**Overall Process of the Project in high-level View**

**1. Data Augmentation and Preparation**

This includes operations like horizontal flipping, vertical flipping, diagonal flipping, and various rotations. For each transformation, corresponding JSON files are created that map the bounding boxes and the classes (rotated or missing) for objects in the images. These data augmentations help in creating a robust dataset, which is essential for training a reliable model that can generalize well on unseen data.

**2. Object Detection Model – a pre-trained model already has weights (Transfer learning)**

The object detection model is based on a pre-trained Faster R-CNN network with a ResNet-50 backbone, modified to detect specific objects (components) on a PCB.

**Here's the key workflow:**

***Model Setup:*** The Faster R-CNN model is adapted to the number of classes (types of components on the PCB) plus a background class.

***Training:*** The model is trained using the augmented images and their corresponding bounding box annotations (from JSON files). The bounding boxes help the model learn where each component is located on the PCB.

***Detection Task:*** During inference, this model predicts bounding boxes and class labels (type of component) for each component visible in the image.

**3. Classification Model – a pre-trained model already assigned weights (Transfer Learning)**

After detecting the components, a classification model is employed to determine whether each component is rotated or missing.

***This step involves:***

***Model Setup:*** A VGG-16 model with batch normalization, pre-trained on ImageNet, is used. The final layer is replaced to output predictions for two classes the number of components (rotated or missing).

***Training:*** The inputs to this model are the cropped images of components based on the bounding boxes identified by the object detector. It learns to classify each component as either rotated or missing.

***Classification Task:*** For each detected component, the model classifies it as rotated or missing.

**4. Integration and Inference**

During the inference (demo mode), the workflow integrates ***both models***:

***Object Detection:*** First, the object detection model identifies components and their locations.

***Classification:*** For each detected component, the classification model determines if it is rotated or missing.

***Final Output(demo):*** The final output specifies which components are missing or rotated based on the detections and classifications.

**5. Practical Deployment**

The models are ***trained separately*** but used ***sequentially during inference*.**

**Object Detection Part**

**Architecture of Object Detection Model**

***Backbone Network:*** Faster R-CNN typically uses a pre-trained backbone network, such as ResNet, to extract feature maps from the input images. In this code, it uses ResNet-50 as the backbone.

***Region Proposal Network (RPN):*** The RPN generates region proposals (bounding boxes) that are likely to contain objects. It slides a small network over the convolutional feature maps produced by the backbone network to predict bounding boxes and objectness scores for these regions.

***Region of Interest (RoI) Pooling or RoI Align***: RoI pooling or RoI align is used to extract fixed-size feature maps from the convolutional feature maps for each proposed region. These features are then fed into the subsequent layers for classification and bounding box regression.

***Classification Head:*** This part of the network classifies the proposed regions into different object classes and background. In the provided code, the classification head is replaced with a custom head suitable for the specific classification task.

***Bounding Box Regression Head:*** Along with classification, Faster R-CNN also predicts the bounding box coordinates (offsets) for each proposed region to refine the bounding boxes.

***Anchor Boxes:*** Anchor boxes are predefined bounding boxes of different scales and aspect ratios that are used by the RPN to generate region proposals. These anchor boxes cover various spatial locations in the feature maps.

**Below is how the overall object detection model works with the provided image data and XML**

***Object Detection Model:*** The Faster R-CNN model with a ResNet-50 backbone is initialized with pre-trained weights. This means that the model's convolutional layers have already been trained on a large dataset (like ImageNet) to recognize general features in images. By starting with these pre-trained weights, the model has already learned to detect basic shapes, edges, textures, and other visual patterns. During training on the custom dataset, the model fine-tunes its parameters to adapt to the specific task of detecting objects in printed circuit boards (PCBs).

**How the labelling works (XML)**

Annotations are required for this model to locate and identify components on the PCB images. Each bounding box in the annotation (.xml) files corresponds to a component on the PCB. These bounding boxes provide the location information necessary for the object detection model to learn where the components are placed on the PCB and what their shapes are. The presence or absence of bounding boxes indicates whether components are present or missing. Additionally, the <rotated> field in the annotation files indicates if a component is incorrectly oriented, which helps in detecting rotated components

**Classification: Part**

**The architecture of the classification model**

***Convolutional Layers:*** VGG-16 is made up of a series of convolutional layers stacked upon each other. These layers use filters (or kernels) to perform convolution operations that capture spatial hierarchies between pixels by learning image features at various levels of abstraction. For instance, early layers might capture edges and textures, while deeper layers can capture more complex patterns like shapes or specific parts of objects.

***Activation Function****:* After each convolution operation, a non-linear activation function called ReLU (Rectified Linear Unit) is applied to introduce non-linear properties to the system, helping it learn more complex patterns.

***Pooling Layers:*** These layers reduce the spatial size of the representation, making the detection of features invariant to scale and orientation changes and also reducing the computational complexity.

***Fully Connected Layers:*** After several convolutional and pooling layers, the network uses fully connected layers where every input is connected to every output (in the context of the layer). This part of the network is where the final decision-making happens based on the features extracted by the convolutional layers.

***Object Classification Model:*** Similarly, the VGG-16-based classification model initializes its weights with pre-trained values. VGG-16 has been pre-trained on a large dataset to classify images into various categories. By using these pre-trained weights as a starting point, the model already has some understanding of different visual features and can be fine-tuned to classify PCB images into specific classes relevant to the task at hand.

**Training the classification model with XML if training is needed (since this is transfer learning, it’s not a must to train)**

**Example: Rotated Components:**

The <rotated> tag in the XML annotations indicates whether a component is rotated (marked as <rotated>1</rotated>) and can process the XML files to identify components that are marked as rotated.

Similar to missing components,

**Summary of XML Labeled Files:**

* The labeled files for training would consist of pairs of input images (PCB images) and corresponding labels indicating the presence of missing or rotated components.
* For each PCB image with missing or rotated components, it creates a training sample with the image as input and the corresponding label as the target output.
* These training samples would be used to train the classifier model to detect missing and rotated components on PCB images.

**Overall Training Process is like this: (if training is needed):**

**Object Detection Model Definition:** The first step is to define the object detection model, which includes specifying the architecture (Faster R-CNN with ResNet backbone) and loading a pre-trained model for evaluation.

**Object Detection Training:** Once the object detection model is defined, the next step is to train this model using the augmented dataset. This step builds upon the previous one, as the training requires a defined model.

**Object Classification Model Definition:** After completing the object detection training, the process moves on to defining the object classification model. This involves specifying the architecture (VGG-16 with batch normalization), which is independent of the object detection model but follows it in the workflow.

**Object Classification Training:** Finally, the object classification model is trained using the augmented dataset. Similar to the object detection training, this step builds upon the model definition and requires the dataset prepared during the object detection training.

**Prediction Process with Test Data:**

**Object Detection Inference:**

The object detection model is applied to the new image to detect objects present within it. This involves passing the image through the pre-trained Faster R-CNN model with a ResNet backbone.

The model identifies regions of interest (ROIs) where objects are likely to be located and predicts bounding boxes around these objects along with their corresponding class labels.

These bounding boxes and class labels are then used to localize and classify objects within the image.

**Object Classification Inference:**

Once the objects are detected and localized within the image, the object classification model is applied to each detected region of interest (ROI) individually.

For each ROI, the model predicts the class label of the object contained within it. This involves passing the cropped image region through the pre-trained VGG-16 model with batch normalization.

The model outputs the class probabilities for each possible object category, and the label with the highest probability is assigned to the object within the ROI. (rotated or missing)

**Additional:**

**File Structure:**

**Main Script:** This is the main Python script that orchestrates the entire process. It includes the steps for object detection, object classification, and data preparation.

**Object Detector:** This component is responsible for detecting objects within images. It utilizes a pre-trained Faster R-CNN model for object detection.

**Object Classifier:** This component classifies the detected objects into different categories. It uses a pre-trained VGG-16 model for object classification.

**Data Prep:** This component is responsible for preparing the data for training the object detector and classifier. It includes data augmentation techniques and generates annotations in XML and JSON formats.

**Image Files:** These are the input images on which object detection and classification are performed.

**XML Files:** These files contain annotations for the objects in the images in XML format.

**JSON Files:** These files contain annotations for the objects in the images in JSON format.

**Read the PDF to understand: Annotation (labeling data)**

****