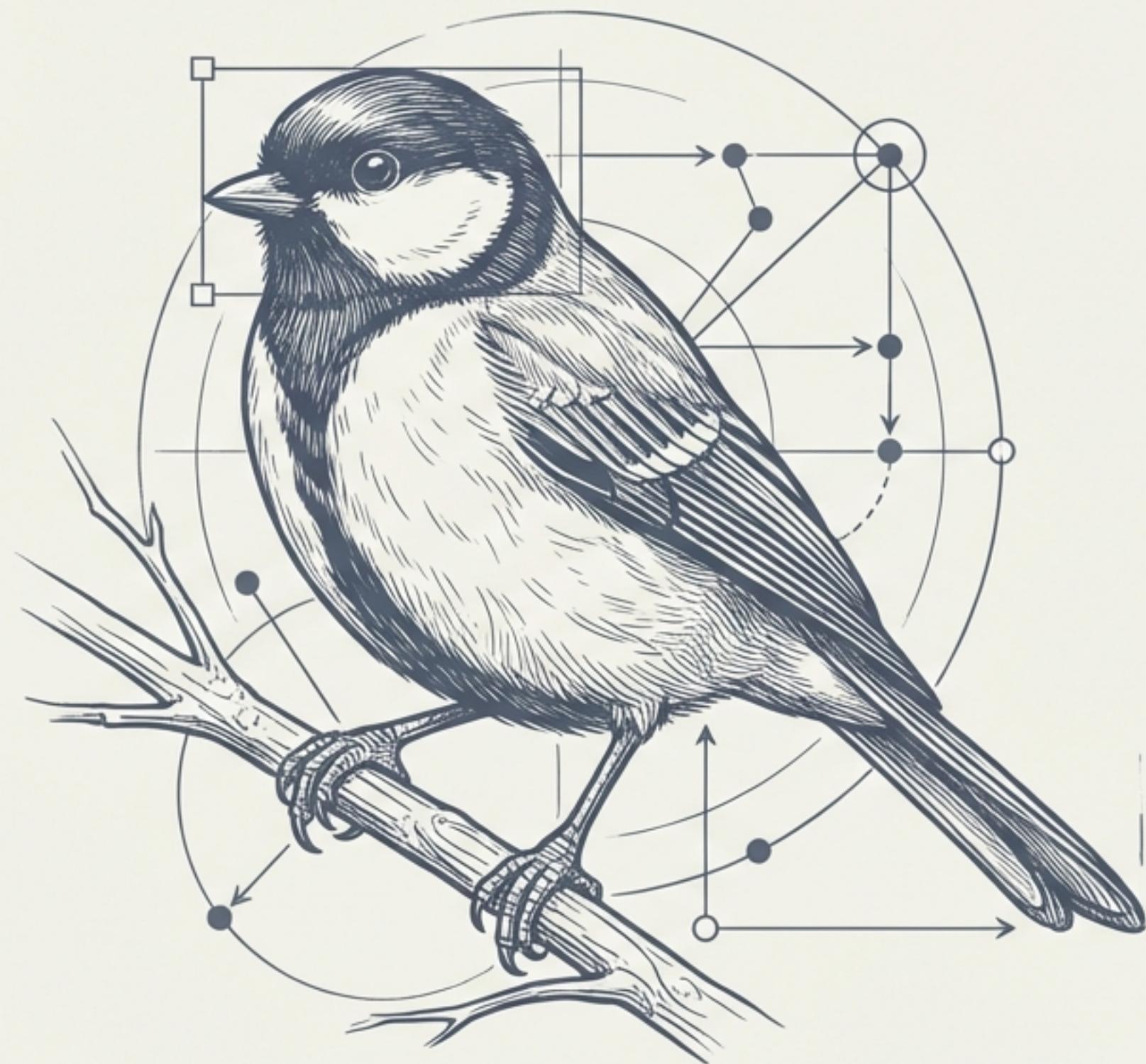
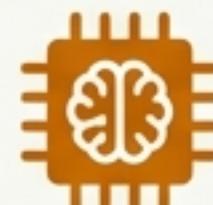


Vom Garten ins Web – Ein KI-gestütztes Ökosystem

Bauanleitung für den Birds AI Classifier:
Von der IP-Kamera bis zur Executable.



1. Input: Hardware-Setup und
Datenerfassung.

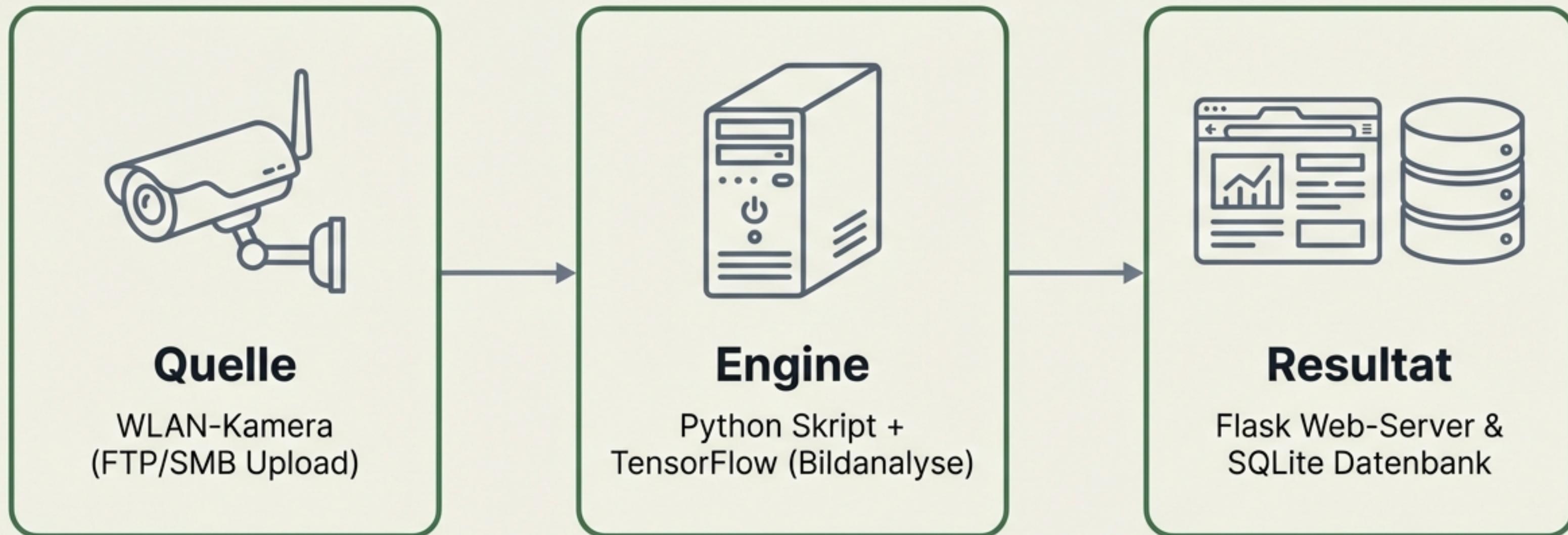


2. Process: KI-Analyse mit
TensorFlow.



3. Output: Web-Statistiken
und Deployment.

Die System-Architektur



Der Datenfluss ist linear: Das Bild entsteht draußen, wird drinnen verarbeitet und im Browser visualisiert.

Die Hardware-Basis: IP-Kamera Setup



Für ein echtes Vogelhaus im Garten ist eine wetterfeste WLAN / IP-Kamera die beste Lösung (z. B. Reolink, TP-Link Tapo).

Anders als bei USB-Webcams streamen wir nicht direkt. Wir nutzen die "Alarm-Funktion".

- Die Kamera erkennt Bewegung (Motion Detection).
- Ein Einzelbild wird automatisch via WLAN versendet.
- Zielort: Ein freigegebener Ordner auf dem PC.

Die Brücke zum PC: FTP & SMB



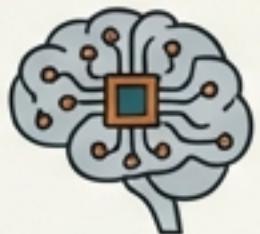
Im Kamera-Menü: Suche nach "Alarm-Einstellungen" oder 'FTP Upload'.
Auf dem PC: Erstelle einen Ordner und gib ihn im Netzwerk frei (Rechtsklick -> Eigenschaften -> Freigabe).

Key Insight: Das Python-Skript greift das Bild sofort ab, sobald es im Ordner landet. Der PC fungiert als lokaler Server.

Die Engine: Installation der Umgebung

```
pip install tensorflow flask pandas matplotlib pillow tk
```

Bevor der Code läuft, müssen die Abhängigkeiten installiert werden.



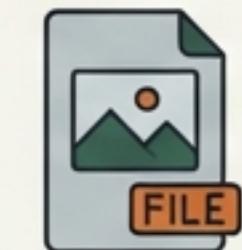
TensorFlow: Das Gehirn
(KI-Modell).



Flask: Der Web-Server für die
Anzeige.



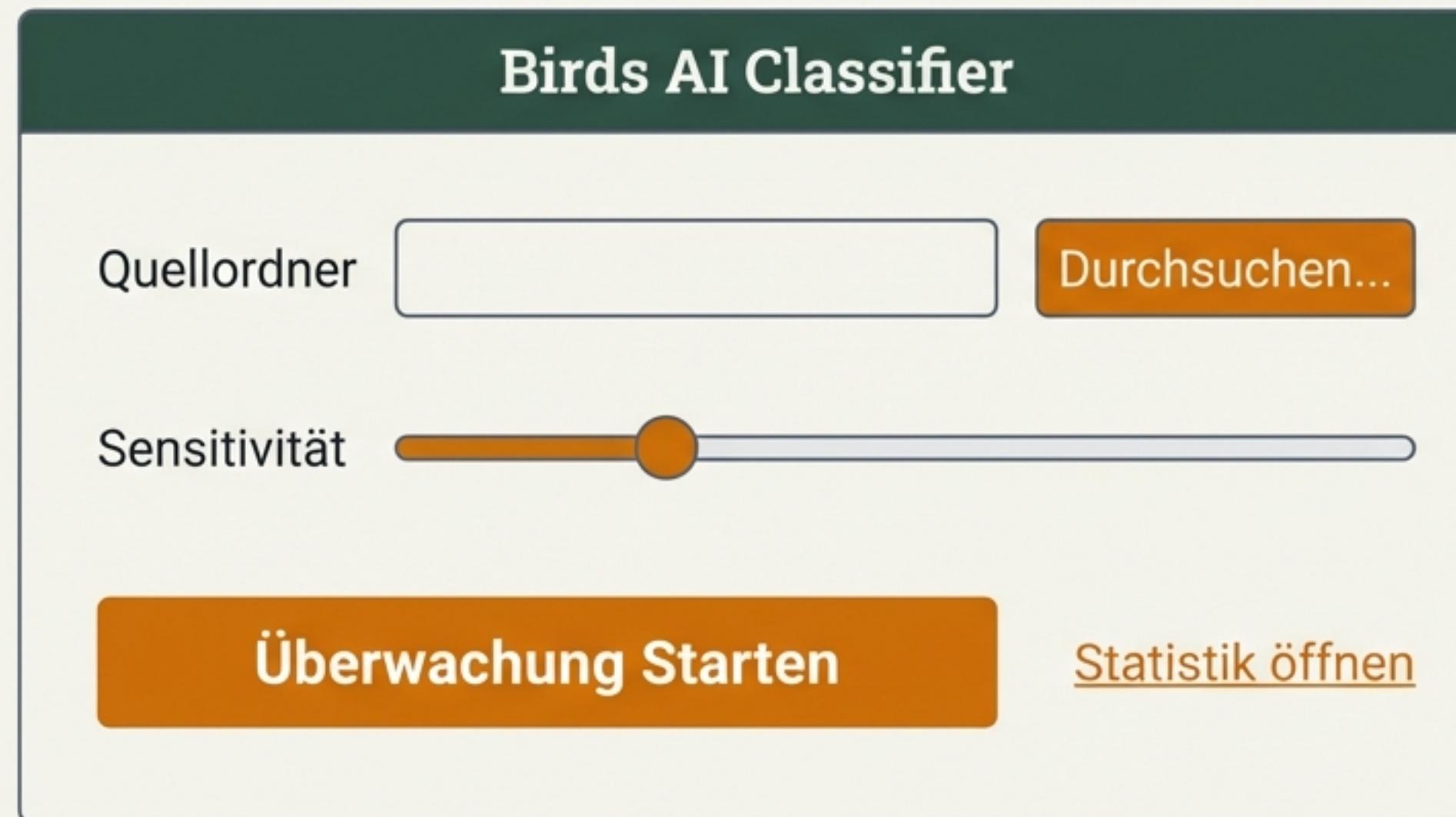
Pandas/Matplotlib:
Datenverarbeitung und Graphen.



Pillow: Bildbearbeitung.

Start der Überwachung

```
python birds_ai_classifier.py
```



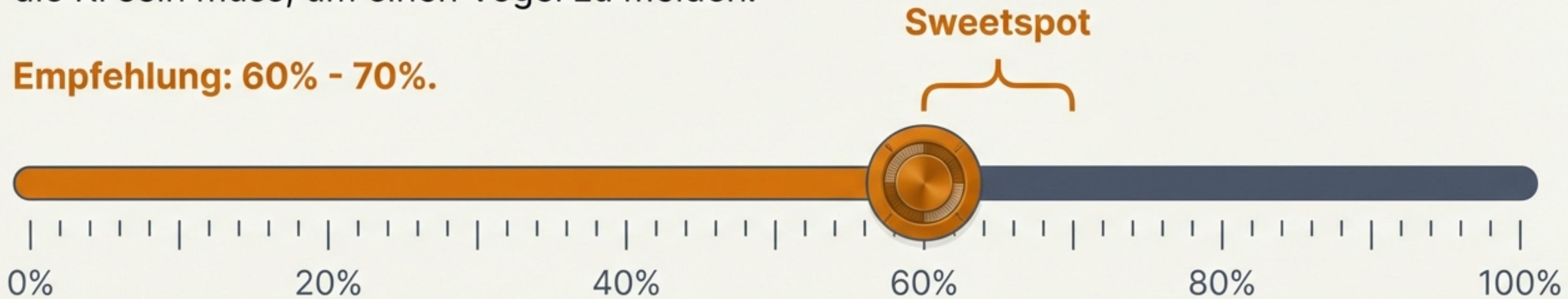
Nach dem Start öffnet sich das Kontrollzentrum.

- 1. Quellordner wählen:** Über "Durchsuchen..." den Netzwerkordner der Kamera verknüpfen.
- 2. Kalibrierung:** Den Schieberegler einstellen.
- 3. Aktivierung:** Button "Überwachung Starten" drücken.

Kalibrierung der KI-Sensitivität

Der Schieberegler bestimmt, wie 'sicher' sich die KI sein muss, um einen Vogel zu melden.

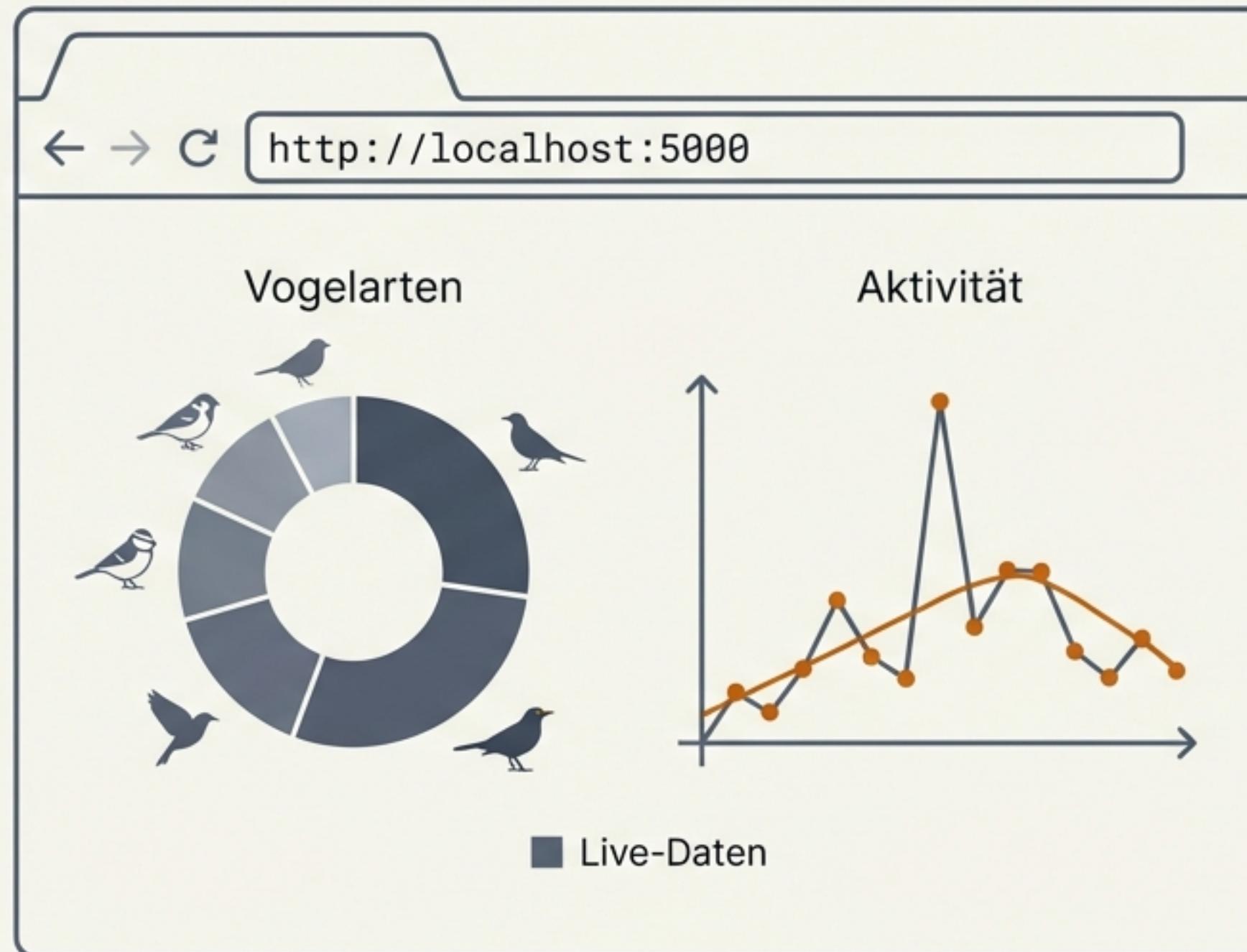
Empfehlung: 60% - 70%.



< 60%: Gefahr von Fehlalarmen
(z. B. ein Blatt wird als Vogel erkannt).

> 80%: Gefahr, dass echte Vögel
übersehen werden, wenn das Bild
unscharf ist.

Das Dashboard: Analyse im Browser



Sobald die Überwachung läuft, startet im Hintergrund ein lokaler Webserver.

- Klicke auf '**Statistik öffnen**' im GUI.
- Live-Visualisierung der erkannten Vogelarten.
- Zeitliche Verläufe und Häufigkeitsverteilung.

System-Wartung & Speicher-Management



Speicher-Warnung

Behalte die MB-Anzeige im Auge.

Alarm: Die Anzeige wird **rot**, sobald 500MB überschritten werden.



Reset

Button: '**Datenbank leeren**'.

Funktion: Löscht alle Statistiken (`birds_stats.db`), behält aber die gesammelten Bilder auf der Festplatte.

Deployment: Vom Skript zur .exe



Um das Programm ohne offenes Terminal oder auf anderen PCs laufen zu lassen, kompilieren wir es mit **PyInstaller**.

Schritt 1: Installation des Tools.

```
pip install pyinstaller
```

Dieser Schritt macht die Anwendung portabel und unabhängig von einer installierten Python-Umgebung auf dem Zielrechner.

Der Build-Prozess

```
pyinstaller --noconsole --onefile --name "birds_ai_classifier"  
birds_ai_classifier.py
```

Packt Python, TensorFlow, Pandas und das Skript in eine einzige Datei (statt eines Ordners).

Unterdrückt das schwarze Terminal-Fenster (für eine reine GUI-Erfahrung).

Legt den finalen Dateinamen fest.

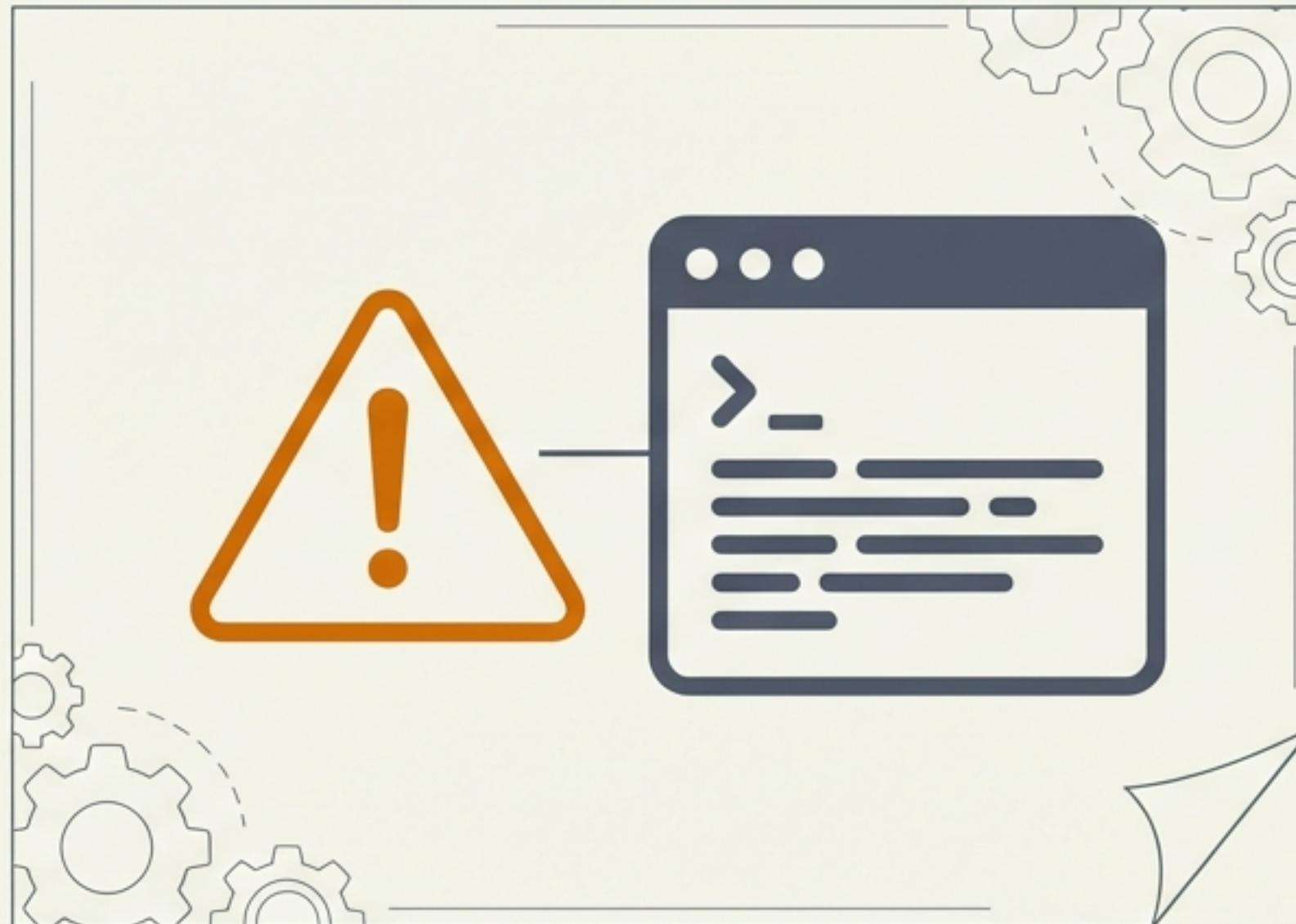
Geduld beim Kompilieren



PyInstaller analysiert und packt riesige Bibliotheken
(vor allem TensorFlow).

- Der Prozess kann **mehrere Minuten** dauern.
- Warnmeldungen im Terminal können meist ignoriert werden.
- **Ziel:** Warten auf die Meldung "completed successfully".
- **Fundort:** Die fertige Datei liegt im neuen Ordner `dist/`.

Troubleshooting & Debugging



Problem:

Die .exe schließt sich sofort wieder oder startet nicht?

Lösung:

Erstelle den Build neu – ohne den --noconsole Befehl.

```
pyinstaller --onefile --name  
"bird_ai_classifier_debug"  
birds_ai_classifier.py
```

Warum?

Jetzt bleibt das Terminal-Fenster offen und zeigt Fehlermeldungen an (z. B. fehlende Bibliotheken), die sonst unsichtbar wären.



Kommandostand (Cheat Sheet)

****Setup****

```
pip install tensorflow pillow  
flask matplotlib pandas tk
```

****Start****

```
python birds_ai_classifier.py
```

****Web****

```
http://localhost:5000/
```

```
delete birds_stats.db (zum Leeren)
```

****Build****

```
pyinstaller --noconsole  
--onefile ...
```