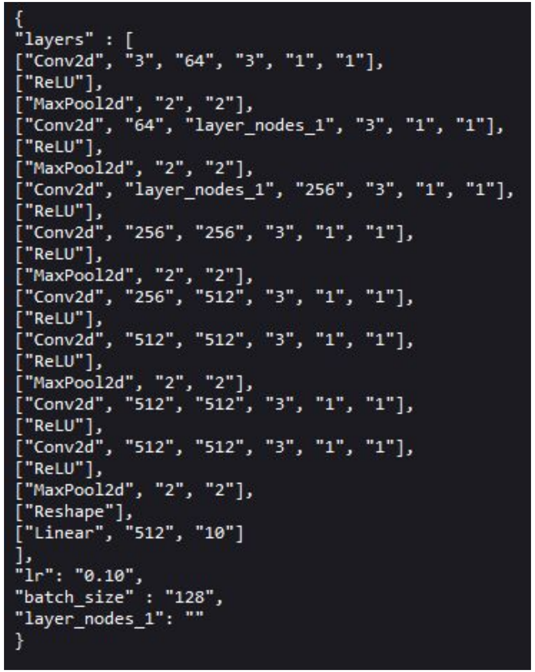


```
{
"layers" : [
["Conv2d", "3", "64", "3", "1", "1"],
["ReLU"],
["MaxPool2d", "2", "2"],
["Conv2d", "64", "128", "3", "1", "1"],
["ReLU"],
["MaxPool2d", "2", "2"],
["Conv2d", "128", "256", "3", "1", "1"],
["ReLU"],
["Conv2d", "256", "256", "3", "1", "1"],
["ReLU"],
["MaxPool2d", "2", "2"],
["Conv2d", "256", "512", "3", "1", "1"],
["ReLU"],
["Conv2d", "512", "512", "3", "1", "1"],
["ReLU"],
["MaxPool2d", "2", "2"],
["Conv2d", "512", "512", "3", "1", "1"],
["ReLU"],
["Conv2d", "512", "512", "3", "1", "1"],
["ReLU"],
["MaxPool2d", "2", "2"],
["Reshape"],
["Linear", "512", "10"]
],
"lr": "0.10",
"batch_size" : "128"
}
```

Machine Learning - Adversarial

- layers
- lr (learning rate)
- batch size
- Other blank values can be mapped

to layers directly



```
{
"layers" : [
["Conv2d", "3", "64", "3", "1", "1"],
["ReLU"],
["MaxPool2d", "2", "2"],
["Conv2d", "64", "layer_nodes_1", "3", "1", "1"],
["ReLU"],
["MaxPool2d", "2", "2"],
["Conv2d", "layer_nodes_1", "256", "3", "1", "1"],
["ReLU"],
["Conv2d", "256", "256", "3", "1", "1"],
["ReLU"],
["MaxPool2d", "2", "2"],
["Conv2d", "256", "512", "3", "1", "1"],
["ReLU"],
["Conv2d", "512", "512", "3", "1", "1"],
["ReLU"],
["MaxPool2d", "2", "2"],
["Conv2d", "512", "512", "3", "1", "1"],
["ReLU"],
["Conv2d", "512", "512", "3", "1", "1"],
["ReLU"],
["MaxPool2d", "2", "2"],
["Reshape"],
["Linear", "512", "10"]
],
"lr": "0.10",
"batch_size" : "128",
"layer_nodes_1": ""
}
```

Machine Learning - Adversarial

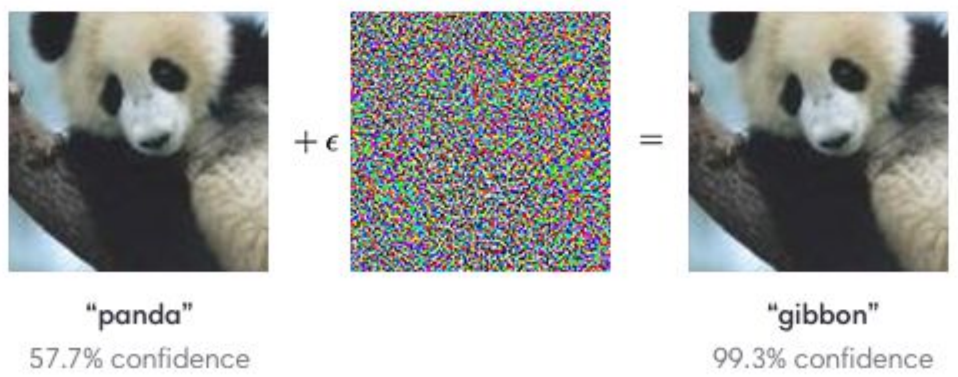
- Model robustness refers to the security and accuracy of a machine learning model
- Depends on:
  - Set of hyperparameters
  - Performance and security breaches

Machine Learning - Adversarial

Adversarial Attacks:

- Main cause of security breaches in modern day machine learning models
- Modifications to input data that is misclassified by computers but correctly classified by humans

Machine Learning - Adversarial



https://openai.com/blog/adversarial-example-research/

Machine Learning - Adversarial

- Fast Gradient Sign Method
    - Introduces perturbations to the image that are non-perceivable to the human eye
    - Does the opposite of what gradient descent tries to achieve, essentially maximizes the loss instead of minimizing it
    - Adds the gradient created to the original input and create the adversarial example that is used to fool the model

Future Work

- Expand the machine learning models that can be optimized and attacked
- Expand the assortment of attacks available to include more cases and evaluations and provide higher confidence intervals
- Provide users with the perturbed dataset to defend against the adversarial attacks

Demo

Questions?