FOOD101 GradCAM & XRAI

Tom Schütt, Leonard Michalsky und Lars Obrath

Food101

Datensatz

101.000 Bilder

101 Essenskategorien

- Für jede Kategorie:
  - 750 Trainingsbilder
  - 250 Testbilder

Ursprüngliche Herkunft: foodspotting.com

Veröffentlicht in Paper:
Bossard, L. et al (2014). Food-101 – Mining Discriminative Components with Random Forests.

Eigenschaften der Bilder

Skaliert auf 512 Pixel Seitenlänge	Noise in Trainingsdaten zur Erhöhung der Robustheit
Bereinigte Testdaten	Ähnliche Essenskategorien vorhanden: *Paella, Risotto* und *Omlette*

Noise in den Daten - Beispiel Ravioli

Auszug aus Kategorien

Apple pie	Beignets	Ceviche	Chocolate mousse	Croque madame
Baby back ribs	Bibimbap	Cheesecake	Churros	Cup cakes
Baklava	Breakfast burrito	Cheese plate	Clamchowder	Deviled eggs
Beef carpaccio	Bruschetta	Chicken curry	Club sandwich	Donuts
Beef tartare	Caesar salad	Chicken wings	Crab cakes	Dumplings
Beet salad	Carrot cake	Chocolate cake	Creme brulee	Edamame

Grad-CAM

Gradient-weighted Class Activation Mapping

- Visualisierung und Erklärung von neuronalen Netzen
- Hervorhebung relevanter Bildbereiche
- Darstellung wichtiger Bereiche als “Heatmap”

- Erkennung von Bias im Datensatz
- Verständnis von falschen Vorhersagen

Funktionsweise

Annahmen

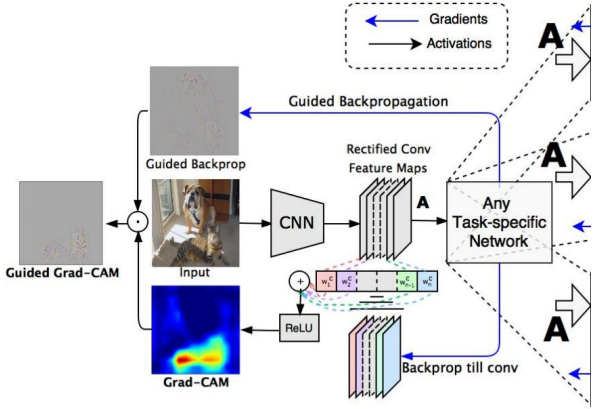
- tiefere Convolutional Layer in einem CNN haben high-level Informationen über visuelle Zusammenhänge
- Verlust dieser Informationen in vollvermaschten Schichten

=> Arbeit auf letzter Convolutional Schicht

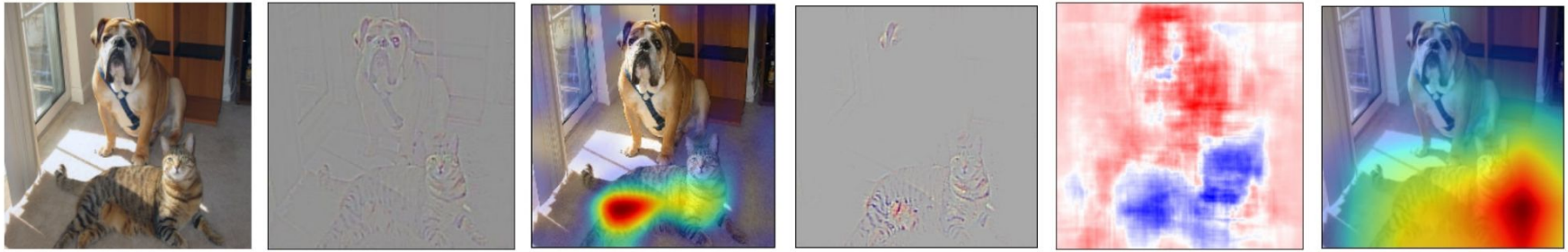
(kann aber für alle Schichten benutzt werden)

Funktionsweise

1. Eingabebild wird vorwärts propagiert
2. Erhalt des Scores für die vorhergesagte Kategorie
3. Gradienten aller anderen Kategorien werden auf 0 gesetzt
4. Gradienten der vorhergesagten Kategorie auf 1 gesetzt
5. Signal wird **zurück propagiert** zur Erstellung einer Feature Map
6. **Erstellung der Heatmap**
7. (Guided Grad-CAM: Multiplikation mit guided backpropagation)
8. Heatmap wird **über Originalbild** gelegt



https://arxiv.org/pdf/1610.02391



(a) Original Image (b) Guided Backprop 'Cat' (c) Grad-CAM 'Cat' (d)Guided Grad-CAM 'Cat' (e) Occlusion map 'Cat' (f) ResNet Grad-CAM 'Cat'

(g) Original Image (h) Guided Backprop 'Dog' (i) Grad-CAM 'Dog' (j)Guided Grad-CAM 'Dog' (k) Occlusion map 'Dog' (l)ResNet Grad-CAM 'Dog'

Vor- und Nachteile

- Leicht verständlich, da Visualisierungen mit menschlicher Aufmerksamkeit korreliert
- Liefert, bei der Lokalisierung von Bildobjekten, gute Ergebnisse

- Visualisierungen oft zu grob für kleine Bildobjekte
- Funktioniert nur bei Gradient based Verfahren

XRAI Verfahren

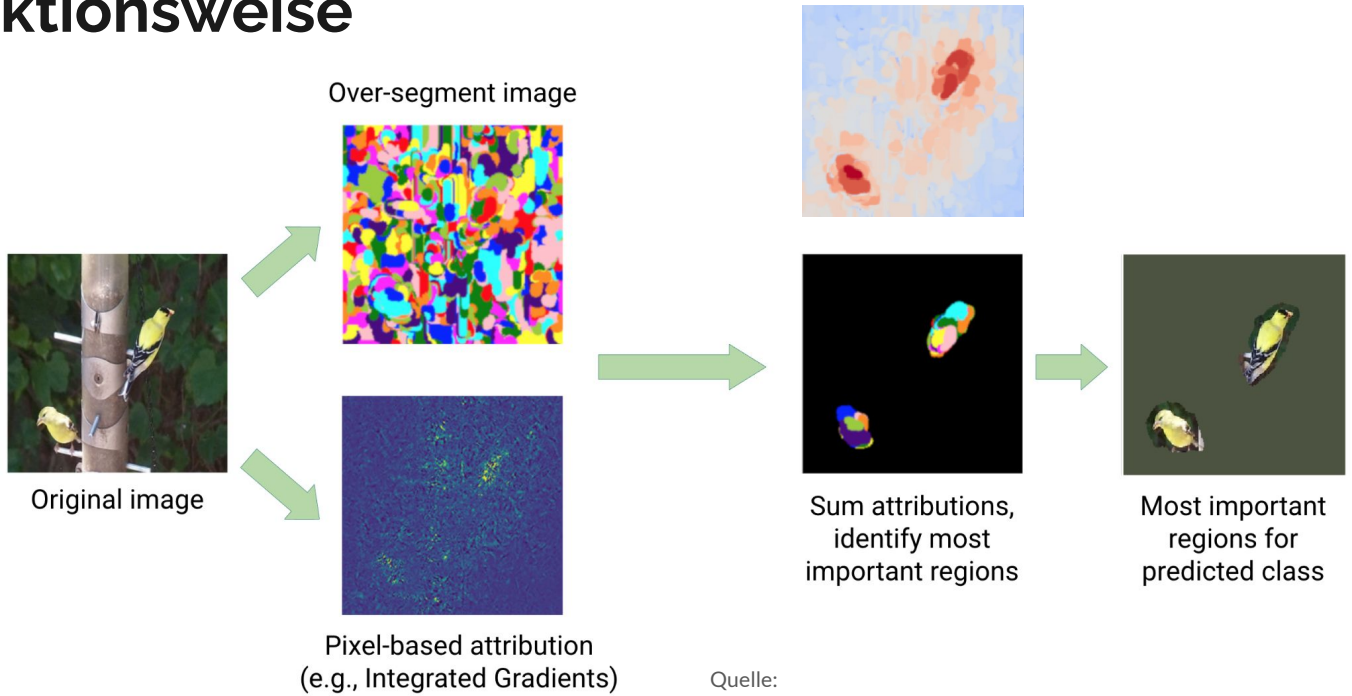
XRAI Verfahren

- Erklärverfahren für DNN Modelle, hauptsächlich für Bilder
- Visualisierung der Merkmale, die am meisten zu einer Vorhersage beigetragen haben
- Basiert auf den **Integrated Gradients** Verfahren
    - **Problem**: Pixel-basierte Darstellung kann schwierig zu lesen und zu interpretieren sein
- **Lösung:** Identifizierung auffälliger Regionen
- Extrahiert interessante Regionen indem das Bild in Segmente aufgeteilt wird

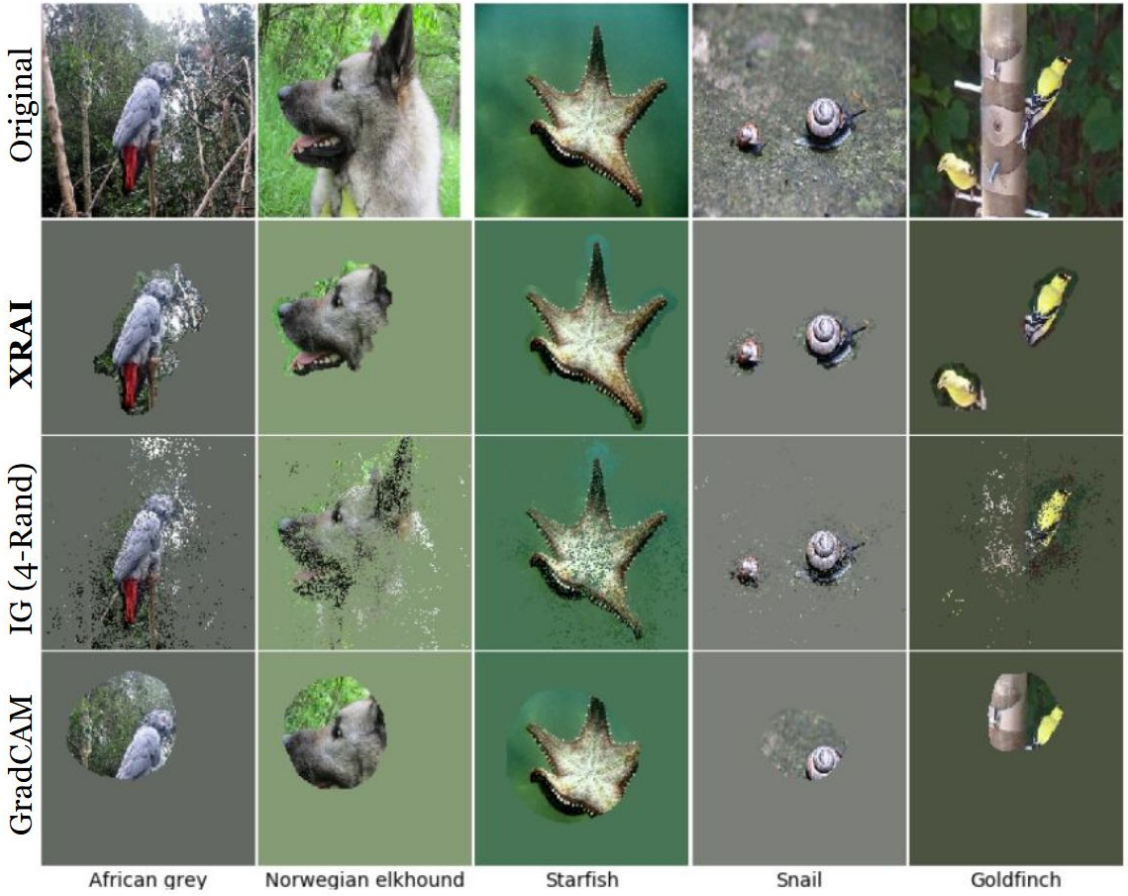
Funktionsweise

1. Anwendung von **Integrated Gradients** für eine Pixel-basierte Zuordnung
2. **Unterteilung des Bildes** in ähnliche Regionen
    - Mittels **Felzenszwalb Algorithmus:** Graph-basiertes Verfahren
    - Sehr recheneffizient; Zeitkomplexität *O(n log n)*
3. **Bewertung der Wichtigkeit** der einzelnen Regionen
    - Aufsummierung der Pixel-basierten Werte aus 1. Schritt für jede Region
    - Bildung einer Rangfolge basierend darauf, welche Region am meisten zum Ergebnis beigetragen hat

Funktionsweise



Quelle:
https://pair-code.github.io/saliency/docs/ICCV_XRAI_Poster.pdf



Quelle:
https://pair-code.github.io/saliency/docs/ICCV_XRAI_Poster.pdf

Vorteile und Nachteile

Vorteile

- Leicht verständliche und interpretierbare Visualisierung
- Funktioniert gut für natürliche Bilder (z.B. Tiere, Gegenstände)

Nachteile

- Weniger Details, keine pixelgenaue Darstellung
- Funktioniert nicht so gut für Bilder mit niedrigem Kontrast (z.B. Röntgenbilder)

Implementierung

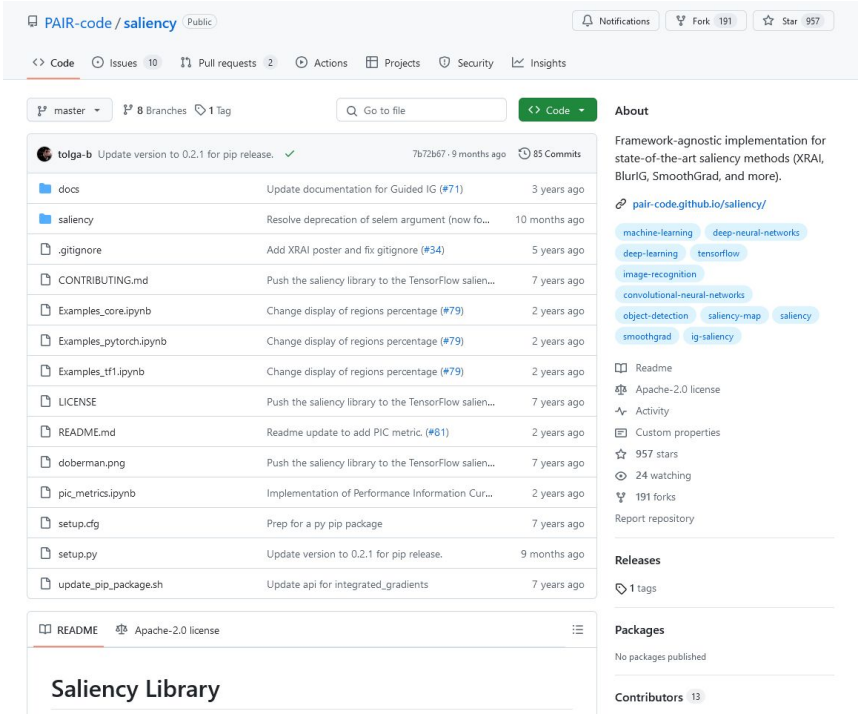
Offizielle Python Bibliothek: [Saliency]

- Framework-agnostisch

```
call_model_args = {class_idx_str: prediction_class}

# Construct the saliency object.
xrai_object = saliency.XRAI()

# Compute XRAI attributions with default parameters
xrai_attributions = xrai_object.GetMask(x,
                                        call_model_function,
                                        call_model_args)
```



Code-Demonstration

Ende

Habt ihr Fragen?