```
!pip install ultralytics gradio opencv-python
```

Show hidden output

```
with gr.Blocks(
    css="""
    body { overflow: hidden !important; }
    .gradio-container { min-height: 100svh !important; }
    """
) as demo:
    ...
```

```
/ipython-input-3573456719.py:1: DeprecationWarning: The 'css' parameter in the Blocks constructor will be removed in Gradio 6
th gr.Blocks(
```

```
demo.launch(inline=True)
```

Show hidden output

Next steps: Deploy to Cloud Run

```
import gradio as gr
import cv2
import time
from ultralytics import YOLO

# ---------------- LOAD MODEL ----------------
model = YOLO("yolov8n.pt")

# Emission factors (kg CO2 per second of idling)
EMISSION_MAP = {
    "car": 0.02,
    "bus": 0.08,
    "truck": 0.12,
    "motorcycle": 0.01
}

# ---------------- CORE LOGIC ----------------
def analyze_traffic(video, current_waits, total_saved, is_running, frame_idx):
    if not is_running or video is None:
        yield "System Offline", [], current_waits, total_saved, f"{total_saved:.2f} kg CO2", frame_idx
        return

    # Get video path safely
    video_path = video if isinstance(video, str) else video["name"]
    cap = cv2.VideoCapture(video_path)

    fps = cap.get(cv2.CAP_PROP_FPS)
    if fps <= 0:
        fps = 30

    # 🔑 Move to NEW frame for every cycle
    cap.set(cv2.CAP_PROP_POS_FRAMES, frame_idx)

    ret, frame = cap.read()
    if not ret:
        frame_idx = 0
        cap.set(cv2.CAP_PROP_POS_FRAMES, 0)
        ret, frame = cap.read()

    lane_data = {
        "North": {"count": 0, "emit_rate": 0},
        "South": {"count": 0, "emit_rate": 0},
        "East":  {"count": 0, "emit_rate": 0},
        "West":  {"count": 0, "emit_rate": 0}
    }

    # ---------------- DETECTION ----------------
    if ret:
        h, w, _ = frame.shape
        results = model(frame, verbose=False)[0]

        for box in results.boxes:
            label = model.names[int(box.cls[0])]
            if label not in EMISSION_MAP:
```

```
                continue

            x_center = int((box.xyxy[0][0] + box.xyxy[0][2]) / 2)
            y_center = int((box.xyxy[0][1] + box.xyxy[0][3]) / 2)

            # ✅ REALISTIC 4-WAY INTERSECTION LOGIC
            if y_center < h * 0.5 and abs(x_center - w * 0.5) < w * 0.25:
                lane = "North"
            elif y_center >= h * 0.5 and abs(x_center - w * 0.5) < w * 0.25:
                lane = "South"
            elif x_center < w * 0.5:
                lane = "West"
            else:
                lane = "East"

            lane_data[lane]["count"] += 1
            lane_data[lane]["emit_rate"] += EMISSION_MAP[label]

    cap.release()

    # ---------------- ANALYTICS ----------------
    table_data = []
    priority_scores = {}

    for lane in lane_data:
        idling_risk = lane_data[lane]["emit_rate"] * current_waits[lane]
        score = (idling_risk * 10) + (current_waits[lane] * 0.5)
        priority_scores[lane] = score

        table_data.append([
            lane,
            lane_data[lane]["count"],
            round(idling_risk, 3),
            current_waits[lane],
            round(score, 2)
        ])

    # ---------------- SIGNAL DECISION ----------------
    green_lane = max(priority_scores, key=priority_scores.get)

    duration = min(
        max(15 + lane_data[green_lane]["count"] * 3, 15),
        60
    )

    total_saved += lane_data[green_lane]["emit_rate"] * duration

    new_waits = {
        l: 0 if l == green_lane else current_waits[l] + duration
        for l in current_waits
    }

    # ⏩ Advance video for NEXT cycle
    frame_idx += int(fps * 2)

    # ---------------- COUNTDOWN ----------------
    for remaining in range(duration, -1, -1):
        status = f"🟢 GREEN SIGNAL: {green_lane} | ⏱️ {remaining}s"
        if remaining == 0:
            status = "🔴 SWITCHING SIGNAL – RE-SCANNING..."

        yield (
            status,
            table_data,
            new_waits,
            total_saved,
            f"{total_saved:.2f} kg CO2",
            frame_idx
        )
        time.sleep(1)

# ---------------- UI ----------------
with gr.Blocks(theme=gr.themes.Soft(), title="Smart Traffic Management System",css="""
    body { overflow: hidden !important; }
    .gradio-container { min-height: 100svh !important; }
    """) as demo:
    gr.Markdown("# 🚦 Smart Traffic Management System for Emission Reduction")

    waits_state = gr.State({"North": 30, "South": 30, "East": 30, "West": 30})
    savings_state = gr.State(0.0)
    frame_state = gr.State(0)
    running_flag = gr.State(False)
```

```
        with gr.Row():
            with gr.Column(scale=1):
                video_in = gr.Video(label="Traffic Video Input")
                carbon_display = gr.Label(
                    label="🌿 TOTAL CO2 EMISSION SAVED",
                    value="0.00 kg CO2"
                )
                btn_start = gr.Button("🚀 START SYSTEM", variant="primary")
                btn_stop = gr.Button("🛑 STOP SYSTEM", variant="stop")

            with gr.Column(scale=2):
                status_box = gr.Textbox(
                    label="Live Signal Status",
                    interactive=False
                )
                stats_table = gr.Dataframe(
                    headers=[
                        "Lane",
                        "Vehicles",
                        "Idling Risk (kg CO2)",
                        "Wait (s)",
                        "Priority Score"
                    ],
                    label="Intersection Analytics"
                )

        btn_start.click(
            lambda: True,
            outputs=running_flag
        ).then(
            analyze_traffic,
            inputs=[
                video_in,
                waits_state,
                savings_state,
                running_flag,
                frame_state
            ],
            outputs=[
                status_box,
                stats_table,
                waits_state,
                savings_state,
                carbon_display,
                frame_state
            ]
        ).then(
            analyze_traffic,
            inputs=[
                video_in,
                waits_state,
                savings_state,
                running_flag,
                frame_state
            ],
            outputs=[
                status_box,
                stats_table,
                waits_state,
                savings_state,
                carbon_display,
                frame_state
            ]
        )

        btn_stop.click(lambda: False, outputs=running_flag)

    if __name__ == "__main__":
        demo.launch()
```

```
Creating new Ultralytics Settings v0.0.6 file ✅
View Ultralytics Settings with 'yolo settings' or at '/root/.config/Ultralytics/settings.json'
Update Settings with 'yolo settings key=value', i.e. 'yolo settings runs_dir=path/to/dir'. For help see https://docs.ultraly
Downloading https://github.com/ultralytics/assets/releases/download/v8.4.0/yolov8n.pt to 'yolov8n.pt': 100% ━━━━━━━━━━━━ 6.2
/tmp/ipython-input-2377679525.py:127: DeprecationWarning: The 'theme' parameter in the Blocks constructor will be removed ir
  with gr.Blocks(theme=gr.themes.Soft(), title="Smart Traffic Management System",css="""
/tmp/ipython-input-2377679525.py:127: DeprecationWarning: The 'css' parameter in the Blocks constructor will be removed in G
  with gr.Blocks(theme=gr.themes.Soft(), title="Smart Traffic Management System",css="""
It looks like you are running Gradio on a hosted Jupyter notebook, which requires `share=True`. Automatically setting `share

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://9e628aa0da8fb4af0a.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the
```

🚦 Smart Traffic Management System for Emission Reduction

Traffic Video Input

Drop Video Here

- or -

Click to Upload

Live Signal Status

Intersection Analytics

Lane	Vehicles	Idling Ris…	Wait (s)	Priority S…

TOTAL CO2 EMISSION SAVED

0.00 kg CO2

Next steps: Deploy to Cloud Run

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Smart Traffic Management System</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>

<div class="container">
  <h1> 🚦 Smart Traffic Management System</h1>
  <p class="subtitle">
    Real-time vehicle detection, emission-aware & waiting-time optimized signal control
  </p>

  <!-- EMBED GRADIO APP -->
  <div class="iframe-wrapper">
    <iframe
      src=" https://9e628aa0da8fb4af0a.gradio.live"
      allow="camera; microphone; clipboard-read; clipboard-write"
      loading="lazy">
    </iframe>
  </div>

  <footer>
    <p>AI-powered traffic signal optimization for carbon emission reduction</p>
  </footer>
</div>

<script src="script.js"></script>
```

```
</body>
</html>
```

Style.css

```
/* ================= RESET ================= */
* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}

html, body {
  height: 100%;
  overflow: hidden;
}

/* ================= BACKGROUND ================= */
body {
  font-family: "Poppins", Arial, sans-serif;
  background:
    linear-gradient(rgba(0, 0, 0, 0.65), rgba(0, 0, 0, 0.65)),
url("https://storage.googleapis.com/kaggle-datasets-images/4169612/7207127/c1ae22ec804f4d1481f35f45e39b2735/dataset-cover.jpg?t=2023-12-15-07-56-14");
  background-size: cover;
  background-position: center;
  background-attachment: fixed;

  display: flex;
  align-items: center;
  justify-content: center;
```

```
}

/* ================= CONTAINER ================= */
.container {
  width: min(96%, 1200px);
  height: min(92svh, 92vh); /* mobile-safe viewport */
  background: rgba(255, 255, 255, 0.14);
  backdrop-filter: blur(16px);
  -webkit-backdrop-filter: blur(16px);

  border-radius: 22px;
  padding: clamp(12px, 2vw, 24px);

  display: flex;
  flex-direction: column;
  gap: clamp(8px, 1.5vh, 14px);
}

/* ================= HEADER ================= */
h1 {
  color: #00ffae;
  text-align: center;
  font-size: clamp(1.4rem, 2.5vw, 2.2rem);
  line-height: 1.2;
}

.subtitle {
  text-align: center;
  color: #e0e0e0;
  font-size: clamp(0.8rem, 1.4vw, 1rem);
}
```

```
/* ================= IFRAME ZONE ================= */
.iframe-wrapper {
  flex: 1;
  min-height: 0; /* CRITICAL */
  border-radius: 16px;
  overflow: hidden;
  border: 2px solid rgba(0, 255, 174, 0.55);
  background: rgba(0,0,0,0.35);
}

/* ================= GRADIO ================= */
iframe {
  width: 100%;
  height: 100%;
  border: none;
  display: block;
}

/* ================= FOOTER ================= */
footer {
  text-align: center;
  font-size: clamp(0.7rem, 1vw, 0.85rem);
  color: #cccccc;
  opacity: 0.85;
}

/* ================= TABLET ================= */
@media (max-width: 1024px) {
  .container {
    width: 98%;
```

```
    border-radius: 18px;
  }
}

/* ================= MOBILE ================= */
@media (max-width: 600px) {
  body {
    background-attachment: scroll; /* better performance */
  }

  .container {
    height: 100svh;
    width: 100%;
    border-radius: 0;
    padding: 10px;
  }

  h1 {
    font-size: 1.3rem;
  }

  .subtitle {
    display: none; /* save space */
  }

  footer {
    display: none; /* no scrolling */
  }
}

/* ================= SMALL HEIGHT DEVICES ================= */
```

```
@media (max-height: 550px) {
  h1 {
    font-size: 1.2rem;
  }

  footer {
    display: none;
  }
}



console.log("Smart Traffic UI Loaded");
```