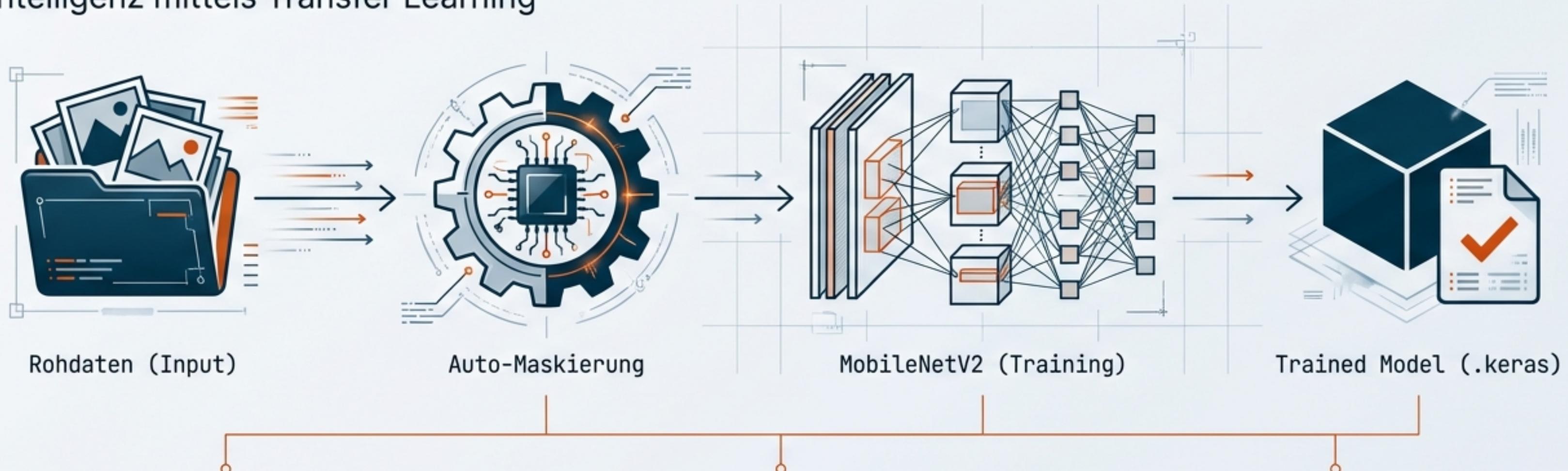


# Automatisierte Pipeline für Bildklassifizierung

Ein Workflow von Rohdaten zur künstlichen Intelligenz mittels Transfer Learning



## Zielsetzung

Training eines maßgeschneiderten KI-Modells zur Erkennung spezifischer Bildinhalte (z.B. Vogelarten).

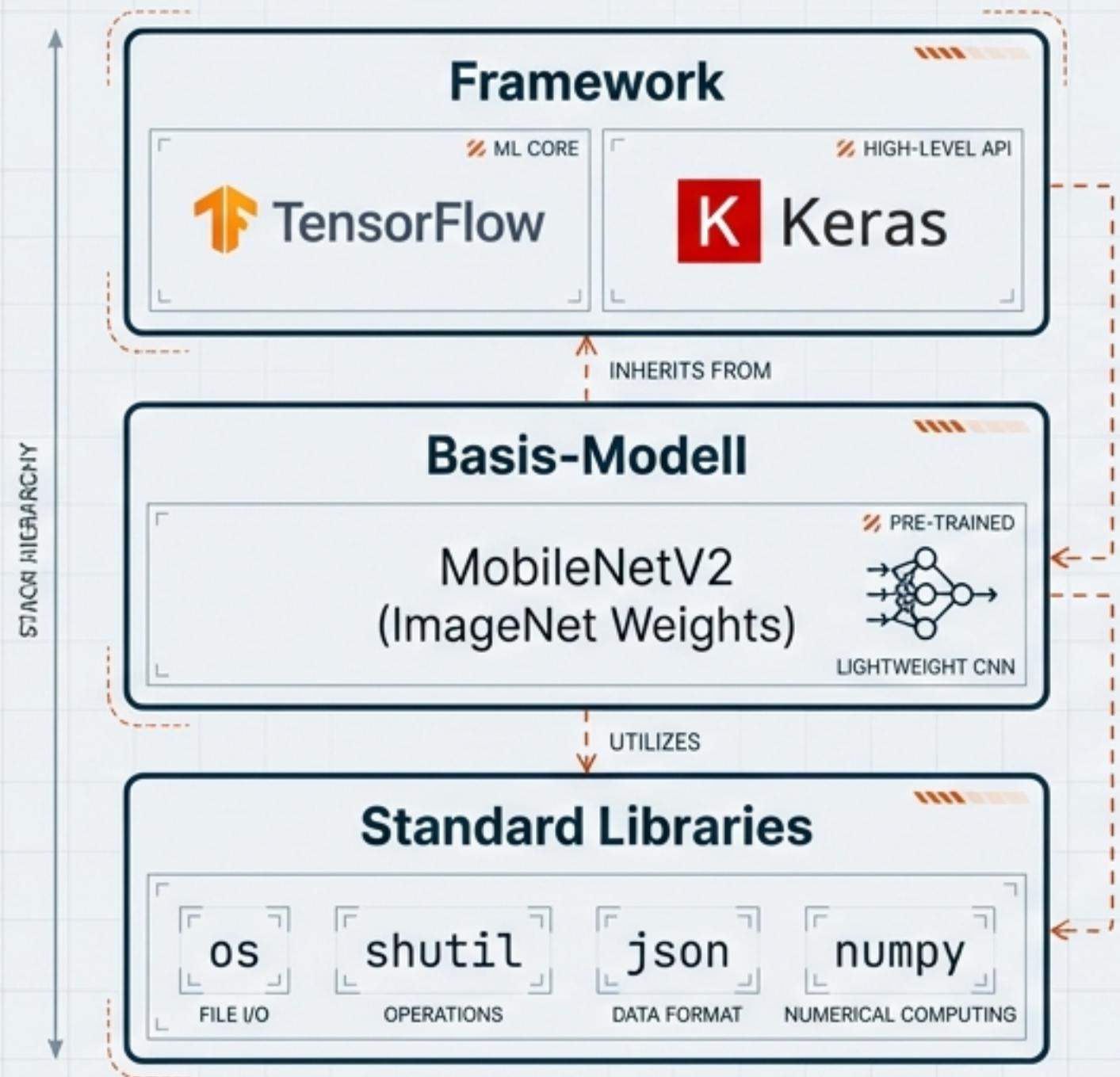
## Kerntechnologie

Nutzung von TensorFlow und MobileNetV2 für schnelle, hochpräzise Ergebnisse auf begrenzten Datensätzen.

## Prozess

Vollautomatische Vorverarbeitung, Datenanreicherung (Augmentation) und Training in einem einzigen Durchlauf.

# Systemarchitektur & Abhängigkeiten

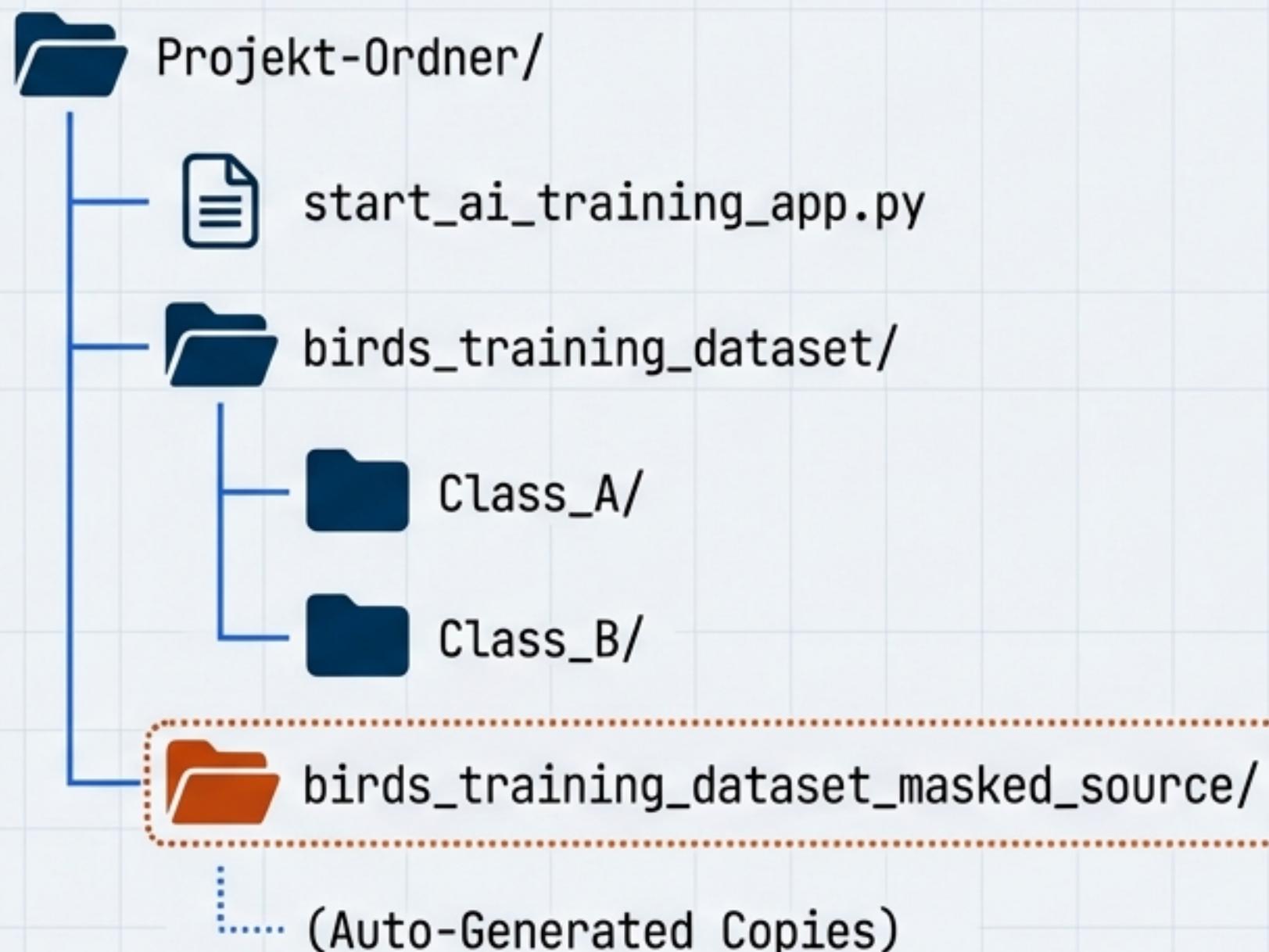


Das Skript orchestriert Datei-Operationen und neuronale Netzwerk-Operationen in einer nahtlosen Sequenz.

**⚠️ Kritische Installation**

Der Befehl `pip install scipy` ist **zwingend** erforderlich, damit die Bildtransformationen im Hintergrund berechnet werden können. Ohne diese Bibliothek schlägt die Datenanreicherung fehl.

# Projektstruktur & Datenorganisation



## Input (Benutzerdefiniert)

**Ordner:** `birds\_training\_dataset`

Hier liegen die Rohbilder, sortiert nach Klassen in Unterordnern. Der Ordnername definiert das Label.

## Output (Systemgeneriert)

**Ordner:** `birds\_training\_dataset\_masked\_source`

Dieser Ordner wird vom Skript automatisch erstellt und bei jedem Neustart bereinigt.

## Wichtiger Hinweis

Der generierte Ordner dient als bereinigte Datenquelle für das Training. Löschen Sie diesen Ordner nicht während der Laufzeit.

# Konfiguration & Parameter

## SYSTEM CONTROLS

**EPOCHS**  
**10**

Anzahl der Durchläufe durch den gesamten Datensatz.

## SYSTEM CONTROLS

**BATCH\_SIZE**  
**8**

Anzahl der Bilder, die pro Schritt verarbeitet werden.

**IMG\_SIZE** (224, 224)

Standardauflösung für MobileNetV2 Input-Layer.

## DATEI-PFADE KONFIGURATION

```
DATASET_NAME = `birds_training_dataset`  
MODEL_OUTPUT = `my_birds_modell.keras`  
LABEL_OUTPUT = `model_labels.json`
```

JETBRAINS MONO 'CY

# Feature Spotlight: Automatisierte Maskierung



Original (Input)

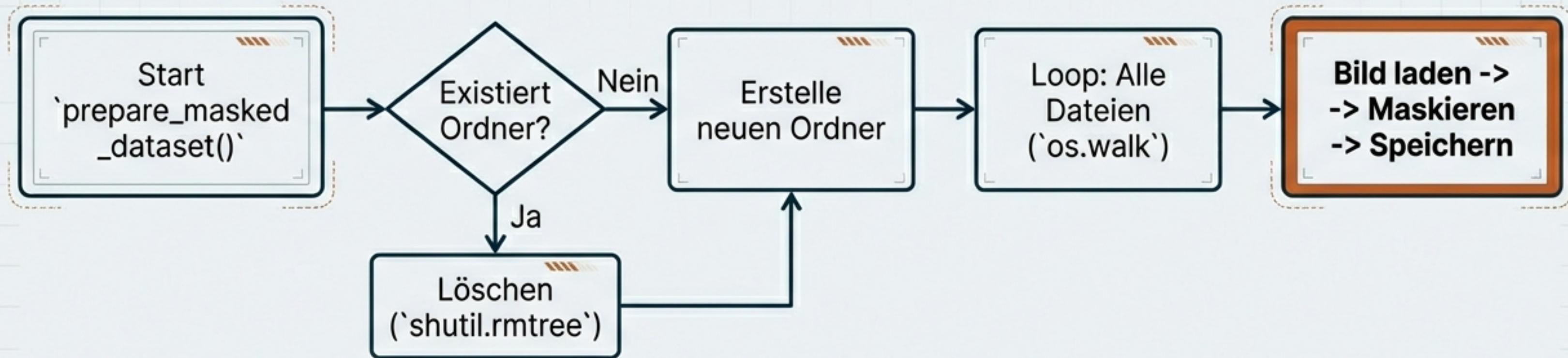


Maskiert (Training Input)

- **Funktion**  
`apply\_mask\_to\_array`  
Das Skript schwärzt gezielt Bildbereiche, um Störfaktoren wie Datumsstempel oder Copyright-Texte zu entfernen. Das Modell lernt somit nur das Motiv, nicht den Randtext.

- `MASK\_BOTTOM = 10 px`

# Der Vorverarbeitungs-Workflow



## Schritt 1: Reset.

Der Zielordner wird bei jedem Start komplett entfernt und neu angelegt, um Datenkonsistenz zu sichern.

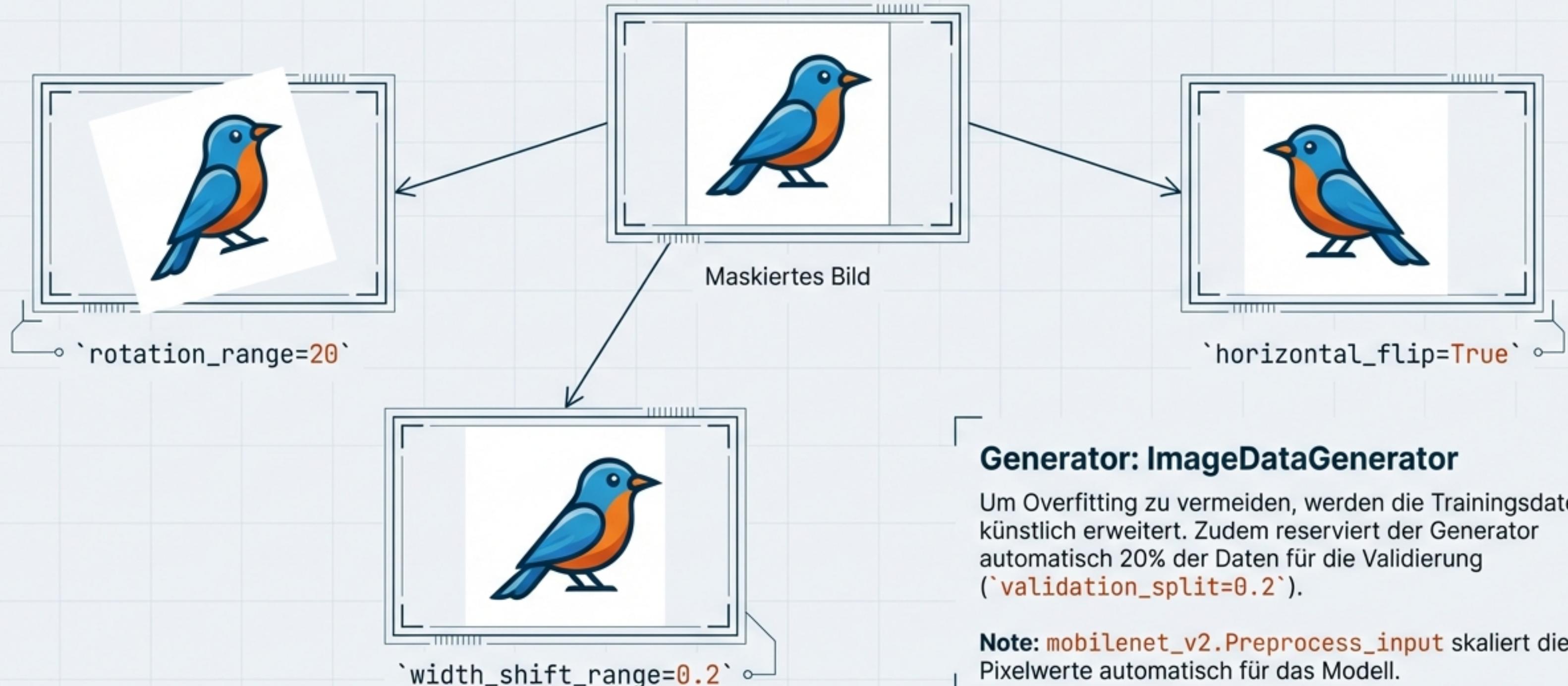
## Schritt 2: Transformation.

Jedes Bild (.png, .jpg, .jpeg) wird geladen, oben/unten geschwärzt und in eine gespiegelte Verzeichnisstruktur gespeichert.

## Sicherheit.

Das Training erfolgt ausschließlich auf diesen kopierten Daten. Die Originale bleiben unberührt.

# Data Augmentation Strategie



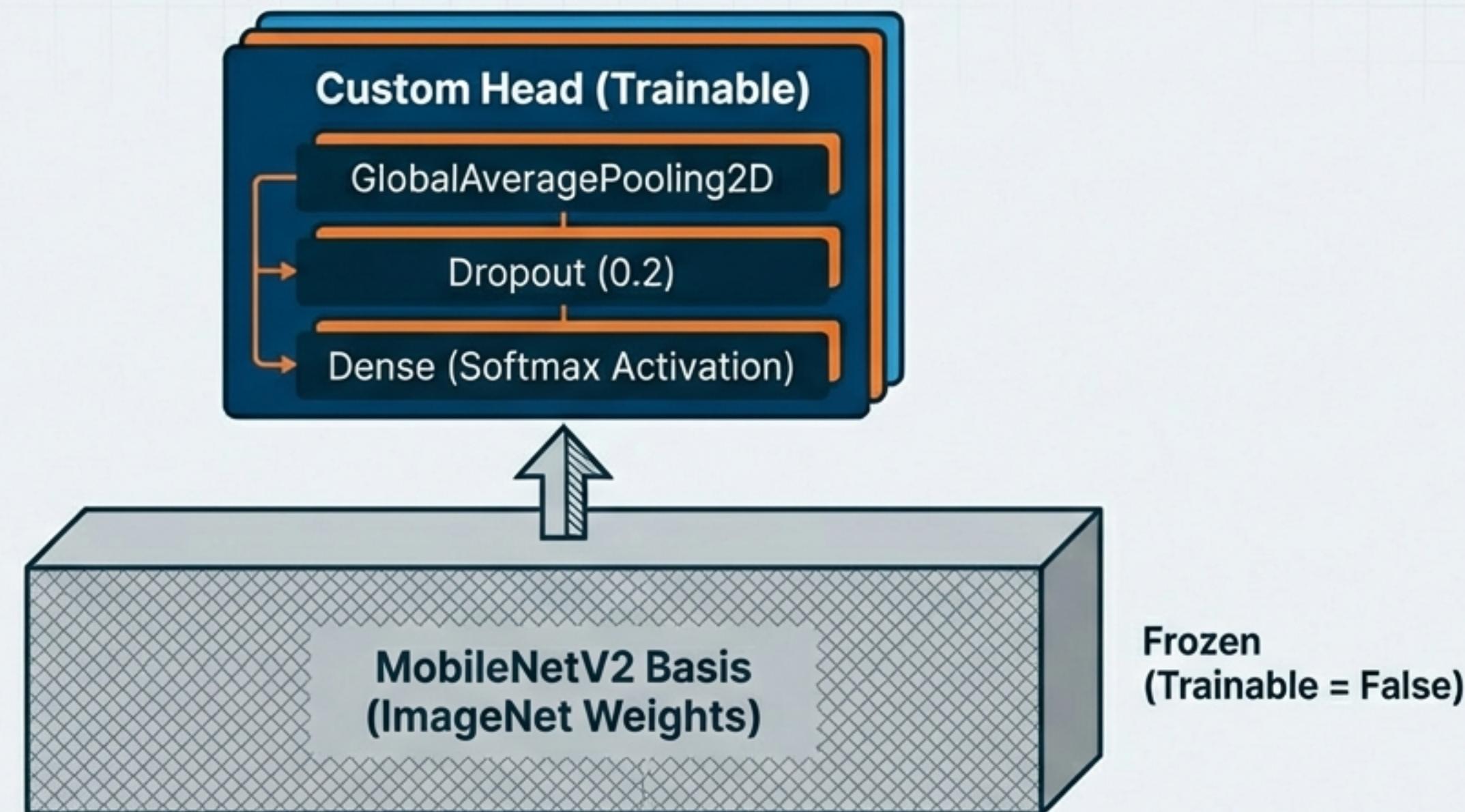
## Generator: ImageDataGenerator

Um Overfitting zu vermeiden, werden die Trainingsdaten künstlich erweitert. Zudem reserviert der Generator automatisch 20% der Daten für die Validierung (`validation_split=0.2`).

**Note:** `mobilenet_v2.Preprocess_input` skaliert die Pixelwerte automatisch für das Modell.



# Modell-Architektur: MobileNetV2



Enthält vortrainiertes Wissen aus Millionen von Bildern.  
Wir nutzen Transfer Learning: Die Basis ist 'eingefroren', um Rechenzeit zu sparen  
und Robustheit zu garantieren. Trainiert wird nur der neue 'Kopf',  
der die spezifischen Klassenentscheidungen trifft.

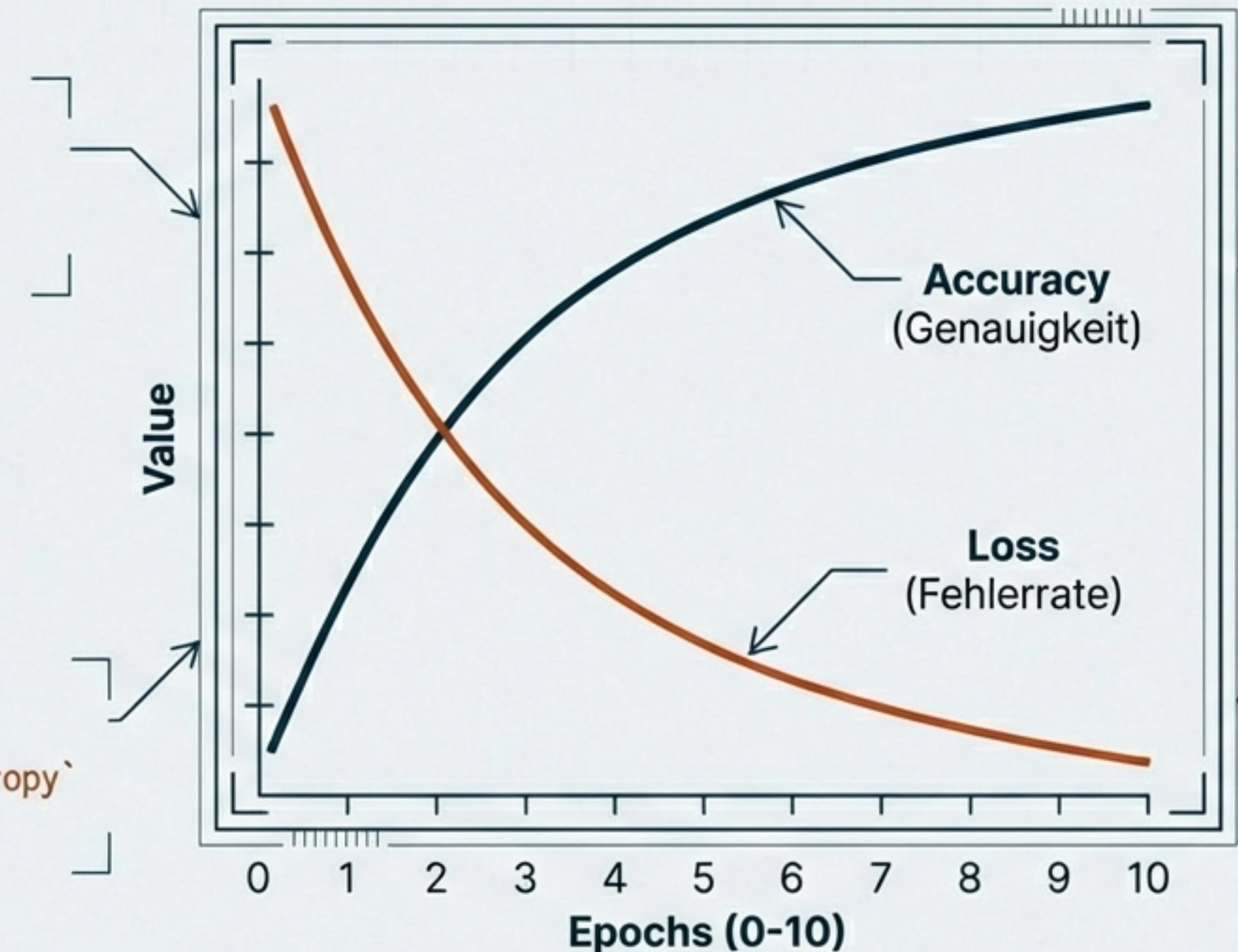
# Der Trainingsprozess

## Kompilierung

Optimizer: `'Adam'`  
(Learning Rate 0.001)

## Loss Function

`'categorical_crossentropy'`  
(Multi-Class)



## Execution

`'model.fit()'` führt das Training über 10 Epochen durch.

## Safety

Falls `'train_generator.samples == 0'`, bricht das Skript sicher ab.

# Qualitätskontrolle & Validierung

## Manuelle Prüfung erforderlich

- Starten Sie das Training.
- Warten Sie auf Nachricht:  
"Erstelle maskierten Datensatz..."
- Öffnen Sie Ordner:  
`birds_training_dataset_masked_source`
- Prüfen Sie die Maskierung visuell.



# Output & Deployment-Artefakte



`my\_birds\_modell.keras`

Das Gehirn. Enthält die trainierten Gewichte und die Netzwerk-Architektur.

`model\_labels.json`

Der Übersetzer. Ordnet numerische Vorhersagen den Klassennamen zu (z.B. 0: 'Amsel').

Deployment



**Wichtig:** Beide Dateien müssen zusammen in den Ordner des Classifier-Skripts verschoben werden, damit die App funktioniert.

# Zusammenfassung & Best Practices



## Daten-Check

Ordner `birds\_training\_dataset` muss existieren. Nur .png, .jpg, .jpeg Formate.



## Masking

Visuelle Prüfung der schwarzen Balken ist essenziell für sauberes Training.



## Labels

Klassennamen ergeben sich automatisch aus den Unterordner-Namen.



## Dependencies

Stellen Sie sicher, dass `scipy` installiert ist (pip install scipy).

Fazit: Eine robuste, in sich geschlossene Lösung für lokale KI-Entwicklung und schnelles Prototyping.