Integrated Grad-CAM: Sensitivity-Aware Visual Explanation of Deep Convolutional Networks via Integrated Gradient-Based Scoring

Sam Sattarzadeh, Mahesh Sudhakar, Konstantinos N. Plataniotis, Jongseong Jang, Yeonjeong Jeong, Hyunwoo Kim

University of Toronto, LG AI research

Introduction

- Explainable AI (XAI): Understanding Convolutional Neural Networks (CNNs) is crucial for high-impact and high-risk applications in computer vision.
- Our aim: Visual Explainability: Visualizing the input features responsible for CNN prediction.

![Diagram of CNN with integrated Grad-CAM](image)

Background

- Methods for visual explainability:
  - Backpropagation-based methods: Computing the gradient of CNNs output to the input features or hidden neurons.
  - CAM-based methods: Visualizing the features extracted in a single layer of the CNNs.
  - Perturbation-based methods: Probing the model’s behavior using perturbed copies of the input image.

![Diagram of CAM method](image)

Contributions

- Our proposed approach: Integrated Grad-CAM
- Addressing the limitations of backpropagation in explaining non-linear models.
- Solving the gradient limitations by employing gradients.

Integrated Grad-CAM

Novelty:
Scoring the feature maps in the last convolutional layer of CNNs based on Integrated Average Gradient values, instead of “Average Gradient” values utilized in Grad-CAM.

Intuition: Sensitivity axiom: (Sundararajan et al., '17) For each pair of input and baseline differing only in one feature, an attribution method should highlight this difference by assigning different values corresponding to that feature.

Idea: Calculating the integral of gradient values in a path that links a certain baseline to the input.

Path Integral

Defining a path linking a baseline \( l' \) and an input \( l \):

Path equation:

\[
\gamma(t) = l' + f(\alpha) \times (l - l')
\]

- \( f(\alpha) : \mathbb{R} \rightarrow \mathbb{R} \): A differentiable and monotonically increasing function.
- \( 0 \leq \alpha \leq 1 \): \( f(0) = 0 \) and \( f(1) = 1 \).

For each pair of functions \( h, g \):

\[
\text{Path}_{G_h}(g) = \int_0^1 \frac{dh(\alpha)}{\partial \alpha} [g(\gamma(t)) - g(l')] \, d\alpha
\]

Methodology

Feature maps derived from a convolutional layer \( l \): \( \{A^1(l), A^2(l), ..., A^K(l)\} \)

Grad-CAM formulation:

\[
M_{\text{Grad-CAM}} = \text{ReLU} \left( \sum_{k=1}^{N} \sum_{i \in j} \frac{\partial y}{\partial A^k(i)} A^k(i) \right)
\]

Our method: Replacing gradient terms with integrated gradient terms:

\[
M_{\text{ICG-CAM}} = \int_0^1 \frac{1}{N} \text{ReLU} \left( \sum_{k=1}^{N} \sum_{i \in j} \frac{\partial y}{\partial A^k(i)} A^k(i) - A^k(l') \right) \, d\alpha
\]

Limitation: The equation above is hard to implement.

Implementation

For simplicity, we assume the path between \( l' \) and \( l \) to be linear. Then, we approximate the equation (3) using Reimann’s approximation.

\[
M' \approx \sum_{i \in j} \text{ReLU} \left( \sum_{k=1}^{m} \sum_{l=1}^{L} \frac{\partial y}{\partial A^k(i)} A^k(i) - A^k(l) \right) \]

The number of sampled points along the linear path: \( m \) (set to 20 by default.)

Experiments

Dataset: PASCAL VOC 2007
- Purpose: Multi-label image classification, Object Detection.
- Containing 4963 test images in 20 classes, Bounding boxes provided.
- A VGG-16 model and a ResNet-50 model trained on this dataset are utilized.

![Image of PASCAL VOC 2007 dataset](image)

Quantitative Evaluation

Evaluation metrics:
- Ground truth-based like Energy-based Pointing Game (EBPG), Mean Intersection-over-Union (mIoU) and Bounding Box (bbox) are used to verify the meaningfulness of explanation methods, and their ability in feature visualization.
- Model truth-based like Drop and Increase rate are employed to justify the faithfulness and validity of the generated explanations from the model’s perspective.

<table>
<thead>
<tr>
<th>Model</th>
<th>Metric</th>
<th>Grad-CAM</th>
<th>Grad-CAM++</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG16</td>
<td>EBPG</td>
<td>55.44</td>
<td>46.29</td>
</tr>
<tr>
<td></td>
<td>Bbox</td>
<td>51.7</td>
<td>55.59</td>
</tr>
<tr>
<td>Drop</td>
<td>49.47</td>
<td>60.63</td>
<td>47.96</td>
</tr>
<tr>
<td>Increase</td>
<td>31.08</td>
<td>23.89</td>
<td>34.49</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>EBPG</td>
<td>60.08</td>
<td>47.78</td>
</tr>
<tr>
<td></td>
<td>Bbox</td>
<td>60.25</td>
<td>58.66</td>
</tr>
<tr>
<td>Drop</td>
<td>35.90</td>
<td>41.77</td>
<td>34.49</td>
</tr>
<tr>
<td>Increase</td>
<td>36.59</td>
<td>32.15</td>
<td>36.84</td>
</tr>
</tbody>
</table>

Conclusion

Integrated Grad-CAM Takeaways:
- Circumvented the underestimations in Grad-CAM and Grad-CAM++.
- Addressed the issues caused by backpropagation in the methods above.
- Though slower than the conventional methods, offers an acceptable run-time to be used in real-world applications.
- The takeaways above are verified through extensive experiments on the PASCAL VOC 2007 dataset.

References