Here's the step-by-step guide For PPE Detection Model Training

Step 01: Install Required Packages

🛠️ **Install necessary PYTHON libraries for model training, dataset handling, and video downloading.**
After installation, remember to restart the runtime to apply changes.

python

Copy code

```
!pip install super-gradients==3.1.0

!pip install imutils

!pip install roboflow==1.0.6

 # Use latest Roboflow version for compatibility

!pip install pytube --upgrade
```

Step 02: Import Necessary Libraries

📚 **Import essential libraries, including components from SuperGradients for model training and metric evaluation.**

```
from super_gradients.training import Trainer

from super_gradients.training import dataloaders

from super_gradients.training.dataloaders.dataloaders import coco_detection_yolo_format_train,
coco_detection_yolo_format_val

from IPython.display import clear_output

from super_gradients.training.losses import PPYoloELoss

from super_gradients.training.metrics import DetectionMetrics_050

from super_gradients.training.models.detection_models.pp_yolo_e import
PPYoloEPostPredictionCallback
```

```
from super_gradients.training import models
```

Step 03: Set Up Checkpoint Directory and Trainer

💾 **Define a directory to store checkpoints and initialize the Trainer to manage experiments and configurations.**

```
CHECKPOINT_DIR = '/content/checkpoints'

trainer = Trainer(experiment_name='ppe_yolonas_run', ckpt_root_dir=CHECKPOINT_DIR)
```

Step 04: Import Dataset from Roboflow

📥 **Use your Roboflow API key to import the dataset directly. Update with the latest version or adjust the parameters based on your dataset's requirements.**

```
from roboflow import Roboflow


rf = Roboflow(api_key="YOUR_API_KEY")

# Replace "YOUR_API_KEY" with your Roboflow API key

project = rf.workspace("project-uyrxf").project("ppe_detection-v1x3l")

dataset = project.version(2).download("yolov5")

# Ensure this version is available in your Roboflow project
```

Step 05: Set Up Dataset Parameters

🗂️ **Define dataset paths and class labels to keep configurations organized.**

```
dataset_params = {

    'data_dir': '/content/PPE_Detection-2',

  # Root directory for the dataset
```

```
    'train_images_dir': 'train/images',

    'train_labels_dir': 'train/labels',

    'val_images_dir': 'valid/images',

    'val_labels_dir': 'valid/labels',

    'test_images_dir': 'test/images',

    'test_labels_dir': 'test/labels',

    'classes': ['Dust Mask', 'Eye Wear', 'Glove', 'Protective Boots', 'Protective Helmet', 'Safety Vest', 'Shield']

}
```

Step 06: Load Dataset for Training, Validation, and Testing

🚀 **Load the dataset for training, validation, and testing using defined parameters.**

```
train_data = coco_detection_yolo_format_train(

    dataset_params={

        'data_dir': dataset_params['data_dir'],

        'images_dir': dataset_params['train_images_dir'],

        'labels_dir': dataset_params['train_labels_dir'],

        'classes': dataset_params['classes']

    },

    dataloader_params={

        'batch_size': 16,

        'num_workers': 2

    }

)


val_data = coco_detection_yolo_format_val(
    dataset_params={

        'data_dir': dataset_params['data_dir'],

        'images_dir': dataset_params['val_images_dir'],

        'labels_dir': dataset_params['val_labels_dir'],

        'classes': dataset_params['classes']

    },

    dataloader_params={

        'batch_size': 16,

        'num_workers': 2

    }

)



test_data = coco_detection_yolo_format_val(

    dataset_params={

        'data_dir': dataset_params['data_dir'],

        'images_dir': dataset_params['test_images_dir'],

        'labels_dir': dataset_params['test_labels_dir'],

        'classes': dataset_params['classes']

    },

    dataloader_params={

        'batch_size': 16,

        'num_workers': 2

    }

)
```

Step 07: Inspect the Dataset

🔍 Inspect dataset configurations and adjust transformations if necessary. This step allows you to visualize and fine-tune data augmentation.

```
# Visualize the applied transformations

print(train_data.dataset.transforms)

train_data.dataset.dataset_params['transforms'][1]['DetectionRandomAffine']['degrees'] = 10.42

# Adjust transformation parameters if needed

train_data.dataset.plot()

# Plot to visualize data augmentations
```

By following each step carefully—from installing packages and importing libraries to configuring datasets and inspecting transformations—you have established a solid foundation for successful PPE detection model training. The organized configurations allow for efficient experimentation and performance evaluation for real-world PPE detection tasks.