2.1 Source Code Overview

Folder: /artefacts/source_code/

Description:
The source code implements a **complete inference pipeline** for detecting cracks on aircraft fuselages using a Roboflow-hosted computer vision model. The pipeline is written in **Python** and performs the following steps:

1. Loads a local image provided by the user.
2. Sends the image to the Roboflow **inference API** using a secure API key and the model URL.
3. Receives the **prediction results** from the model.
4. Draws **red bounding boxes** around detected cracks.
5. Saves and displays the resulting annotated image.

Key Technical Points:

- Communication with Roboflow via **HTTP requests**.
- Secure handling of the **API key**.
- Parsing of **JSON prediction outputs**.
- Image processing to **draw bounding boxes** around predicted objects.

Main.py – Main Script

This script handles the complete inference pipeline: sending an image to Roboflow, receiving predictions, converting coordinates, and drawing bounding boxes.

Imports and Configuration

```
import sys
import json
from inference_sdk import InferenceHTTPClient
from utils import draw_boxes


API_URL = "https://serverless.roboflow.com"
API_KEY = "0ni1aLOl4URBmjk08gjY"
MODEL_ID = "my-first-project-utgik/1"
```

Function: Convert Predictions

```
def convert_predictions_to_boxes(predictions):
    boxes = []
    for pred in predictions:
        x, y = pred["x"], pred["y"]
        w, h = pred["width"], pred["height"]

        x1 = x - w / 2
        y1 = y - h / 2
        x2 = x + w / 2
        y2 = y + h / 2

        boxes.append([x1, y1, x2, y2])
    return boxes
```

- Converts Roboflow predictions (x, y, width, height) into [x1, y1, x2, y2] format.
- (x, y) is the center of the box, so conversion is necessary for drawing rectangles.

Main Function

```
def main():
    if len(sys.argv) < 2:
        print("Usage : python main.py image.jpg")
        sys.exit(1)

    image_path = sys.argv[1]
    output_path = "output.jpg"

    client = InferenceHTTPClient(api_url=API_URL, api_key=API_KEY)
    print("Sending image to model...")
    result = client.infer(image_path, model_id=MODEL_ID)

    print("\n--- JSON Response ---")
```

```
    print(json.dumps(result, indent=2, ensure_ascii=False))

    predictions = result.get("predictions", [])
    boxes = convert_predictions_to_boxes(predictions)

    print("\nDrawing boxes...")
    draw_boxes(image_path, boxes, output_path)

    print(f"\nAnnotated image saved as: {output_path}")

if __name__ == "__main__":
    main()
```

Workflow:

1. Reads image path from command-line argument.
2. Sends image to Roboflow API and receives JSON predictions.
3. Converts predictions to bounding boxes.
4. Draws bounding boxes using draw_boxes().
5. Saves the annotated image.

Utils.py – Drawing Helper

This module handles drawing **red rectangles** around predicted boxes on an image.

```
from PIL import Image, ImageDraw

def draw_boxes(input_image_path, boxes, output_image_path):
    with Image.open(input_image_path) as img:
        draw = ImageDraw.Draw(img)
        outline_color = (255, 0, 0)  # red
        line_width = 3

        for box in boxes:
```

```
    try:
        x1, y1, x2, y2 = box
        rect = [int(x1), int(y1), int(x2), int(y2)]
        draw.rectangle(rect, outline=outline_color, width=line_width)
    except Exception:
        continue

img.save(output_image_path)
```

Key Points:

- Opens the input image using Pillow.
- Draws red rectangles for each bounding box.
- Saves the annotated image to the specified output path.
- Skips any malformed boxes silently.

Pipeline Summary

1. **User Command:**

python main.py image.jpg

2. main.py sends the image to Roboflow.
3. Roboflow returns JSON predictions.
4. Predictions are converted to [x1, y1, x2, y2] bounding boxes.
5. draw_boxes() draws red rectangles around cracks.
6. Annotated image is saved as output.jpg.