**Project Overview - Job Site Safety**

**Authors: Jason Raimondi,** [Jeremy Cryer](mailto:jcryer@sandiego.edu)**,** [Maimuna Bashir](mailto:mbashir@sandiego.edu)

**Business Name: PPE Vision Solutions**

**Publication Date: 2/26/2024**

**Github Project Link:** [**https://github.com/jerm914/Job-Site-Safety---AAI-540**](https://github.com/jerm914/Job-Site-Safety---AAI-540)

* **Notebooks**
  + **00 - Notebook for setting up data lake (full dataset)**
    - [**https://github.com/jerm914/Job-Site-Safety---AAI-540/blob/main/Main%20Files/00\_S3\_Datalake\_FullDataset.ipynb**](https://github.com/jerm914/Job-Site-Safety---AAI-540/blob/main/Main%20Files/00_S3_Datalake_FullDataset.ipynb)
  + **01 - Notebook for setting up feature store (full dataset)**
    - [**https://github.com/jerm914/Job-Site-Safety---AAI-540/blob/main/Main%20Files/01\_FeatureStore\_FullDataset.ipynb**](https://github.com/jerm914/Job-Site-Safety---AAI-540/blob/main/Main%20Files/01_FeatureStore_FullDataset.ipynb)
  + **02 - Notebook for benchmark model (full dataset)**
    - [**https://github.com/jerm914/Job-Site-Safety---AAI-540/blob/main/Main%20Files/02\_Benchmark\_FullDataset.ipynb**](https://github.com/jerm914/Job-Site-Safety---AAI-540/blob/main/Main%20Files/02_Benchmark_FullDataset.ipynb)
  + **03 - Notebook for CI/CD Pipelines**
    - [**https://github.com/jerm914/Job-Site-Safety---AAI-540/blob/main/Main%20Files/03\_CICD\_Pipelines.ipynb**](https://github.com/jerm914/Job-Site-Safety---AAI-540/blob/main/Main%20Files/03_CICD_Pipelines.ipynb)

**Asana Board Link:** [**https://app.asana.com/0/1206390072539638/1206390072539638**](https://app.asana.com/0/1206390072539638/1206390072539638)

**Project Background:**

1-2 paragraphs describing, at a high level, the problem you are trying to solve. This should provide the reader context for the technical solution they will review. This should be a quick elevator pitch for your project. Be sure to answer the following:

* What is the model’s objective?
* What type of Machine Learning problem will you be solving?

The ultimate objective of the project is to increase safety compliance, specifically adherence to wearing required Personal Protective Equipment (PPE) and reduce occurrences of serious injury on job sites. The objective of the model will be to detect, classify, and report instances of ‘Helmet’, ‘Vest’, and ‘Head’ from a provided input image. Instances of ‘Head’ may indicate absence of a required helmet being worn.

This is an object detection and classification Machine Learning problem to solve. First, the model will need to be able to sufficiently identify when these three types of objects are present in the image. Second, the model will need to be able to accurately classify the detected objects as the correct type or belonging to the correct category.

**Technical Background:**

1-2 paragraphs describing the technical details of the problem you are trying to solve. This should help the reader understand the project constraints. Be sure to answer the following:

* How will you evaluate your model?
* What is your data source?
  + How will you need to prepare your data?
  + How will you explore your data?
  + What do you hypothesize your main features will be?
* What type of model do you want to use?

To evaluate the model, a few different business metrics can be considered. First, the business can monitor and track instances of serious injury over time to see if the model implementation may be improving safety records (i.e., assuming business is intervening with employees when unsafe conditions are investigated). Second, the business can track instances of ‘Head’ being detected in a given time period, work shift, etc. to see if their interventions are having a positive impact (i.e., instances of ‘Head’ reduced over time in favor of instances of ‘Helmet’). Important model losses to minimize during training include box loss (i.e., reduce error in object detection) and classification loss (i.e., reduce errors in classification). During model evaluation, Mean Average Precision (mAP) is a common metric used in object detection problems, as it averages the precision across all of the classes in the dataset. This is the metric we will use to compare performance between model versions.

The data source is from Kaggle:

HardHat-Vest Dataset

<https://www.kaggle.com/datasets/muhammetzahitaydn/hardhat-vest-dataset-v3/data>

The dataset contains 23,673 images, and there are at least 10,000 instances of each class available. Per Kaggle, the images should be consistently sized at 640x640 pixels, so resizing is not expected unless required for the model architecture. However, we plan to still validate the image sizing as claimed. Preparation should be minimal, assuming image annotations are of sufficient quality (expected to be, as the Kaggle usability rating is a 10.00). The dataset should be explored, minimally, by viewing sample images to gain an understanding of contents, considerations, and annotation quality. Bounding boxes will need to be applied to the images to inspect the quality and accuracy. Model features are hypothesized to be constructed via multiple Convolutional Neural Network (CNN) layers to obtain lower-level features and then higher-level features to ultimately support the classification task after detection. An initial suggestion would be to use a pre-trained YOLOv8 model architecture for this project, with fine-tuning for the specific dataset to improve the performance initially and to keep maintainability and adaptability in mind for ongoing use.

**Goals vs Non-Goals:**

Write a bulleted list of (3-5 points each) of goals and non-goals. This should help the reader understand the context that would factor into solution selections and trade-offs. Goals will help the reader understand what a successful outcome looks like. Non-goals will help limit the scope of your project and prevent scope creep.

Goals:

* Reduced instances of serious injury on construction sites (Business Metric)
* Increased safety compliance (wearing required PPE) (Business Metric)
* Mean Average Precision 50 (mAP50) for model evaluation/selection (Model Metric)
  + Low box loss (specific amount to be determined w/ company) (Model Loss, training)
  + Low classification loss (focus on ‘Helmet’ and ‘Head’) (Model Loss, training)

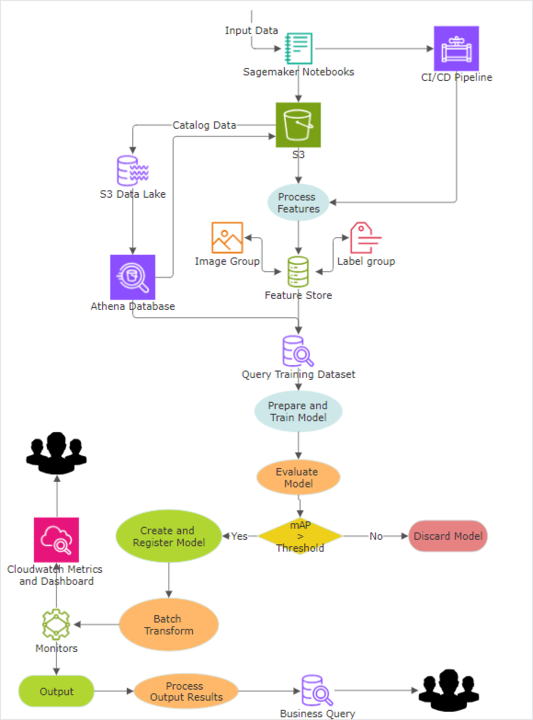
Non-Goals:

* Resolving annotation issues / bounding boxes (if issues identified during exploration, discard from dataset)
* Model metrics for ‘Vest’ category (not the primary focus of this project to maintain simplicity)
* Use of multiple model architectures (also for simplicity)

**Solution Overview**

1-2 paragraphs summarizing your ML System. Include a system architecture diagram containing the components you use to store data, pre-process data, engineer features, build/train/debug your model, and deploy your model. Also add notes on what you will monitor and what you will test prior to releasing a new model.

**System Architecture Diagram**



In summary, we opted to work on a challenging Computer Vision problem using Script Mode instead of an available algorithm. We stored raw image and label files in S3 by using the AWS CLI sync command to reference a local repository. To create tabular data to store within Athena, we chose to create a “catalog” of data based on general properties of the sample files (e.g., file names, extensions). We then processed the raw files to gather and engineer features which we were then able to store in image and label feature groups and make available in both an online and offline feature store configuration. To build our training dataset, we were able to query our catalog data for a preliminary dataset (e.g., JPG files only) then combine it using a primary key (i.e., sample\_id) with our image and label features.

As we opted to use script mode to support our YOLOv8 model, we made use of PyTorch containers and customized preprocessing, train, evaluation, and inference scripts, as well as requirements and yaml files to provide details regarding our custom dataset. Preprocessing was performed using a PyTorch processor and custom script to split our dataset into appropriate directories that our algorithm was expecting. Training was performed using a PyTorch estimator and passing in hyperparameters to our customized train py script. After training, the model weights are saved in a PyTorch format. For evaluation, we again use another PyTorch processor with another custom script to load our model weights and evaluate on the test data split. This then obtains our desired mAP50 metric, adds this to a report dictionary, and outputs this file in json format for later use in our condition step. The condition step will succeed if the metric is above our defined threshold, otherwise it will fail and discard the current model version. If it does succeed, the model will be created and registered in the model registry, and a batch transform job will be created. The transformer is created from our PyTorch model and outputs results in separate JSON files in S3, one file resulting from each individual image. Infrastructure monitors and dashboard are in place to watch over CPU, disk, and memory utilization. Finally, once the batch results are obtained, we process these results to combine them which can then be used for business queries (e.g., observing trends). In terms of this project, we have a manual approval method before deploying any new models in production. Our intention is to obtain a confirmation of a successful User Acceptance Test before manually approving a model for production use.

**Data Sources:**

What is your data source?

Kaggle - HardHat-Vest Dataset

<https://www.kaggle.com/datasets/muhammetzahitaydn/hardhat-vest-dataset-v3/data>

What is your data volume?

23,673 images.

Why did you select this data set?

Our team has interest in the business problem that this dataset can help solve, and this dataset appears to be sufficient to support development of a solution to this problem. There are at least 10,000 instances of each class, including ‘Helmet’, ‘Vest’, and ‘Head’, which should provide a sufficient number of samples for our computer vision task.

Any risks (bias, sensitive features, etc)?

We are not aware of any bias or sensitive features present at this time. However, we will, minimally, inspect image samples and predictions to determine if there could be any considerations. For instance, we would want to make sure that we understand if performance may vary when detecting and classifying objects when considering different genders, race, skin tones, hairstyles, etc. Another risk we need to keep in mind is if there are any types of occlusions that could impact performance. If a worker is wearing sunglasses, a respirator, or if there is any other type of occlusion, does this affect the object being detected and/or classified correctly? This could also include other items such as ball caps and other items that are similar and may conflict with the PPE being detected in this problem. We need to be aware of any limitations such as this so that the solution can be advertised and implemented in suitable environments while providing us the necessary information to continually improve the solution.

**Data Engineering:**

How will you store this data?

What data pre-processing do you need to do before you feed it into your ML system?

As anticipated, we had minimal preprocessing needed for our data. The images are consistently sized at 640x640 pixels, so resizing was not needed. Preparation was minimal, since image annotations are of sufficient quality (as was expected to be, as the Kaggle usability rating is a 10.00). We stored the raw data into S3 initially and then preprocessed and loaded the resulting image/label feature data into the feature store. To simplify our problem, we opted to query our data catalog for only JPG files (i.e., disregarding PNG file types) when building our training dataset.

**Training Data:**

How will you split your data into training, test and validation?

We split the data using the recommendations in the final project instructions:

* ~40% Train
* ~10% Validation
* ~10% Test
* ~40% “Production Data”

Will you use any data labeling techniques?

We did not need to use any data labeling techniques. As mentioned, the Kaggle usability rating is high, and the images come with annotations.

**Feature Engineering:**

Which fields from your data will you use or exclude?

We used class instances of ‘Helmet’ and ‘Head’ but excluded instances of ‘Vest’ in an effort to simplify the implementation and put more focus on the overall ML system.

Which fields will be combined or bucketed?

We did not combine or bucket any fields. We are working on a computer vision problem using raw data and not a dataset with predefined fields.

What other data transformations will you apply to your data?

We are not performing any data transformations but are instead engineering new features derived from properties and contents of the raw data files.

**Model Training & Evaluation:**

How will you train your model?

We opted to use the latest YOLOv8 model available, which is not available natively in AWS. Considering this, we used the SageMaker Python SDK to train our model using the PyTorch framework in “Script Mode” to offer us the customization we required to support this. We created a custom requirements file and training script in order to support our algorithm and save the trained model in a PyTorch model format.

What algorithm will you use?

We used a PyTorch estimator to provide compatibility with our intended YOLO model deployment.

What parameters will you use?

We used the following parameters when training our model:

* framework\_version - 2.1.0
  + This is the PyTorch framework version to allow for training on CPU, as GPU is not available in this lab environment. It also offers support for Python 3.10.
* py\_version - py310
  + This specifies the Python version as 3.10.
* instance\_count - 1
* instance\_type - ml.m5.xlarge
* Hyperparameters
  + The following are custom hyperparameter arguments we passed in to our training py script. The values vary depending on the specific model variant.
    - data
      * Specifies the .yaml config file for our custom dataset
    - Epochs
      * Specifies the number of training epochs
    - Batch
      * Batch size
    - Yolo\_model
      * Specifies the pretrained yolo base model for instantiating an initial model instance
    - Saved\_model\_weights
      * Specifies the name for the saved model weights which are saved and loaded later during inference

How will you evaluate your model?

We used Mean Average Precision 50 (i.e., mAP50) as our evaluation metric to compare performance between models. This is the metric we defined a threshold for to also use during a condition step in the pipeline.

**Model Deployment:**

What instance size will you use?

We primarily used ml.m5.xlarge for the instance size, other than also specifying an accelerator type of ml.eia1.medium for the model step and ml.t2.medium for inference purposes. We did attempt to use a memory-optimized instance (e.g., ml.r5.xlarge), however, we encountered errors in the lab environment indicating that it was not a supported option to use.

Regarding the YOLOv8 model, we used the “nano” variant with 3.2m parameters, as the AWS student environment does not have access to GPUs. We prioritized simplicity and computational efficiency to work within these constraints and focus efforts on the overall ML system. The larger model variants are in the range of 11.2 to 68.2m parameters, in comparison.

Will your model function as a batch or real time model? Why?

Our model functions as a batch. This will increase computational efficiency to process in batches. It is also not a requirement of our ML solution to process and produce results in or near real-time. It is instead to provide insights on trends in the data.

**Model Monitoring:**

How will you monitor your model?

Initially, we planned to monitor the model by observing the distribution of predictions and tracking concept drift over time. However, due to the added complexity of our Computer Vision use case, we decided to postpone this for future work.

How will you monitor your infrastructure?

To effectively monitor our infrastructure, we tracked CPU utilization, which offers vital insight into the efficiency of our processing power usage. We also observed disk utilization, ensuring our storage systems operate optimally without excess burden. Additionally, we monitored memory usage, another crucial aspect for ensuring smooth operations.

How will you monitor your data?

We explored numerous methods to monitor our data, aiming to find the optimal solution. We identified that ensuring data consistency and integrity would be most beneficial. However, given the added complexity of our Computer Vision use case, we decided to defer this to future work.

Due to additional complexity with our Computer Vision use case and using Script Mode with container customizations, we opted to focus on the infrastructure metrics and monitoring and save model and data metrics as future work. One of the challenges presented is that the default model monitor expects a dataset in the format of either CSV, JSON, or parquet. However, we instead have images. Considering this, we still opted to go through the steps of creating baseline results and a monitoring schedule. We did this by using our catalog data as input, which is in CSV format. However, we would desire to plan for additional engineering efforts as future work to construct this monitor in a more robust way. The catalog data on its own does not have much useful information for a real-world use case. We would want to instead consider potentially using our feature store data, exporting it to a compatible format, and using this format as input for our batch job instead of directly using our image format. Alternatively, we could research additional monitoring methods that may be able to handle image data.

**Model CI/CD:**

What checkpoints will your CI/CD pipeline contain?

Minimally, we had to include checkpoints for data quality and model evaluation. Although we appear to have a relatively clean, standardized dataset to initially work with, any future incoming data will need to meet the same level of quality to be usable. For instance, if camera settings are changed and training images begin coming in at different sizes or significantly different lighting conditions, etc., additional data preparation may be necessary to be compatible with the current system, or other components of the system may need to be adapted (e.g., model training methods). Model evaluation is a necessary checkpoint to confirm improved performance prior to considering movement to the deployment phase.

What tests will your CI/CD pipeline contain?

Minimally, we did include tests for model performance (during inference) and user acceptance. This was important for us to validate that the model is performing as we expect it to in production on new, unseen data. It is equally important that the customer or business performs a user acceptance test to validate performance and features in each deployment. As we included a manual approval step in the pipeline, new models will only be approved and deployed to production once we have verified that the user acceptance criteria has been met.

**Security Checklist, Privacy and Other Risks:**

Will this store or process Personal Health Information (PHI)? No

Will this store or process Personal Identifiable Information (PII)? Yes

Will user behavior be tracked and stored? No

Will this store or process credit card information? No

If you answered yes to any of the above questions, please justify.

Our ML system does store and process photographs of individuals at job sites, and these photographs can personally identify an individual. Appropriate measures are to be taken so that we, as ML solution providers, as well as our customers (businesses) and individuals at these sites are aware of and agree to these privacy risks. Generally, as a condition of employment and/or contract, individuals accessing the site are acknowledging and agreeing to the storage and processing of their PII.

What S3 buckets will this application read from or write to?

We used a number of S3 buckets within our default bucket:

* safety-featurestore
  + This includes the offline store for our image and label feature groups
* safety
  + catalog
    - This includes the query results from our catalog CSV file
  + data
    - images
      * Location of raw image files
    - labels
      * Location of raw label files
    - output
      * Stores output of jobs, including model.tar.gz files
    - split
      * Sample dataset split into train, test, val, and batch
    - split\_full
      * Full dataset split into train, test, val, and batch
    - split\_cicd
      * Larger sample used when working on CI/CD iterations

What data bias should be considered?

This is discussed in more detail in the Data Sources section of this document. As of now with the data we have, bias is hard to control and identify. Realistically bias will come from human features that would need to be better understood of what is existing in the photos we currently have. Right now there is not enough time and resources to go through the images and identify this risk as human traits aren't categorized with the data. If we had data control with a constant incoming of new photos we could have a team review all images and identify characteristics that may not be as varying and could cause an identification issue later on. Trial and error would be the best strategy to root bias out with our current dataset.

Will your model have potential for bias along sensitive features (race, ethnicity, gender, age, religion, disability, sexual orientation, or other personal attributes)?

Yes, as mentioned in the previous response, race, ethnicity, and gender could be a potential for bias. Additionally, age could be another potential. These are not features or biases that, if present, are expected to have a direct, individual impact. However, these types of biases, if present, could impact the system performance and result in negative safety trends. This is something that would need to be rooted out on model trial and testing done on additional iterations. We would also need control over what data is incoming to better control this.

Are there any ethical concerts with the data or business problems that should be addressed?

The only ethical concerns for this model currently are for the diversity of data being used. We do not have an influx of incoming data and only have a set from Kaggle. Ideally being able to have a working group assess the diversity of data being constantly trained on would key. In this initial iteration we only have what we have so this is not something we can properly account for. In a second iteration with live data, a better assessment and diversity control would be possible. Otherwise the model currently doesn't have any ethical concerns on how the architecture is designed.

**Future Enhancements:**

Provide at least 3 ways you would improve your ML system if you had more time or additional resources.–

1. Increase model instance size and complexity
   1. In the case we have access to a GPU-enabled environment
2. Scale the business use case
   1. Detect instances of ‘Vest’ in the dataset
   2. Contractor badges - different colors
   3. Correlation between companies
   4. Lack of PPE
   5. Are the right/expected or unauthorized persons in the area
3. Consider a video-feed system for real-time monitoring and alerting
   1. Will allow constant incoming data as well to better assess diversity of images as well
4. Implement facial recognition to allow images to be labeled by individual people
   1. Allows individuals to opt out