Convolutional Networks Explainer

Reading Group Spring 2019

Zhiping (Patricia) Xiao

University of California, Los Angeles

May 14, 2019

Outline

Background Introduction

Why explaining CNN?

- CNN is powerful and popular.
- Interpretability Matters. Like *"why they predict what they predict."*
  - identify the failure modes
  - establish appropriate trust and confidence in users
  - teach a human how to make better decisions
- There's a natural trade-off between accuracy and simplicity or interpretability.
- Visualization helps better understanding the properties.
- Understanding CNN is GNN's foundation.

The Basic CNN model

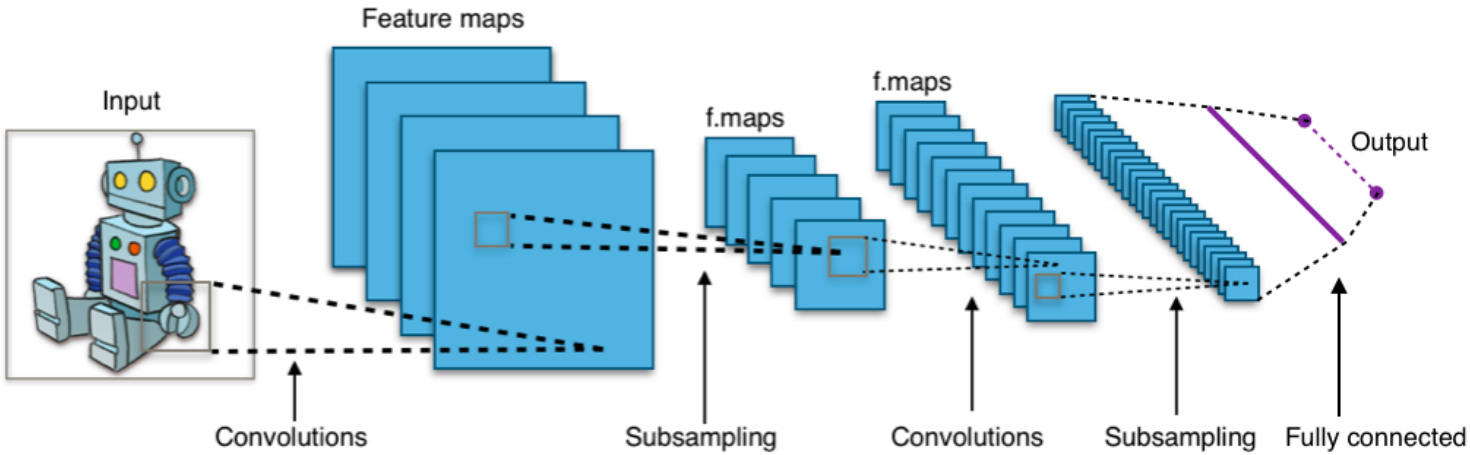


Figure: Convolutional layers (with nonlinear activation), pooling layers (subsampling), finally fully connected (FC) layer whose input is all the output from the last layer and whose output is an N-dimensional tensor (N is # classes). Image from `https://en.wikipedia.org/wiki/Convolutional_neural_network`.

The Basic CNN model

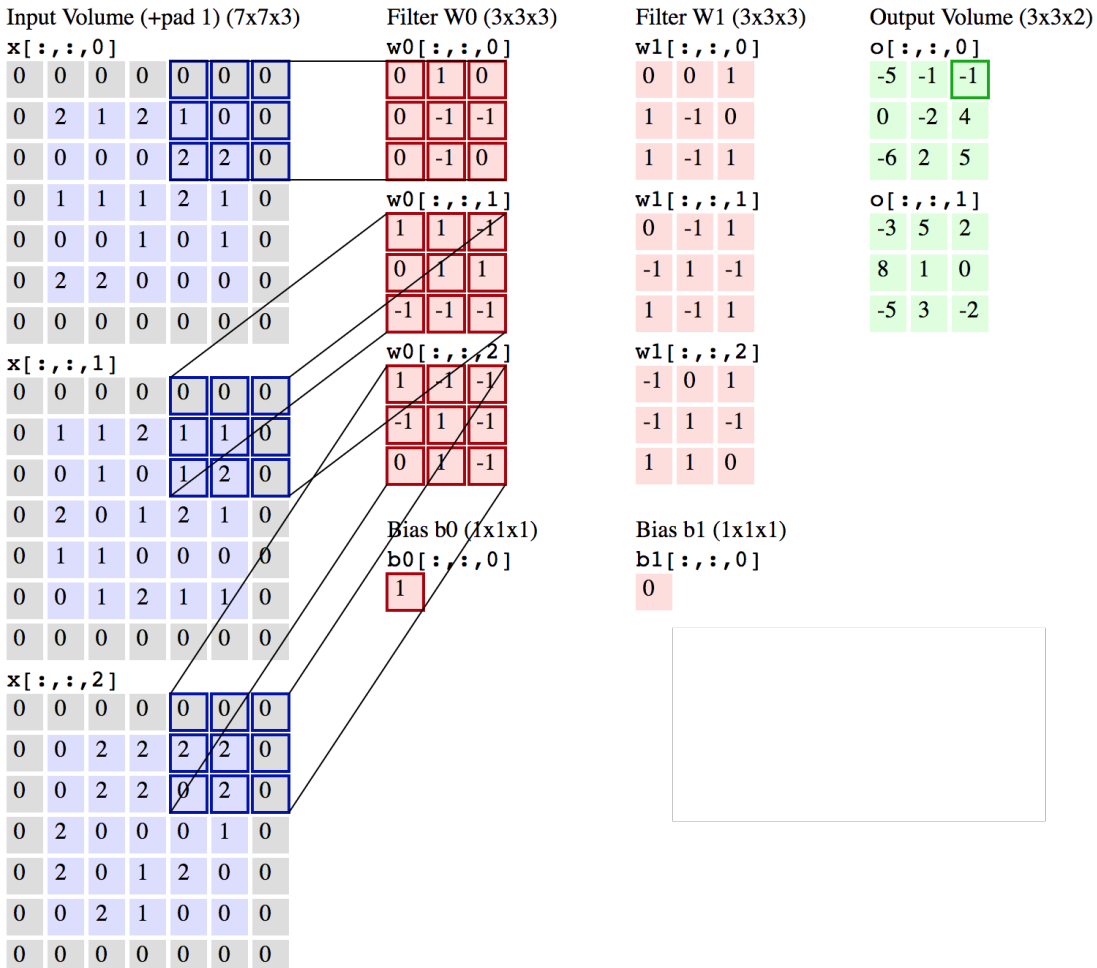


Figure: Convolutional Layer.[1] Output Volume = *feature maps*.

[1] http://cs231n.github.io/convolutional-networks/.

The Basic CNN model

Convolutional Neural Networks for *classification*:

- perform convolution in the lower layers
- the *feature maps*[2] of the last convolutional layer are vectorized
- the vectorized feature map is fed into fully connected layers
- followed by a softmax logistic regression layer

For more introduction, please visit
`http://cs231n.github.io/convolutional-networks/`.

[2] features and their locations, convolutional layer outputs

Fully-Connected-Layer: Limitations

- Overfitting: too many parameters in the model
  - Solution 1: Dropout
- Loosing localization ability
  - Solution 2: replacing the FC layer with something else (e.g. FCN (fully-convolution), **GAP (global average pooling)**). These methods also solve the overfitting problem.

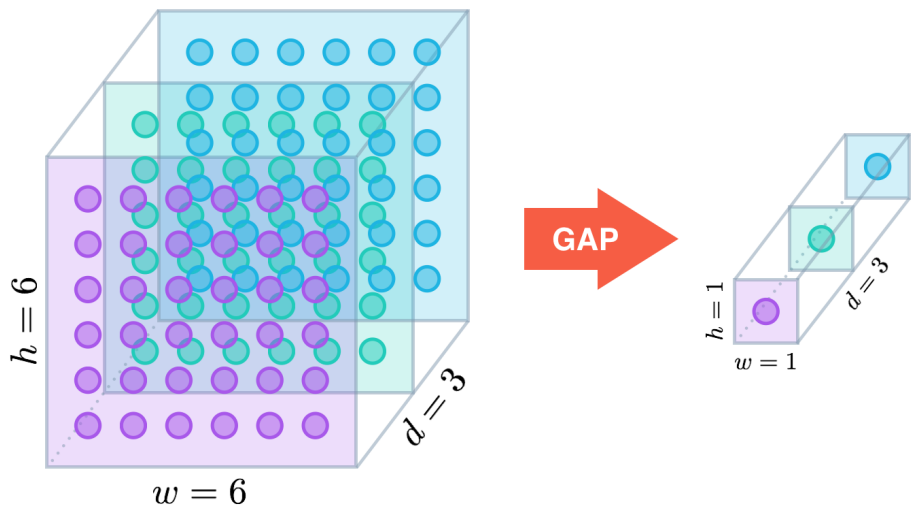


Figure: Global Average Pooling [3]

[3] From Alexis Cook's blog.

More on Global Average Pooling

First proposed in Network In Network, 2014.

Global Average Pooling reduces each feature map to a single number.

- *Feature map* is the output(s) of a convolutional layer, sometimes also referred to as "Convolved Feature" or "Activation Map".

Global *Maximize* Pooling could be used for localization - but only to a specific point. (Maxime Oquab et al.)

Upsampling

- Goal: "reshape" the feature map to the same size of the input figure.
- Methods:
  - Most common upsampling (zero padding + interpolation) methods:
    - Nearest Neighbor
    - Bilinear: $(x, y \in [0, 1])$

$$f(x,y) \approx \begin{bmatrix} 1-x & x \end{bmatrix} \begin{bmatrix} f(0,0) & f(0,1) \\ f(1,0) & f(1,1) \end{bmatrix} \begin{bmatrix} 1-y \\ y \end{bmatrix}$$

    - Bicubic
    - other, e.g. Lanczos resampling
  - Transposed convolution: trainable (ICLR'16, Alec & Luke et al.)
  - Unpooling, ...
- Well implemented in libraries. Bilinear interpolation is actually done by "deconvolution" (transposed convolution) in caffe. [4]

[4] `https://github.com/vdumoulin/conv_arithmetic`

An example: based on the Intersect Over Union (ioU) between the prediction and the ground truth.

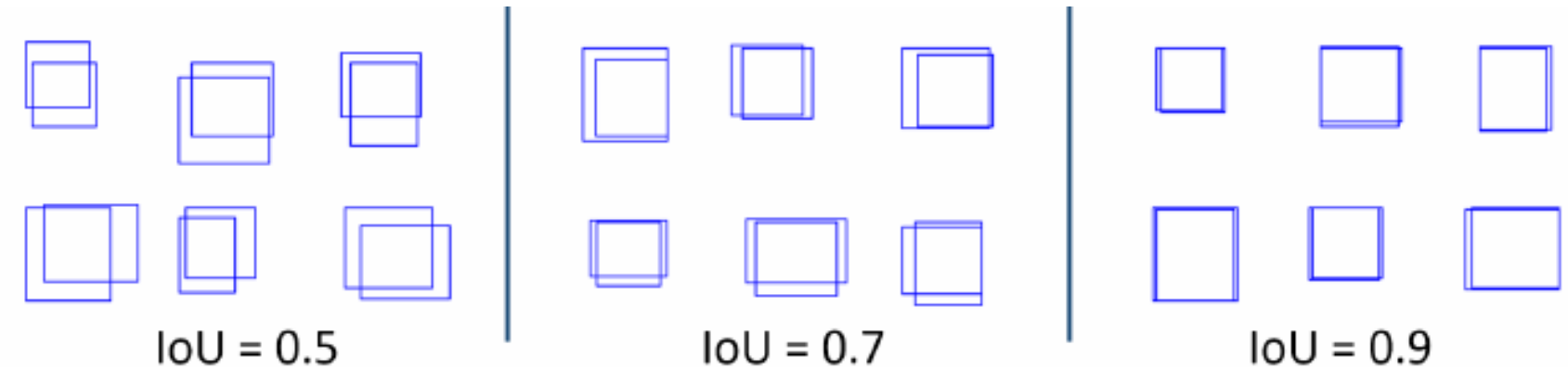


Figure: Illustration. Figure comes from *leonardoaraujosantos's gitbooks*

CNN Explaining works selected

- *Bolei Zhou et al.* Learning Deep Features for Discriminative Localization *(CVPR'16, MIT group)*
- *Ramprasaath R. Selvaraju* Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization *(ICCV'17, VT & GATech)*

Methods in Related Work

Object Localization:

- Option 1: Masking out regions;
- Option 2: Combining multiple-instance learning;
- Option 3: Use mid-level image representation.
- ...

Visualization:

- Analyzing the convolutional layers (ignoring the fully-connected layers): e.g. using deconvolutional networks to visualize pattern $\rightarrow$ unit activation;
- Inverting deep features at all layers (showing what information is preserved, but no importance information)
- ...

Main points:

- The convolutional layers actually behave as object detectors despite **no** localized label provided
- CNNs + GAP layers, trained for a classification, can also do object localization; heat map ("class activation map") is used to denote the localization.
- Pros: "a glimpse into the soul of CNNs"
  - End-to-end with only one forward pass
  - Accurate discriminative localization
  - Understanding the network from the beginning to the end
  - Highlighting the relative importance or regions

Class Activation Mapping (CAM)

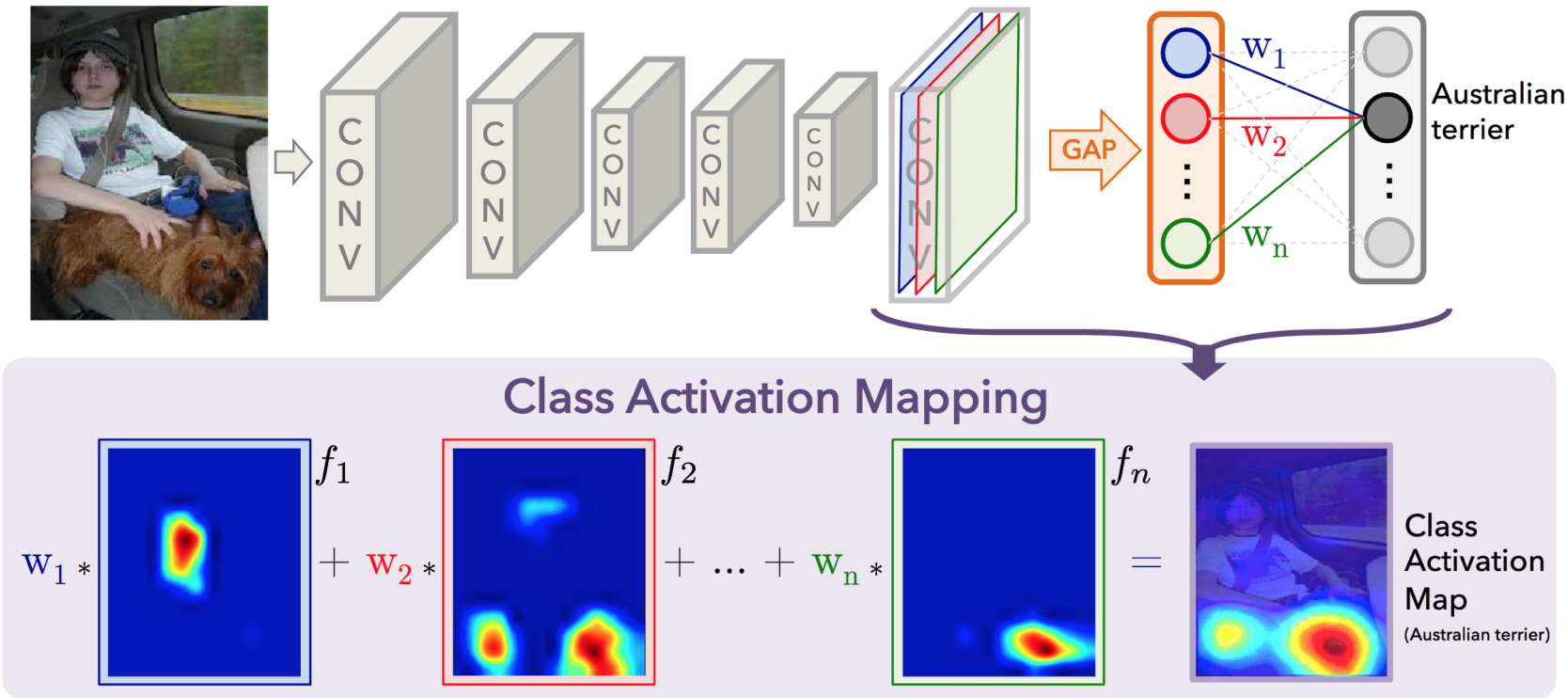


Figure: Figure 2 from Zhou's paper. f_i are the feature maps (upsampled to the original size). "We use class activation map to refer to the weighted activation maps generated for each image".

Class Activation Mapping (CAM)

$$S_c = \sum_k w_k^c \sum_{x,y} f_k(x,y) = \sum_{x,y} \sum_k w_k^c f_k(x,y)$$

where S_c is the class score, (x, y) the index of a point, k the index of feature map (among all feature maps), it is also the index of the unit in the GAP layer, and w_k^c the weight corresponding to class c for unit k.

Class Activation Mapping (CAM)

$$S_c = \sum_k w_k^c \sum_{x,y} f_k(x,y) = \sum_{x,y} \sum_k w_k^c f_k(x,y)$$

where S_c is the class score, (x, y) the index of a point, k the index of feature map (among all feature maps), it is also the index of the unit in the GAP layer, and w_k^c the weight corresponding to class c for unit k.

$$M_c(x,y) = \sum_k w_k^c f_k(x,y)$$

where M_c is the CAM for class c.

Training

Training:

- train a one-vs-all linear SVM for each scene category;
- compute the CAMs using the weights of the linear SVM.

How to measure

1. Segment the regions of which the value is above 20% of the max value of the CAM;
2. Take the bounding box covering the largest connected component in the segmentation map;
3. Do this for top-5 predicted classes (for the top-5 localization evaluation metric).
4. Heuristic: as a trade-off between classification and localization accuracy, select two bounding boxes (tight + loose) from the class activation map of the top-2 predicted classes and one loose bounding boxes from the top 3^{rd} predicted class. (Helps)

Experiment Results

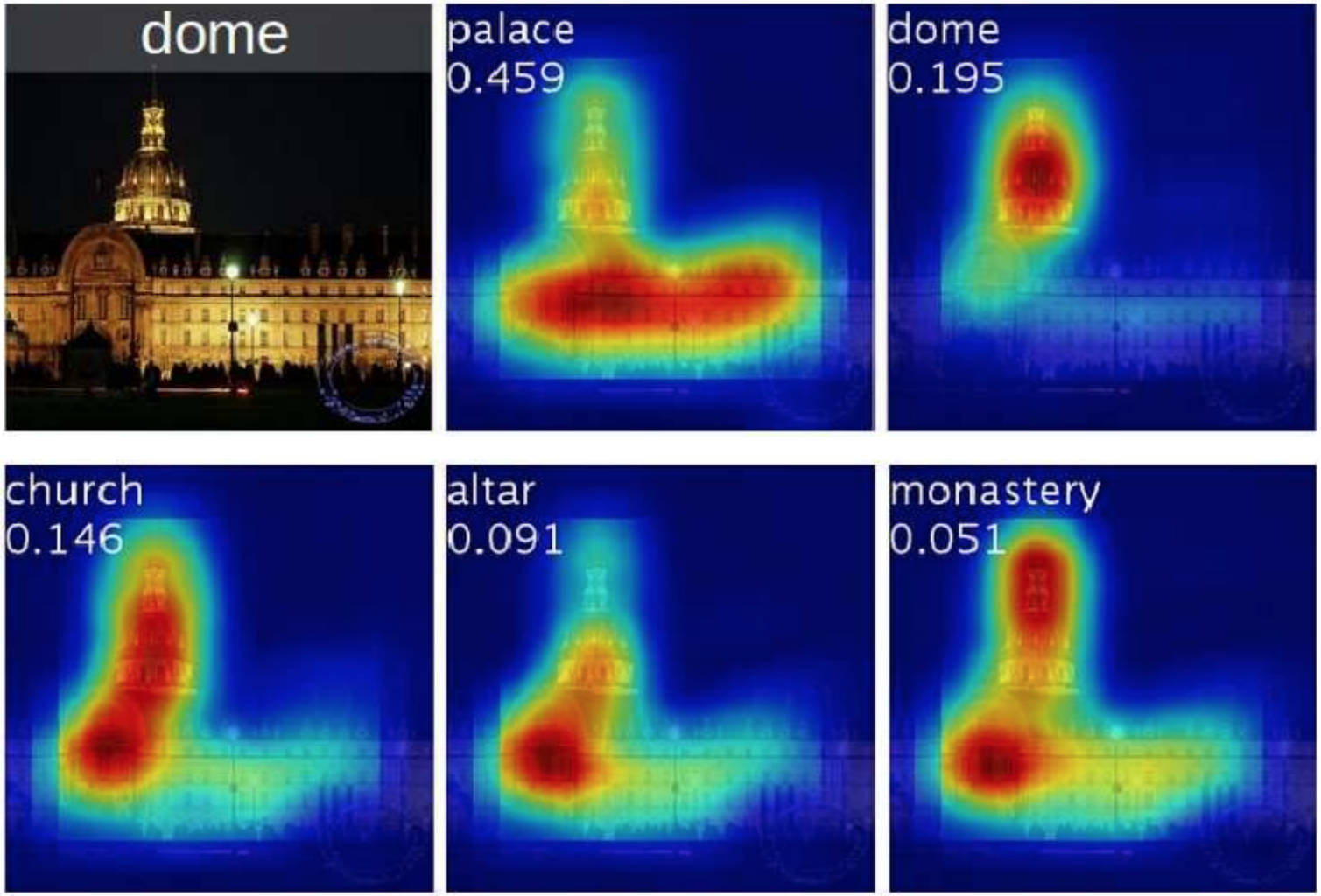


Figure: An example of the localization results.

- Global Average Pooling vs. Global Max Pooling: similar classification accuracy, better localization.
- GoogLeNet-GAP performs comparably to existing approaches even without bounded-box annotation. Could be even better if:
  - First run: threshold-localization boxing by CAM;
  - Second run: crop the areas outside the boxes and use the inner part to extract features.
- After changing the structure, a small performance drop (1% to 2 %).

Experiment Results

Visualizing Class-Specific Units:

- Definition: class-specific units
- Method:
  1. Getting GAP.
  2. Segmenting the image using receptive fields and feature maps of different units (similar approach with *OBJECT DETECTORS EMERGE IN DEEP SCENE CNNS*) and then softmax the weights to find the most important units.
     - In the **final convolutional layer**, segment the top activation images **of each unit**.
     - Calculate the softmax weights to rank the units for a given class
  3. *There're naturally receptive field differences in different units; as the layers go deeper, the RF size gradually increases and the activation regions become more semantically meaningful.

Experiment Results

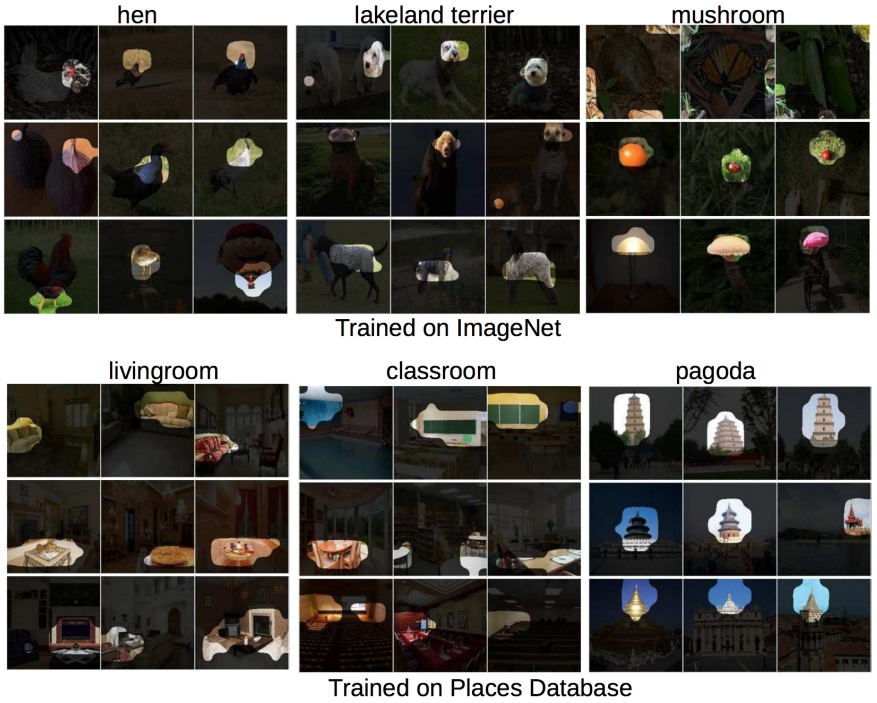


Figure: An example of the visualization results. Top 3 units for each selected class is shown for each dataset. Each row shows the *most confident images* segmented by the receptive field of *a specific unit*. For example, units detecting blackboard, chairs, and tables are important to the classification of classroom for the network trained for scene recognition

Grad-CAM: Why?

Why Grad-CAM is needed when we have CAM?

CAM has certain limitations:

- network architecture is changed, no FC;
- accuracy is harmed

A generalization of CAM:

- make existing state-of-the-art deep models interpretable (apply to more general CNNs):
  - could include FC layers;
  - could be used for structured outputs (e.g. captioning);
  - could be CNNs used in tasks with multi-modal inputs (e.g. VQA) or reinforcement learning.
- without altering the architecture;

Grad-CAM: Idea

Good visual explanation (def):

- class-discriminative (e.g. localization)
- high-resolution (e.g. capture fine-grained detail)

[5] For more, Salient deconvolutional networks.

Grad-CAM: Idea

Good visual explanation (def):

- class-discriminative (e.g. localization)
- high-resolution (e.g. capture fine-grained detail)

Existing Approaches: [5]

- Pixel-space gradient visualizations (e.g. Guided Backpropagation / Deconvolution): high-resolution but not class-discriminative
- CAM: highly class-discriminative

[5] For more, Salient deconvolutional networks.

Good visual explanation (def):

- class-discriminative (e.g. localization)
- high-resolution (e.g. capture fine-grained detail)

Existing Approaches: [5]

- Pixel-space gradient visualizations (e.g. Guided Backpropagation / Deconvolution): high-resolution but not class-discriminative
- CAM: highly class-discriminative

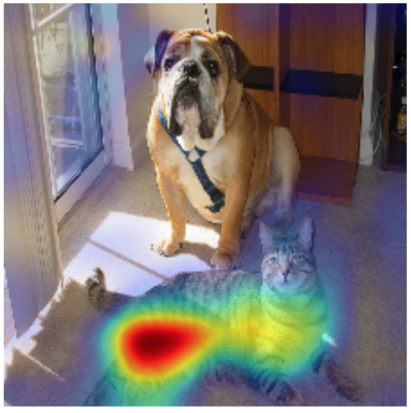
Goal: Combine the best of both worlds to explain the *decisions* made.

[5] For more, Salient deconvolutional networks.

Grad-CAM: Idea

(a) Original Image

(b) Guided Backprop 'Cat'



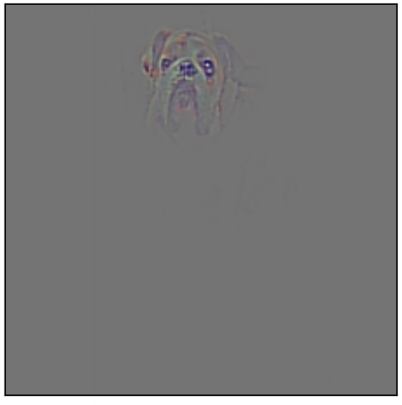
(c) Grad-CAM 'Cat'

(d) Guided Grad-CAM 'Cat'

(g) Original Image

(h) Guided Backprop 'Dog'

(i) Grad-CAM 'Dog'



(j) Guided Grad-CAM 'Dog'

Figure: Selected from Figure 1 of Grad-CAM paper.

Grad-CAM: Idea

Gradient-weighted Class Activation Mapping (**Grad-CAM**):

> *"Using the gradient information flowing into the last convolutional layer of the CNN to understand the importance of each neuron for a decision of interest."*

Grad-CAM: Compute

Grad-CAM of width u and height v for class c:
$L^c_{Grad-CAM} \in \mathbb{R}^{u \times v}$.

1. Get the score for class c (before softmax): y^c
2. Denote the output feature maps of a convolutional layer as A^k
3. Compute the gradient of the score for class c w.r.t. A^k: $\frac{\partial y^c}{\partial A^k}$
4. The gradients via back-propagation ($\frac{\partial y^c}{\partial A^k_{ij}}$) through GAP are the neuron importance weights:

$$\alpha^c_k = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A^k_{ij}} \in \mathbb{R}$$

where α^c_k captures the "importance" of feature map A^k for a target class c.

5. Weighted combination of *forward activation maps* [6]: $\sum_k \alpha_k^c A^k \in \mathbb{R}^{u \times v}$
6. Finally a ReLU:

$$L^c_{Grad-CAM} = ReLU(\sum_k \alpha_k^c A^k) \in \mathbb{R}^{u \times v}$$

which is a *coarse heat-map* of the *same size* as the convolutional feature maps. In general, y^c could be **any** type of input, not only class score.

[6] forward activation maps = feature maps

Recall that in standard CAM:

$$S_c = \sum_k w_k^c \sum_{x,y} f_k(x,y) = \sum_{x,y} \sum_k w_k^c f_k(x,y)$$

$$M_c(x,y) = \sum_k w_k^c f_k(x,y)$$

For the standard CAM it could also be expressed as:

$$\begin{aligned} S_c &= \sum_k w_k^c \frac{1}{Z} \sum_i \sum_j A_{ij}^k \\ &= \frac{1}{Z} \sum_i \sum_j \sum_k w_k^c A_{ij}^k \\ &= \frac{1}{Z} \sum_i \sum_j L_{CAM}^c \end{aligned}$$

For CAM:

$$S_c = \frac{1}{Z} \sum_i \sum_j L^c_{CAM}$$

When Grad-CAM is applied:

$$S_c = \sum_i \sum_j L^c_{Grad-CAM}$$

$$L^c_{Grad-CAM} = ReLU(\sum_k \alpha^c_k A^k)$$

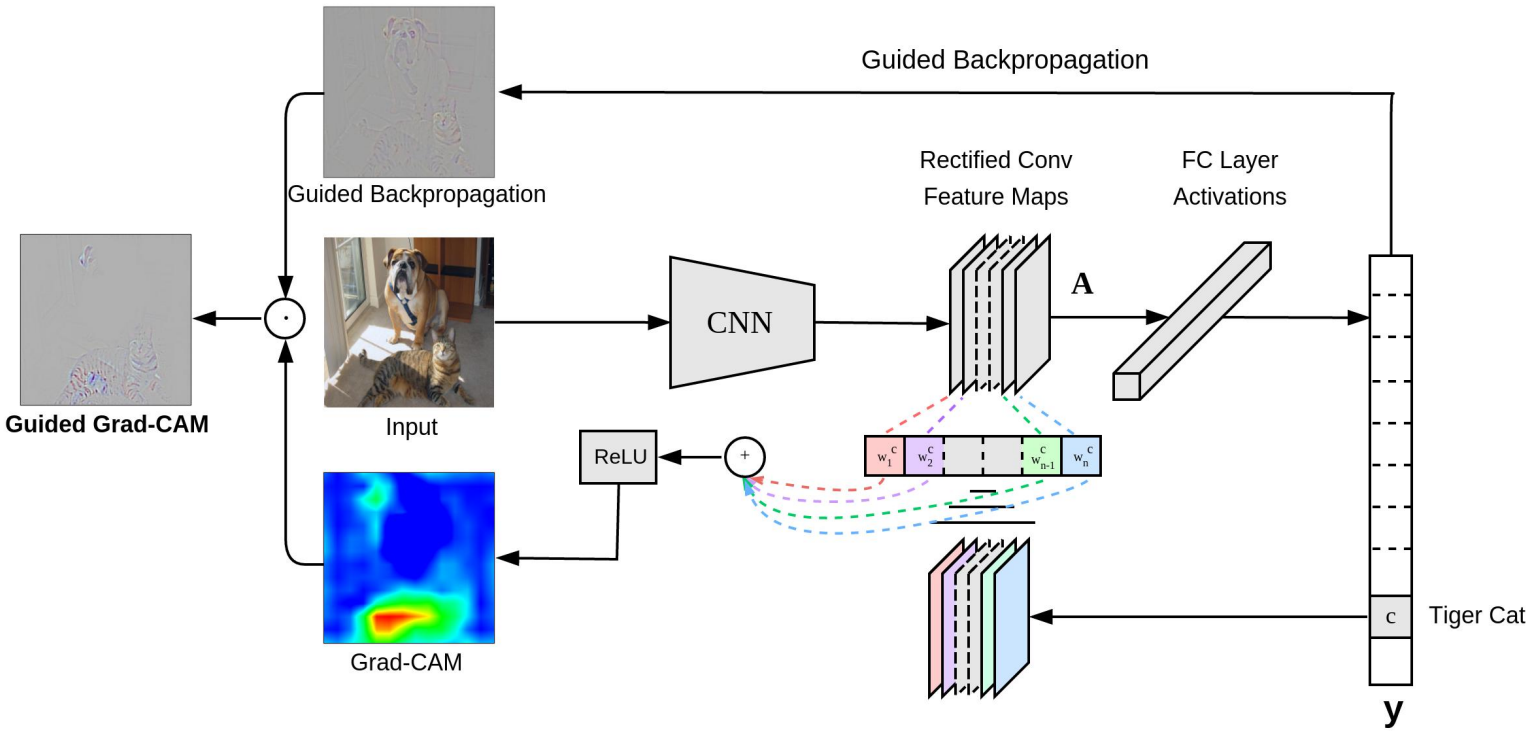


Figure: Guided Grad-CAM pipeline.

Experiment Settings

- Evaluating Trust: human identify crowd-sourcing
- Case studies:
  - Failure modes: use Guided Grad-CAM to analyze; *seemingly unreasonable predictions have reasonable explanations*, e.g. ambiguities inherent in ImageNet classification
  - Proved the robustness of Grad-CAM to adversarial noise
  - Identifying **bias in dataset**: e.g. Grad-CAM visualizations of the model predictions revealed that, the model had learned to look at the person's face / hairstyle to distinguish nurses from doctors, thus learning a gender stereotype.
  - ...

GNN Explainer

Why explaining GNN?

- Similar reasons with explaining CNNs.
- *We* use GNNs more often.
- Mush fewer interpretation work is done for GNNs than for CNNs.

Rex Ying et al. GNN Explainer: A Tool for Post-hoc Explanation of Graph Neural Networks [7]

[7] Stanford team, on ArXiv

General expression 1:

$$
\begin{aligned}
m_{ij}^{l} &= MSG\big(h_i^{l-1}, h_j^{l-1}, r_{ij}\big) \\
M_i^{l} &= AGG\big(\{m_{ij}^{l} | v_j \in \mathcal{N}_{v_i}\}\big) \\
h_i^{l} &= UPDATE\big(M_i^{l}, h_i^{l-1}\big)
\end{aligned}
$$

General expression 2: [8]

$$
h_i^{(l+1)} = \sigma\Big(\sum_{m \in \mathcal{M}_i} g_m(h_i^{(l)}, h_j^{(l)}) \Big)
$$

[8] Gilmer et al. 2017

General expression 1:

$$m_{ij}^{l} = MSG\big(h_i^{l-1}, h_j^{l-1}, r_{ij}\big)$$
$$M_i^{l} = AGG\big(\{m_{ij}^{l} | v_j \in \mathcal{N}_{v_i}\}\big)$$
$$h_i^{l} = UPDATE\big(M_i^{l}, h_i^{l-1}\big)$$

General expression 2: [8]

$$h_i^{(l+1)} = \sigma\Big(\sum_{m \in \mathcal{M}_i} g_m(h_i^{(l)}, h_j^{(l)})\Big)$$

Idea: *"recursively incorporate information/messages from neighboring nodes in the network, naturally capturing the graph structure simultaneously with the node features."*

[8] Gilmer et al. 2017

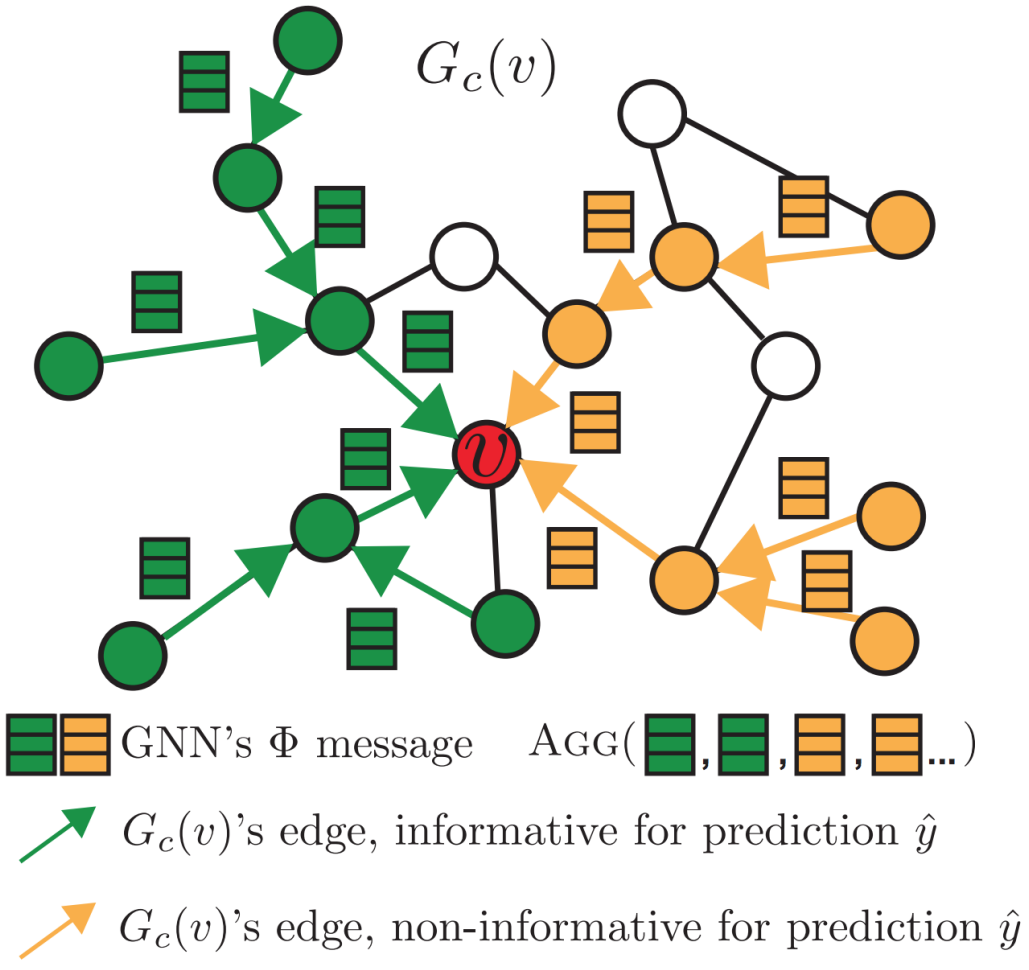


Figure: Illustration of GNN.

GNN Explainer: Idea

Defining computation graph $G_c(v_i)$ as an L-hop neighborhood of v_i in the input graph $G = \{V, E\}$.

Should consider:

- Local edge/node fidelity: graph structure and feature (if any) in G_c;
- The explanation should summarize **where** in G the GNN model looks for evidence for its prediction, and identify the subgraph of G most responsible for a given prediction.
- Apply to any model in GNN-family / model-agnostic: e.g. treat GNN as blackbox;
- Apply to any prediction task on graphs / task-agnostic: e.g. entity classification, link prediction, graph classification.

GNN Explainer: Problem Formulation

Consider node-classification-task case.

- Graph $G = \{V, E\}$ with nodes V and edges E. $|V| = n$. C classes in total. G_c is defined on previous page.
- Node features $\mathcal{X} = \{x_1, \ldots, x_n\}$, where $x_i \in \mathbb{R}^d$, and $X_c = \{x_j | v_j \in G_c(v_i)\}$.
- Label function $f : V \to \{1, 2, \ldots C\}$. $Y = \{1, 2, \ldots C\}$
- GNN model Φ, predicted label y.

Φ is optimized for all nodes in training set (so as to approximate f on testing set).

- Given trained Φ and predicted label(s);
- Consider edge connectivity patterns and node features that provide insights on what the model has learned;
- Generate an explanation.

It is a **post-hoc** interpretation of predictions generated by a **pre-trained** GNN.

Formulated in an *optimization framework*.

The key insight: naturally localized [9]. The prediction $\hat{y}$ is,

$$\hat{y} = \Phi(G_c(v_i), X_c(v_i))$$

GNN Explainer identifies (**jointly determined**) $G_S \subseteq G_c(v_i)$ and a subset of $X_c(v_i)$, $X_S = \{x_i | i \in G_S\}$ that are *most influential* for the prediction.

[9] Brought by convolution, similar with CNN.

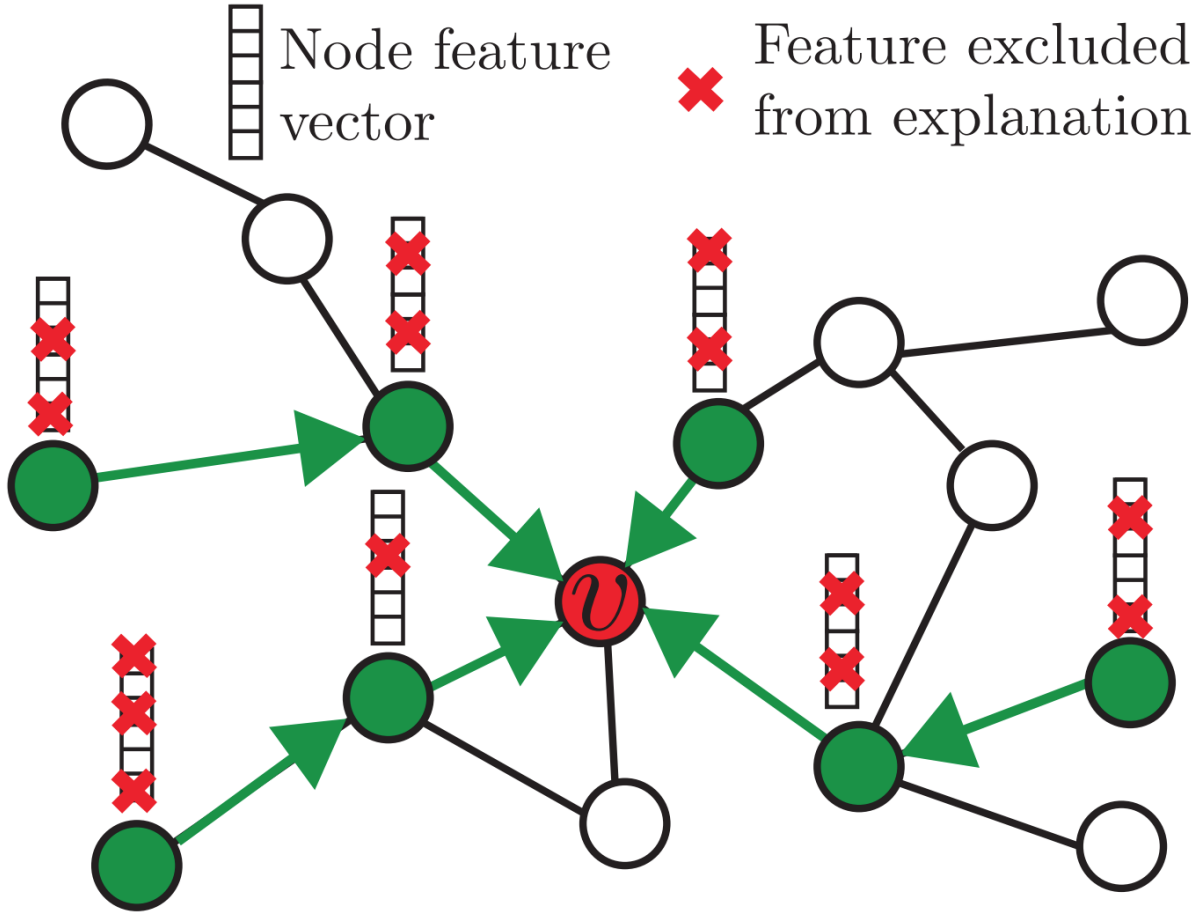


Figure: Illustration of v's explanation G_S, denoted in green.

MI: mutual information, the notation of importance.

$$\begin{aligned} \underset{G_S}{\text{maximize}} \quad & MI(Y, (G_S, X_S)) = H(Y) - H(Y|G = G_S, X = X_S) \\ \text{subject to} \quad & G_S \subseteq G_c(v_i) \\ & X_S = \{x_i | i \in G_S\} \end{aligned}$$

where the entropy term $H(Y)$ is fixed, thus the above problem is equivalent with:

$$\begin{aligned} \underset{G_S}{\text{minimize}} \quad & H(Y|G = G_S, X = X_S) \\ \text{subject to} \quad & G_S \subseteq G_c(v_i) \\ & X_S = \{x_i | i \in G_S\} \end{aligned}$$

The conditional entropy $H(Y|G = G_S, X = X_S)$ could be expressed as:

$$\begin{aligned} & H(Y|G = G_S, X = X_S) \\ = & - \mathbb{E}_{Y|G_S,X_S}[\log P_\Phi(Y|G = G_S, X = X_S)] \end{aligned}$$

Intuitively:

> *Explanation for v_i's prediction $\hat{y}$ is a subgraph G_S and the associated features' subset X_S that minimize the uncertainty of the GNN Φ when the neural message-passing is limited to G_S.*
> *Not all parts of Φ are relevant or important.*

Denoising G_c: keep at most k edges that have the highest MI (for given Y and G_c.).

$$\begin{aligned} \underset{G_S}{\text{minimize}} \quad & H(Y|G = G_S, X = X_S) \\ \text{subject to} \quad & G_S \subseteq G_c(v_i) \\ & X_S = \{x_i | i \in G_S\} \end{aligned}$$

Approximation: [10] similar to soft-attention, from $\{0, 1\}$ to $[0, 1]$. (ECE236B Winter 2018 hw trick) Denote as $G_S \sim \mathcal{G}$, or $G_S = A_S \odot \sigma(M)$ where M is mask.
The cost function is approximated as:

$$\underset{\mathcal{G}}{\text{minimize}} \quad \mathbb{E}_{G_S \sim \mathcal{G}} H(Y|G = G_S, X = X_S)$$

When Φ convex, from Jensen's inequality, we have it equivalent with:

$$\underset{\mathcal{G}}{\text{minimize}} \quad H(Y|G = \mathbb{E}_{\mathcal{G}}[G_S], X = X_S)$$

[10] Directly optimizing is not tractable.

GNN Explainer: Node Features Included

Given G_S (identified explanatory subgraph of G_c), select a more precise subset of X from $X_S = \{x_i | v_i \in G_S\}$.

Different from getting G_S:

- Option 1: for each selected node, optimize the selection of features;
  - finer granularity
  - but more complex explanations
- **Option 2**: optimize a single set of important dimensions of all nodes' features
  - more efficient for computation
  - and provides more concise explanations

GNN Explainer: Node Features Included

Defining the target subset of X_S as X_{ST}, with $T \in \{0,1\}^D$ being the binary feature selector.

Recall that previously we have:

$$\begin{aligned} \underset{G_S}{\text{maximize}} \quad & MI(Y, (G_S, X_S)) = H(Y) - H(Y|G = G_S, X = X_S) \\ \text{subject to} \quad & G_S \subseteq G_c(v_i) \\ & X_S = \{x_i | i \in G_S\} \end{aligned}$$

With feature-selection it becomes:

$$\begin{aligned} \underset{G_S, T}{\text{maximize}} \quad & MI(Y, (G_S, X_{ST})) = H(Y) - H(Y|G = G_S, X = X_{ST}) \\ \text{subject to} \quad & G_S \subseteq G_c(v_i) \\ & X_{ST} \subseteq \{x_i | i \in G_S\} \end{aligned}$$

GNN Explainer: Node Features Included

Optimization: similar with before, use mask, $X_{ST} = X_S \odot M_T$, $M_T \in \mathbb{R}^{|num\_features|}$ is the feature selection mask matrix to be learned.

Different from selecting edges: **zero** could be information. Therefore, they marginalize over all possible choices of features, using Monte Carlo to estimate the marginals.

Key challenge: how to perform backpropagation through random variable X? (allows the backprop of grad from the objective function to the M_T)

- using a reparametrization trick for X

$$\underset{G_S,T}{\text{minimize}} \quad H(Y|G = G_S, X = X_{ST})$$

$$X = Z + (X_S - Z) \odot M_T, \qquad s.t.\ \mathbb{1}^T M_T \leq k$$

GNN Explainer: Additional Constraints

- Explanation size: too large is useless, thus regularize

$$|G_S| = \sum_{i,j} (A_S)_{i,j} \sim \sum_{i,j} \sigma(M_{i,j}) \leq k$$

  where the threshold k is user-specified.
- Application-specific prior knowledge: included by using Laplacian regularization, $f^T L_S f$.
- Mask discreteness: (e.g. masks of structural graph patterns) optimized by regularizing the (sum) element-wise entropy of the mask.
- The explanation, as a subgraph of G_c, should be a valid computation graph. Naturally achieved in GnnExplainer without explicit regularization; it encourages connection to the center v_i while learning.

"“why is a given set of nodes classified with label y”, a global explanation of the class, relation between the identified structure for a given node and a prototypical structure unique for its label.

Solution: an alignment-base multi-instance GnnExplainer.

1. Choose a reference node prototypical for the class c, denoted as v_c. (Choose the one with closest embedding to the class-mean, or simply use prior knowledge.)
2. Get the reference computation subgraph $G_S(v_c)$.
3. Use a relaxed alignment matrix to find correspondence between nodes in $G_S(v_c)$ and in $G_S(v)$, $\forall v$. The relaxed alignment matrix $P \in \mathbb{R}^{n_v \times n^*}$, where there are n_v nodes in $G_S(v)$, n^* nodes in $G_S(v_c)$.

$$\underset{P}{\text{minimize}} \, |P^T A_v P - A^*| + |P^T X_v - X^*|$$

where A^*, X^* are adjacency matrix and feature matrix of reference v_c, A_v and X_v for v.

Prototype by Alignment:

1. align the adjacency matrices of all nodes in class c w.r.t. the ordering defined by *the reference adjacency matrix*.
2. take the average of each edge to generate a prototype:

$$A^c_{prototype} = \frac{1}{n} \sum_i A^c_i$$

It allows users to gain insights into structural graph patterns shared between nodes that belong to the same class.

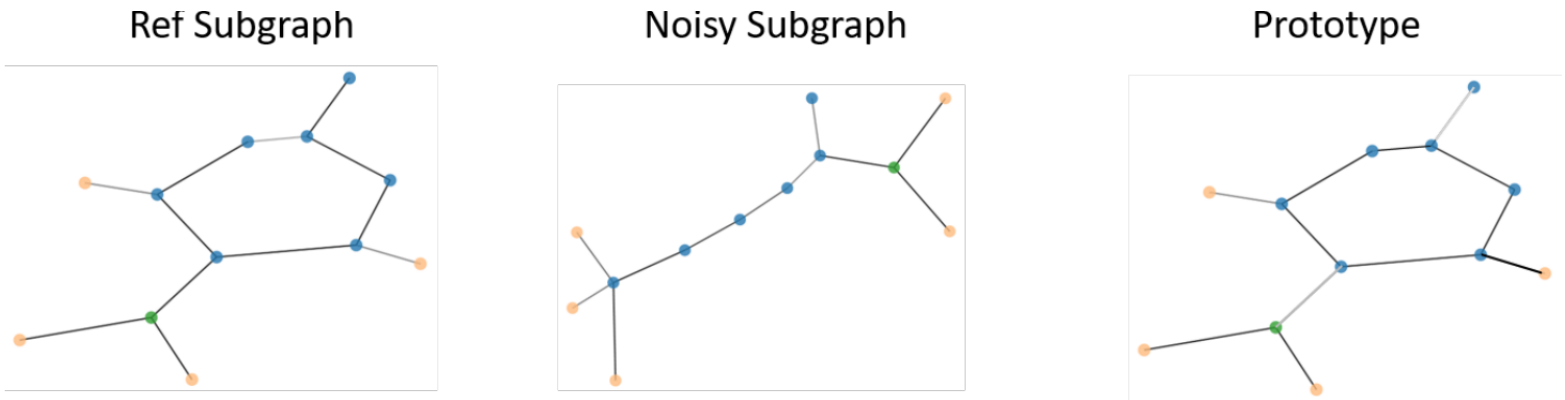


Figure: Multiple-Instance Prototype by Alignment

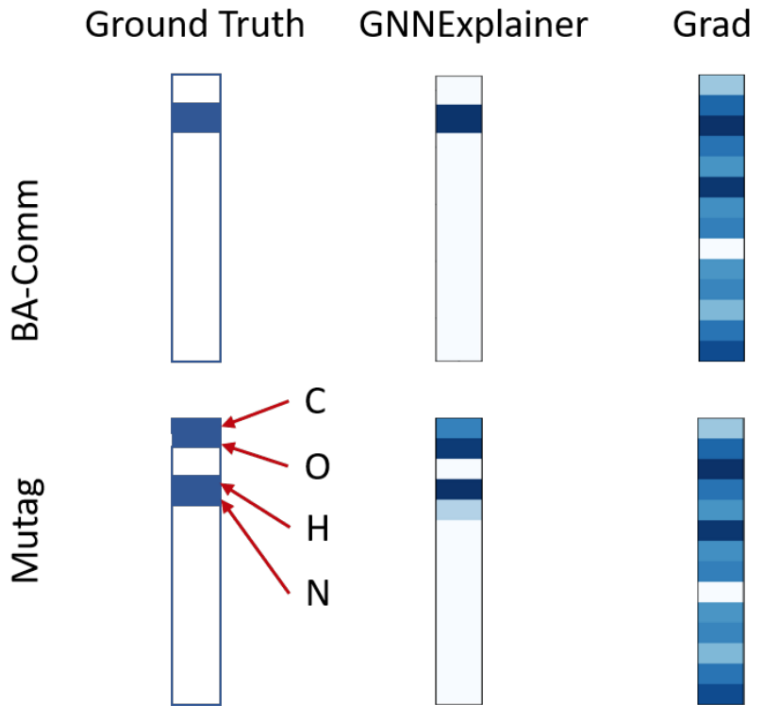


Figure: Feature Importance Visualization

GNN Explainer: Experiment

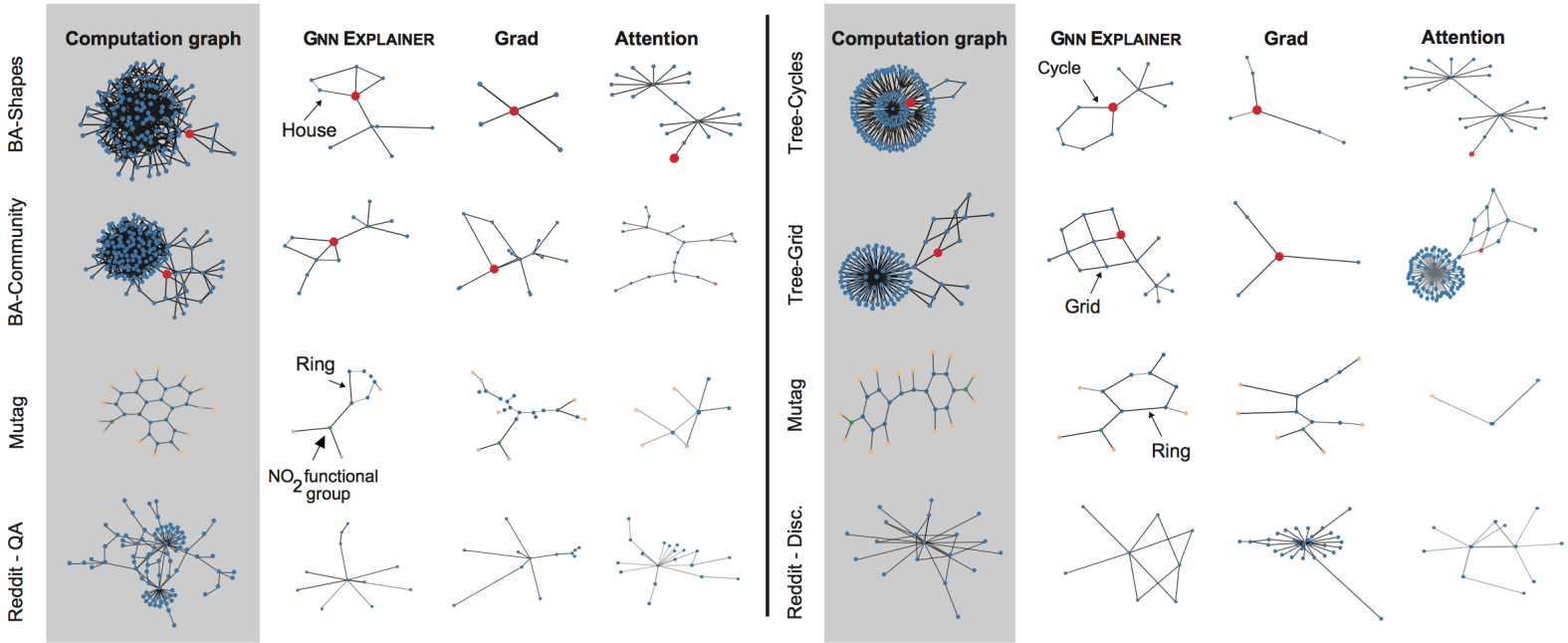


Figure: Results of GnnExplainer + baselines. Red nodes explained. The groundtruths (cycle, house, grid) are used for analyzing the explainer's performance.

More To Read

More . . .

- *Guillaume Alain et al.* What Regularized Auto-Encoders Learn from the Data-Generating Distribution *(JMLR' 15, University of Montreal group)*
- *Jianbo Chen et al.* Learning to Explain: An Information-Theoretic Perspective on Model Interpretation *(ICML'18, Berkeley group)*
- *(Julius Adebayo et al.)* Sanity Checks for Saliency Maps *(NeurIPS'18, Google Brain & Berkeley)*
- *(Hao Li et al.)* Visualizing the Loss Landscape of Neural Nets *(NeurIPS'18)*
- . . .

The End

Thank you all for coming! ☺