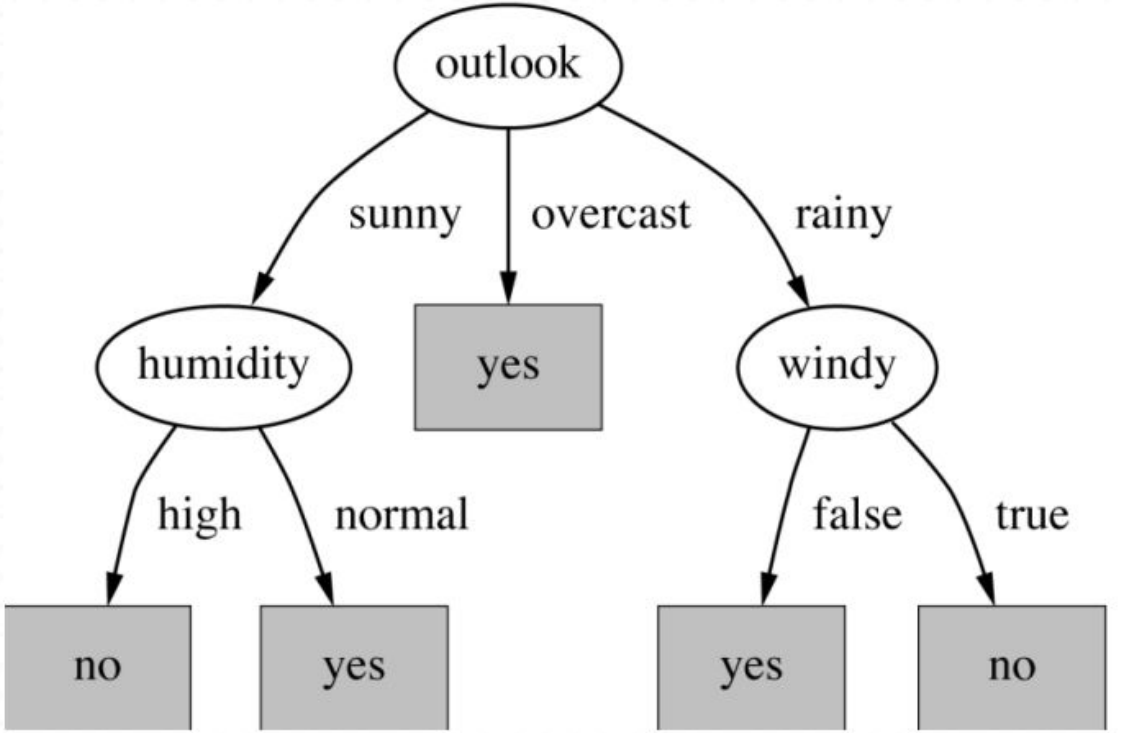
DNNs Interpretability

Georgios M. Moschovis

Introduction to DNNs interpretability goals

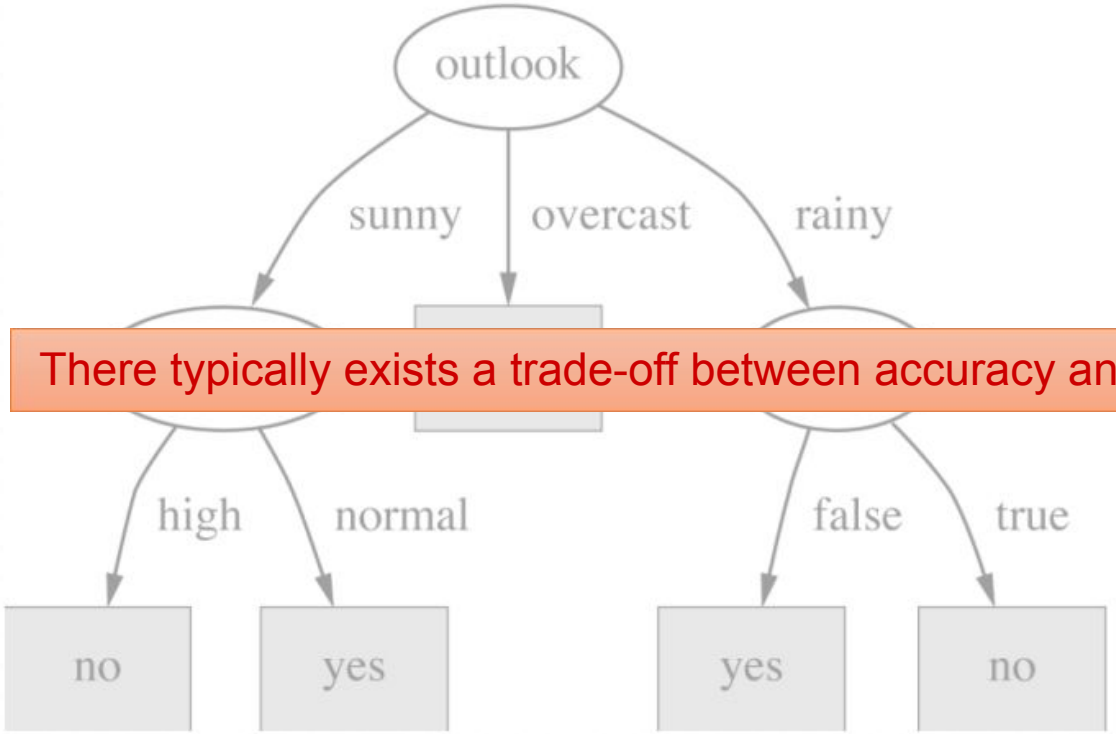
- **The core idea:** specify how a particular neural network comes to a decision, how certain it is about the decision, if and when it can be trusted, or when it has to be corrected (design more transparent models).
- *Explainable AI*: providing explanations for the artificially intelligent model's decision. Specifically, we are interested in explaining classification decisions made by deep neural networks on text or natural images.
  - A *saliency* or *importance map* shows how important each token or image pixel is for the network's prediction.
  - White box (based on the network gradients etc.) vs black box (can work with arbitrary networks) approaches.
- The visual grounding is generally expressed as a saliency or importance map which shows the importance of each token or pixel towards the model decision. Such maps are created for instance using attention operation for input-output relations (e.g. how each word in a caption relates to a pixel).

Introduction to DNNs interpretability goals



Figures from Chefer et al. “Transformer Interpretability Beyond Attention Visualization”, CVPR 2021

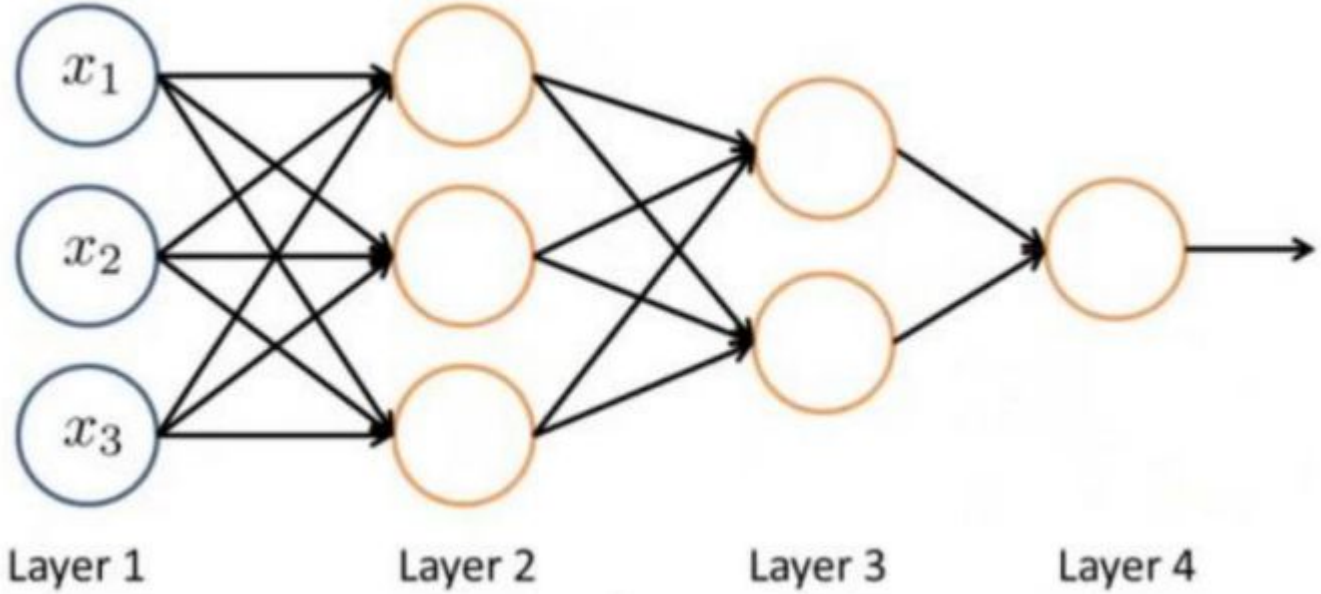
Introduction to DNNs interpretability goals



There typically exists a trade-off between accuracy and simplicity or interpretability.

Figures from Chefer et al. “Transformer Interpretability Beyond Attention Visualization”, CVPR 2021

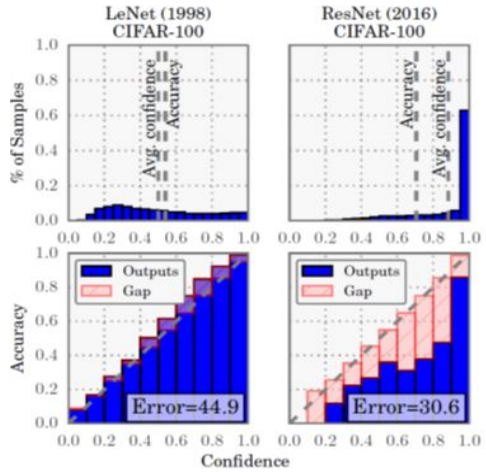
Introduction to DNNs interpretability goals

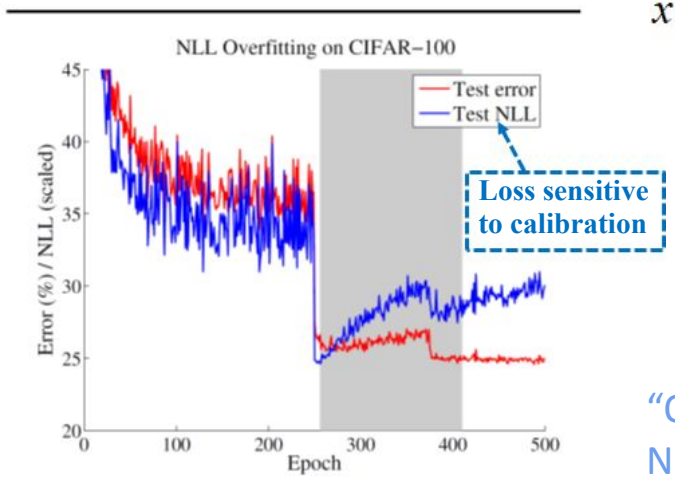


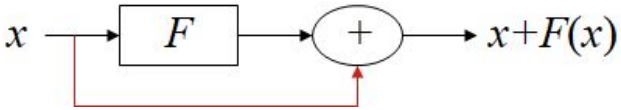
In DNNs with increasing numbers of layers there is a tension between *accuracy* and *interpretability*. Additionally accuracy is dependent to the preferred metric e.g. AUC, P-R, log likelihood etc. (recall sensitivity to calibration).

Introduction to DNNs interpretability goals

- Calibration is the notion (measure) of statistical consistency between the predictive distributions and the actual observations; e.g. if we expect 75% of our data to be of $C = 1$, where $C \in \{0,1\}$, the prediction is calibrated.
- Problem with ResNet: although accuracy improves compared to its counterparts we get lower calibration as seen in Figure 1.







Recap. from uncertainty estimation

"On Calibration of Modern Neural Networks" ICML 2017

Several approaches on interpreting DNNs

- The **Class Activation Mapping (CAM)** approach achieves class-specific importance of each location of an image by computing a weighted sum of the feature activation values at that location across all channels.

For features $\mathcal{F}$ and channels $\mathcal{C}$: $\sum_{f \in \mathcal{F}} \sum_{c \in \mathcal{C}} w_f \times f$

- **Grad-CAM** extends CAM by weighing the feature activation values at every location with the average gradient of the class score (w.r.t. the feature activation values) for every feature map channel.

- **RISE** is a more general black-box framework that estimates the importance of pixels by dimming them in random combinations, reducing their intensities to zero by multiplying an image with a $[0, 1]$ valued mask.

$$f: \mathcal{I} \to \mathbb{R}, \mathcal{I} = \left\{ I \middle| I: \Lambda \to \mathbb{R}^3 \right\}, \Lambda = (\{1, \dots, H\} \times \{1, \dots, W\})$$

Several approaches on interpreting DNNs

- The **Class Activation Mapping (CAM)** approach achieves class-specific importance of each location of an image by computing a weighted sum of the feature activation values at that location across all channels.

  For features $\mathcal{F}$ and channels $\mathcal{C}$: $\sum_{f \in \mathcal{F}} \sum_{c \in \mathcal{C}} w_f \times f$

- **Grad-CAM** extends CAM by weighing the feature activation values at every location with the average gradient of the class score (w.r.t. the feature activation values) for every feature map channel.

- **RISE** is a more general black-box framework that estimates the importance of pixels by dimming them in random combinations, reducing their intensities to zero by multiplying an image with a $[0, 1]$ valued mask.

$$f: \mathcal{I} \to \mathbb{R}, \mathcal{I} = \left\{ I \middle| I: \Lambda \to \mathbb{R}^3 \right\}, \Lambda = (\{1, \dots, H\} \times \{1, \dots, W\})$$

Several approaches on interpreting DNNs

- The **Class Activation Mapping (CAM)** approach achieves class-specific importance of each location of an image by computing a weighted sum of the feature activation values at that location across all channels.

  For features $\mathcal{F}$ and channels $\mathcal{C}$: $\sum_{f \in \mathcal{F}} \sum_{c \in \mathcal{C}} w_f \times f$

- **Grad-CAM** extends CAM by weighing the feature activation values at every location with the average gradient of the class score (w.r.t. the feature activation values) for every feature map channel.

- **RISE** is a more general black-box framework that estimates the importance of pixels by dimming them in random combinations, reducing their intensities to zero by multiplying an image with a $[0, 1]$ valued mask.

$$f: \mathcal{I} \to \mathbb{R}, \mathcal{I} = \left\{ I \middle| I: \Lambda \to \mathbb{R}^3 \right\}, \Lambda = (\{1, \dots, H\} \times \{1, \dots, W\})$$

Several approaches on interpreting DNNs

- **Principal goals** of *Explainable* Artificial Intelligence:
  a. Identify failure modes, help researchers debug and improve their systems.
  b. Establish appropriate trust and confidence to users in the direction of:
     i. *Robustness:* convince users that the system will actually work.
     ii. *Fairness:* ensure that the system will not incorporate biases.
  c. *Machine teaching* a human about how to make better decisions.
- Class Activation Mapping and its generalization Grad-CAM work for CNNs.
- The **Class Activation Mapping (CAM)** approach achieves class-specific importance of each location of an image by computing a weighted sum of the feature activation values at that location across all channels.
- **Grad-CAM** extends CAM by weighing the feature activation values at every location with the average gradient of the class score (w.r.t. the feature activation values) for every feature map channel.

Several approaches on interpreting DNNs

- **Principal goals** of *Explainable* Artificial Intelligence.
- Class Activation Mapping and its generalization Grad-CAM work for CNNs.
- The **Class Activation Mapping (CAM)** approach achieves class-specific importance of each location of an image by computing a weighted sum of the feature activation values at that location across all channels.
  - Training objective: identifying discriminative regions used by a restricted class of image classification CNNs (aka backbones) which do not contain any fully-connected layers.
  - Trades off model complexity and performance for more transparency into the working of the model.
- **Grad-CAM** extends CAM by weighing the feature activation values at every location with the average gradient of the class score (w.r.t. the feature activation values) for every feature map channel.

Several approaches on interpreting DNNs

- **Principal goals** of *Explainable* Artificial Intelligence.
- Class Activation Mapping and its generalization Grad-CAM work for CNNs.
- The **Class Activation Mapping (CAM)** approach achieves class-specific importance of each location of an image by computing a weighted sum of the feature activation values at that location across all channels.
- **Grad-CAM** extends CAM by weighing the feature activation values at every location with the average gradient of the class score (w.r.t. the feature activation values) for every feature map channel.
  - Make existing state-of-the-art deep models interpretable without altering their architecture, thus avoiding the interpretability *vs.* accuracy trade-off and applies to a wider range of CNN models.
  - A 'good' visual explanation from the model for justifying any target class should be class discriminative (i.e. localize the category in the image) and high-resolution (i.e. capture fine-grained detail).

Several approaches on interpreting DNNs

- **Prin**
- Clas INs.
- The fic
  impo of the
  featu
- **Grad**
  locat very
  value ctivation
  - eir
  - plies to a
  - A ‘good’ visual explanation from the model for justifying any target class should be class discriminative (i.e. localize the category in the image) and high-resolution (i.e. capture fine-grained detail).

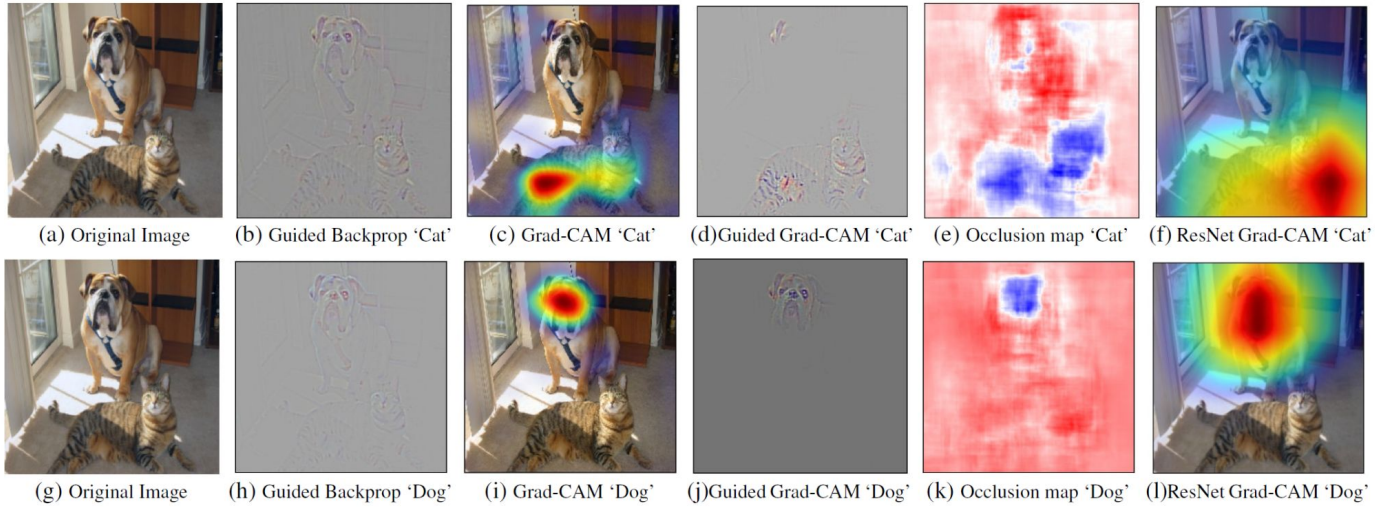


Fig. 1: (a) Original image with a cat and a dog. (b-f) Support for the cat category according to various visualizations for VGG-16 and ResNet. (b) Guided Backpropagation [53]: highlights all contributing features. (c, f) Grad-CAM (Ours): localizes class-discriminative regions, (d) Combining (b) and (c) gives Guided Grad-CAM, which gives high-resolution class-discriminative visualizations. Interestingly, the localizations achieved by our Grad-CAM technique, (c) are very similar to results from occlusion sensitivity (e), while being orders of magnitude cheaper to compute. (f, l) are Grad-CAM visualizations for ResNet-18 layer. Note that in (c, f, i, l), red regions corresponds to high score for class, while in (e, k), blue corresponds to evidence for the class. Figure best viewed in color.

Gradients Class Activation Mapping (Grad-CAM)

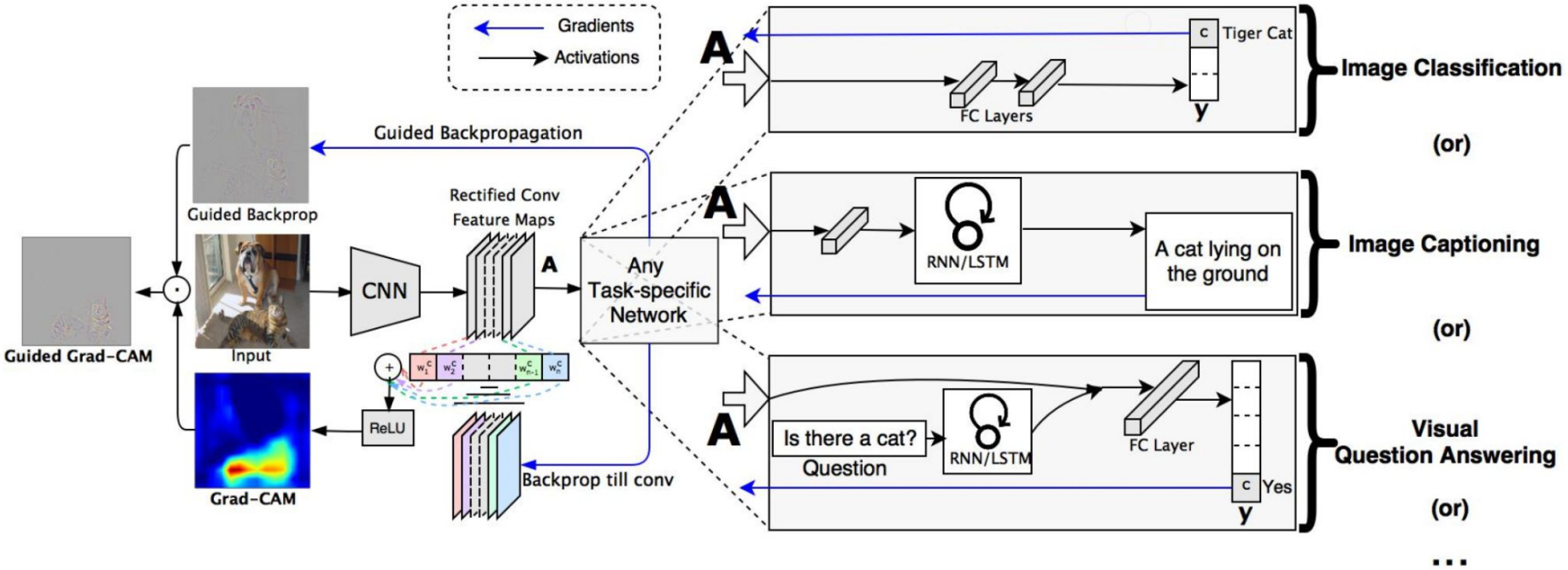


Fig. 2: Grad-CAM overview: Given an image and a class of interest (*e.g.*, 'tiger cat' or any other type of differentiable output) as input, we forward propagate the image through the CNN part of the model and then through task-specific computations to obtain a raw score for the category. The gradients are set to zero for all classes except the desired class (tiger cat), which is set to 1. This signal is then backpropagated to the rectified convolutional feature maps of interest, which we combine to compute the coarse Grad-CAM localization (blue heatmap) which represents where the model has to look to make the particular decision. Finally, we pointwise multiply the heatmap with guided backpropagation to get Guided Grad-CAM visualizations which are both high-resolution and concept-specific.

Gradients Class Activation Mapping (Grad-CAM)

- Compute the gradient of the score for class c, y_c (before the softmax), with respect to feature map activations A^k of a convolutional layer. Afterwards, flow global-average-pooled gradients back over width w and height h dimensions i.e. $\forall i \in \{1, \ldots, w\}$, $\forall j \in \{1, \ldots, h\}$ to obtain the neuron importance weights:

$$\alpha_k^c = \overbrace{\frac{1}{Z}\sum_i \sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{gradients via backprop}}$$

- While **backpropagating gradients with respect to activations**, the exact computation amounts to successive matrix products of the weight matrices and the gradient with respect to activation functions till the final convolution. Hence, weights represent a partial linearization of the CNN downstream from A, and captures the 'importance' of feature map k for a target class c.

Gradients Class Activation Mapping (Grad-CAM)

- We perform a weighted combination of forward activation maps, and follow it by a $ReLU(.)$ to obtain a **coarse heatmap** L^c of the same size as the convolutional feature maps, typically 14×14:

$$L^c_{\text{Grad-CAM}} = ReLU \underbrace{\left(\sum_k \alpha^c_k A^k \right)}_{\text{linear combination}}$$

- Without this $ReLU(.)$, localization maps sometimes highlight more than just the desired class and perform worse at localization. We thus apply a $ReLU(.)$ to the linear combination of maps because:
  - we are only interested in the features that have a *positive* influence on the class of interest, i.e. pixels whose intensity should be *increased* in order to increase y^c (positive class activations/confidence scores).
  - we are not interested in negative pixels belonging to other classes.

Gradients Class Activation Mapping (Grad-CAM)

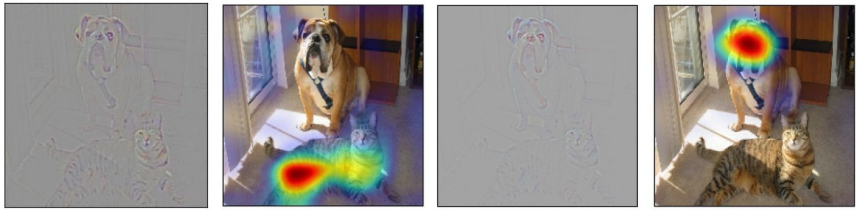
- Grad-CAM is class-discriminative and localizes relevant image regions, it lacks the ability however to **highlight fine/grained details** like pixel-space gradient visualization methods (e.g. it is unclear from the Fig. 1 coarse heatmap why the network predicts this particular instance as ‘tiger cat’).
- **Solution:** fuse Guided Backpropagation and Grad-CAM visualizations via element-wise multiplication (L^c is first upsampled to the input image resolution using bilinear interpolation).
  - **high-resolution visualization** (it identifies important ‘tiger cat’ features like stripes, pointy ears and eyes)
  - **class-discriminative visualization** (it highlights the ‘tiger cat’ but not the ‘boxer (dog)’).
- Replacing Guided Backpropagation with Deconvolution gives similar results, but Deconvolution visualizations have artifacts and Guided Backpropagation was generally proved less noisy

Gradients Class Activation Mapping (Grad-CAM)

- Grad-CAM is class-discriminative and localizes relevant imag
  the ability however to **highlight fine/grained details** like pixe
  visualization methods (e.g. it is unclear from the Fig. 1 coarse
  network predicts this particular instance as 'tiger cat').

SOLVED!

- **Solution:** fuse Guided Backpropagation and Grad-CAM visualizations via element-wise multiplication (L^c is first upsampled to the input image resolution using bilinear interpolation).
  - **high-resolution visualization** (it identifies important 'tiger cat' features like stripes, pointy ears and eyes)
  - **class-discriminative visualization** (it highlights the 'tiger cat' but not the 'boxer (dog)').
- Replacing Guided Backpropagation wit
  but Deconvolution visualizations have a
  was generally proved less noisy



(b) Guided Backprop 'Cat' (c) Grad-CAM 'Cat' (h) Guided Backprop 'Dog' (i) Grad-CAM 'Dog'

Gradients Class Activation Mapping (Grad-CAM)

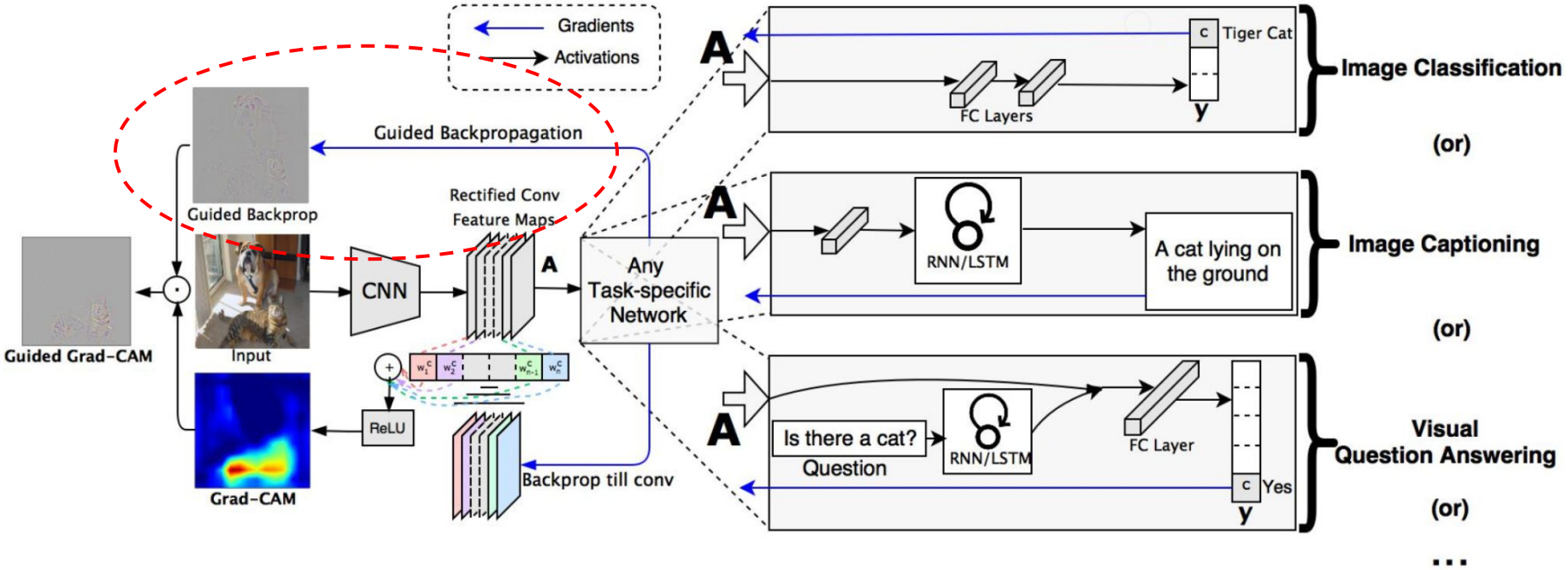


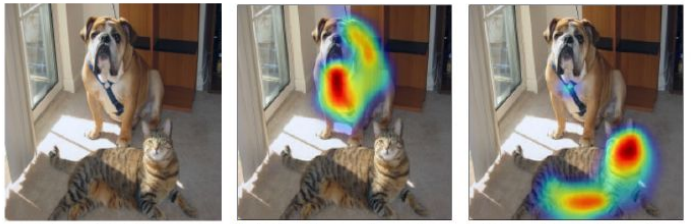
Fig. 2: Grad-CAM overview: Given an image and a class of interest (*e.g.*, 'tiger cat' or any other type of differentiable output) as input, we forward propagate the image through the CNN part of the model and then through task-specific computations to obtain a raw score for the category. The gradients are set to zero for all classes except the desired class (tiger cat), which is set to 1. This signal is then backpropagated to the rectified convolutional feature maps of interest, which we combine to compute the coarse Grad-CAM localization (blue heatmap) which represents where the model has to look to make the particular decision. Finally, we pointwise multiply the heatmap with guided backpropagation to get Guided Grad-CAM visualizations which are both high-resolution and concept-specific.

Gradients Class Activation Mapping (Grad-CAM)

Sometimes we aim **to understand** the input-output behavior of the deep network, which gives us the ability to improve it and denoise the input in some sense:

- Using a slight modification to Grad-CAM, we can obtain explanations that highlight support for regions that would make the network change its prediction. As a consequence, removing concepts occurring in those regions would make the model **more confident about its prediction**.
- We negate the gradient of y_c (score for class c) with respect to feature maps A of a convolutional layer. Thus the importance now become as stated below we take a weighted sum of the forward activation maps, A, with new weights, and follow it by a $ReLU(.)$ to obtain L_c.

$$\alpha_k^c = \overbrace{\frac{1}{Z}\sum_i\sum_j}^{\text{global average pooling}} \underbrace{-\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{Negative gradients}}$$



(a) Original Image (b) Cat Counterfactual exp (c) Dog Counterfactual exp

Gradients Class Activation Mapping (Grad-CAM)

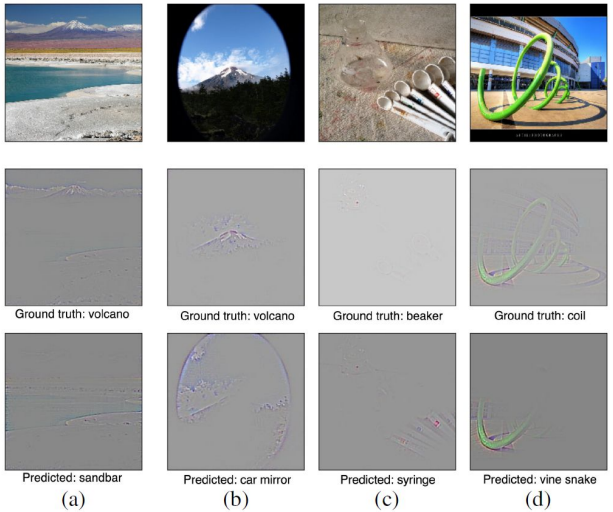


Fig. 6: In these cases the model (VGG-16) failed to predict the correct class in its top 1 (a and d) and top 5 (b and c) predictions. Humans would find it hard to explain some of these predictions without looking at the visualization for the predicted class. But with Grad-CAM, these mistakes seem justifiable.

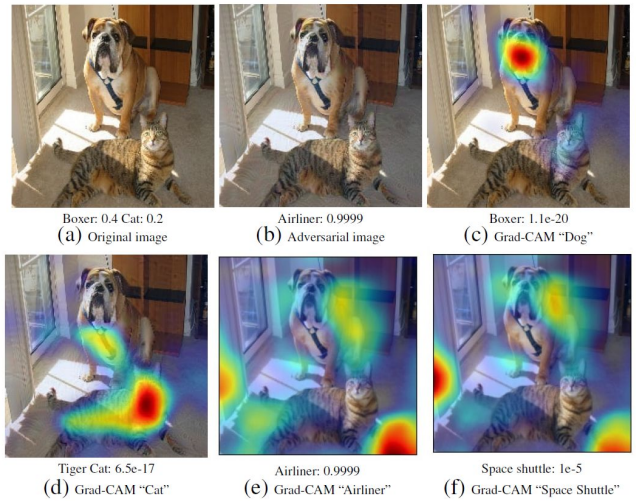


Fig. 7: (a-b) Original image and the generated adversarial image for category "airliner". (c-d) Grad-CAM visualizations for the original categories "tiger cat" and "boxer (dog)" along with their confidence. Despite the network being completely fooled into predicting the dominant category label of "airliner" with high confidence (>0.9999), Grad-CAM can localize the original categories accurately. (e-f) Grad-CAM for the top-2 predicted classes "airliner" and "space shuttle" seems to highlight the background.

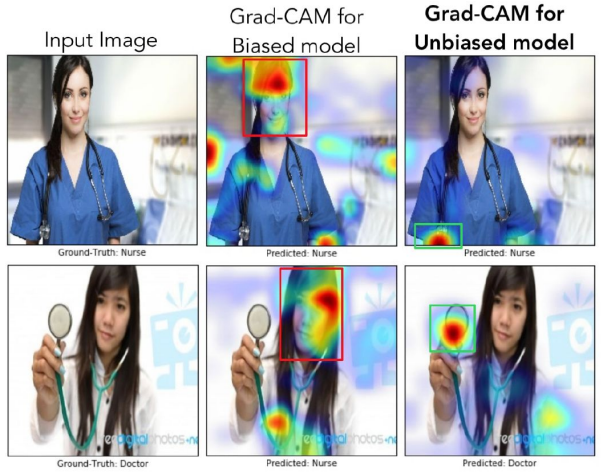


Fig. 8: In the first row, we can see that even though both models made the right decision, the biased model (model1) was looking at the face of the person to decide if the person was a nurse, whereas the unbiased model was looking at the short sleeves to make the decision. For the example image in the second row, the biased model made the wrong prediction (misclassifying a doctor as a nurse) by looking at the face and the hairstyle, whereas the unbiased model made the right prediction looking at the white coat, and the stethoscope.

Gradients Class Activation Mapping (Grad-CAM)

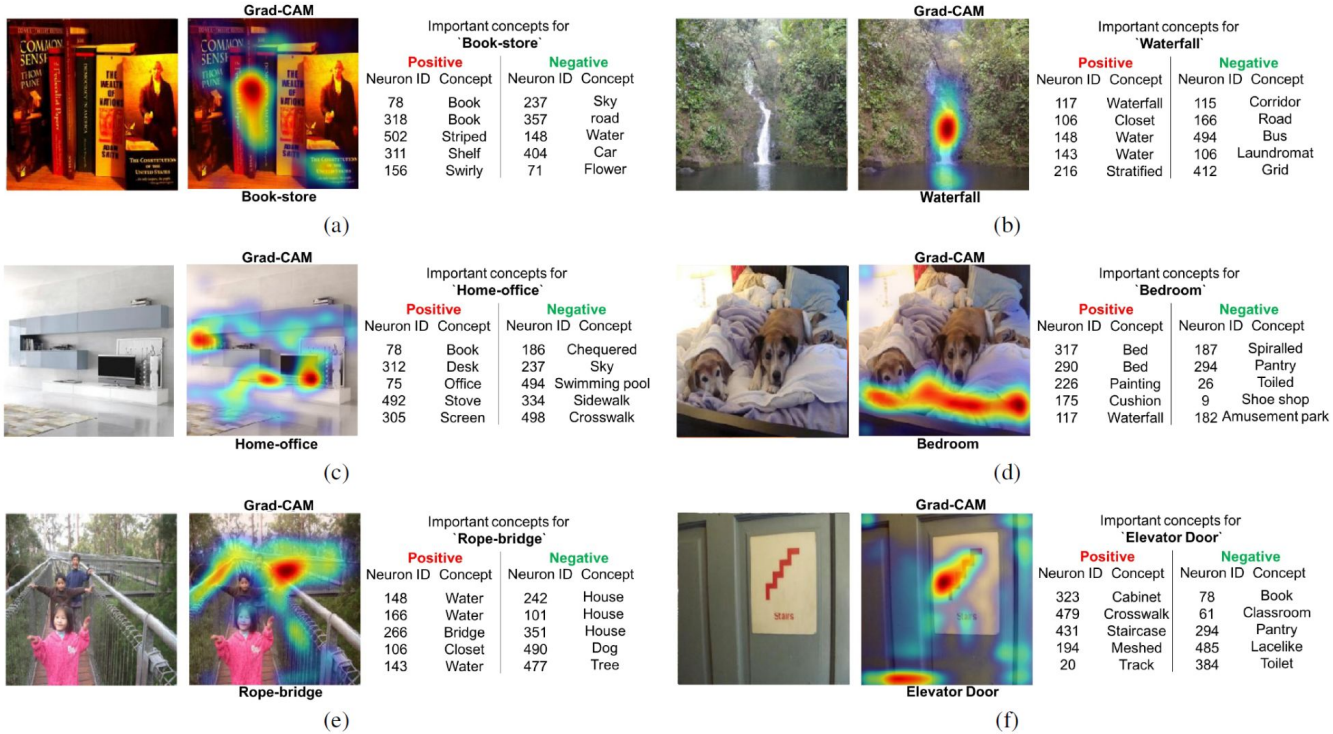


Fig. 9: Examples showing visual explanations and textual explanations for VGG-16 trained on Places365 dataset [61]. For textual explanations we provide the most important neurons for the predicted class along with their names. Important neurons can be either be persuasive (positive importance) or inhibitive (negative importance). The first 2 rows show success cases, and the last row shows 2 failure cases. We see that in (a), the important neurons computed by (1) look for concepts such as book and shelf which are indicative of class 'Book-store' which is fairly intuitive.

Gradients Class Activation Mapping (Grad-CAM)

- DenseCap consists of a Fully Convolutional Localization Network (FCLN) that produces bounding boxes for regions of interest and an LSTM-based language model that generates associated captions, all in a single forward pass. Using DenseCap, we generate 5 region specific captions per image with associated ground truth bounding boxes.
- Grad-CAM for a whole-image captioning model should localize the bounding box the region-caption was generated for. We may quantify this by computing the ratio of mean activation inside vs. outside the box.
  → Higher ratios are better because they indicate stronger attention to the region the caption was generated for.
- Grad-CAM is able to localize regions in the image that DenseCap describes, even the holistic captioning model was never trained with bounding-box annotations.

(b) Comparison to DenseCap

Gradients Class Activation Mapping (Grad-CAM)

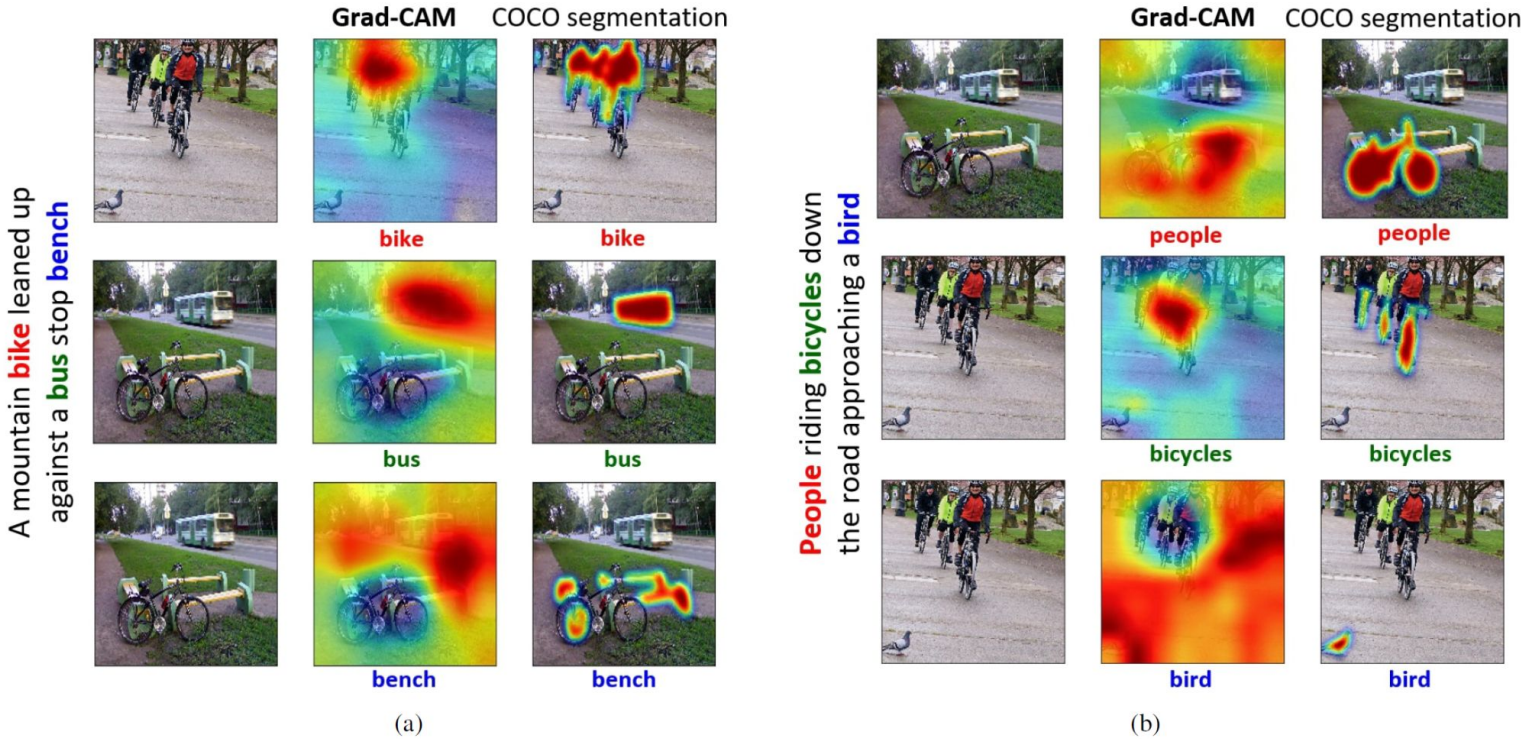


Fig. 11: Qualitative Results for our word-level captioning experiments: (a) Given the image on the left and the caption, we visualize Grad-CAM maps for the visual words “bike", “bench" and “bus". Note how well the Grad-CAM maps correlate with the COCO segmentation maps on the right column. (b) shows a similar example where we visualize Grad-CAM maps for the visual words “people", “bicycle" and “bird".

Randomized Input Sampling for Explanation

- Let $M: \Lambda \to \{0,1\}$ be a random binary mask with distribution $\mathcal{D}$. Consider the random variable $f(I \odot M)$, where $\odot$ denotes element-wise multiplication. We then define importance of pixel $\lambda \in \Lambda$, as the expected score over all possible masks M conditioned on the event that pixel λ is observed, i.e., $M(\lambda) = 1$:

$$S_{I,f}(\lambda) = \mathbb{E}_M \left[f(I \odot M) \mid M(\lambda) = 1 \right].$$

- The intuition behind this is that $f(I \odot M)$ is high when pixels preserved by mask M are important. This is obtained by the following steps:
  a. the image is masked by preserving only a subset of pixels.
  b. the confidence score for the masked image is computed by the black box.
- Summing over all masks $m: \Lambda \to \{0,1\}$

$$S_{I,f}(\lambda) = \sum_m f(I \odot m) P[M = m \mid M(\lambda) = 1]$$

(we want to avoid this cond. prob.)

Randomized Input Sampling for Explanation

- Using Bayes rule:

$$S_{I,f}(\lambda) = \sum_m f(I \odot m) P[M = m \mid M(\lambda) = 1]$$

$$= \frac{1}{P[M(\lambda) = 1]} \sum_m f(I \odot m) P[M = m,\ M(\lambda) = 1].$$

(now this is better!)

$$P[M = m,\ M(\lambda) = 1] = \begin{cases} 0, & \text{if } m(\lambda) = 0, \\ P[M = m], & \text{if } m(\lambda) = 1. \end{cases}$$

$$= m(\lambda) P[M = m].$$

$$S_{I,f}(\lambda) = \frac{1}{P[M(\lambda) = 1]} \sum_m f(I \odot m) \cdot m(\lambda) \cdot P[M = m].$$

Randomized Input Sampling for Explanation

- In matrix notation using $P[M(\lambda) = 1] = \mathbb{E}[M(\lambda)]$:

$$S_{I,f} = \frac{1}{\mathbb{E}[M]} \sum_{m} f\left(I \odot m\right) \cdot m \cdot P[M = m].$$

- Saliency map can be computed as a weighted sum of random masks, where weights are the (prior) probability scores, that masks produce, adjusted for the distribution of the random masks.
- To produce an importance map, explaining the decision of model f on image I:
  - we sample set of masks $\{M\}$ according to $\mathcal{D}$.
  - we probe the model by running it on masked images $(I \odot M)$.
  - we take the weighted average of the masks where the weights are the confidence scores $f(I \odot M)$ and normalize it by the expectation of any M:

$$S_{I,f}(\lambda) \overset{\text{MC}}{\approx} \frac{1}{\mathbb{E}[M] \cdot N} \sum_{i=1}^{N} f(I \odot M_i) \cdot M_i(\lambda).$$

Randomized Input Sampling for Explanation

- Problems with binary masks:
  a. Adversarial effects: a slight change in pixel values may cause significant variation in the model's confidence scores.
  b. Generating masks by independently setting their elements to zeros and ones will result in mask space of size $2^{H \times W}$.

- Some potential solutions:
  a. We sample smaller binary masks.
  b. We upsample small binary masks to larger resolution using bilinear interpolation. Bilinear upsampling does not introduce sharp edges in $I \odot M$ as well as results in a smooth importance map $\boldsymbol{S}$. After interpolation, masks M are no longer binary, but have values in $[0, 1]$.
  c. To allow more flexible masking, we shift all masks by a random number of pixels in both spatial directions.

Randomized Input Sampling for Explanation

The general “recipe” for training RISE:

1. Sample N binary masks of size $h \times w$ (smaller than image size $H \times W$) by setting each element independently to 1 with probability p and to 0 with the remaining probability 1 - p.
2. Upsample all masks to size $(h+1)C_H \times (w+1)C_W$ using bilinear interpolation, where $C_H \times C_W = \lfloor H/h \rfloor \times \lfloor W/w \rfloor$ is the size of the cell in the upsampled mask.
3. Crop areas $H \times W$ with uniformly random indents from $(0,0)$ up to (C_H, C_W).

Randomized Input Sampling for Explanation

- Suppose a base captioning model that models the probability of the next word given a partial sentence s up to this word and an input image I. Input sentence s can be any arbitrary sentence including the caption generated by the base model itself. Base output is: $f(I, s, w_k) = P[w_k | I, w_1, \dots w_{k-1}]$
- We probe the base model by running it on a set of n randomly masked inputs $f(I \odot M_i, s, w_k)$ and compute saliency $(N\mathbb{E}[M])^{-1} \sum_{i=1}^{n} f(I \odot M_i, s, w_k) \; \forall s.$

(a) "A horse and carriage on a city street."

(b) "A horse..."

(c) "A horse and carriage..."

(d) "White..."

Attribution for Deep Networks

- Formally, suppose we have a function $F: \mathbb{R}^n \rightarrow [0,1]$ that represents a deep network, and an input $x \in \mathbb{R}^n$. An **attribution** of the prediction at that input x relative to a baseline input x' is a vector $A_F(x, x') \in \mathbb{R}^n$ where its element a_i is the contribution of x_i to the prediction $F(x)$.
- The intention of this concept is to understand the input-output behavior of the deep network, which gives us the ability to improve it. In diagnostic captioning settings a DNN could help inform the doctor of the part of the image that resulted in the diagnosis.
- In a deep network, we model the absence of some cause using a single baseline input. For most deep networks, a natural baseline exists in the input space where the prediction is neutral.
  - In object recognition networks, the prediction is neutral in the black image.
  - When we assign blame to a certain cause (e.g. some pixels) we implicitly consider its absence as a baseline for comparing outcomes.

Attribution for Deep Networks

- Formally, suppose we have a function $F: \mathbb{R}^n \rightarrow [0,1]$ that represents a deep network, and an input $x \in \mathbb{R}^n$. An **attribution** of the prediction at that input x relative to a baseline input x' is a vector $A_F(x, x') \in \mathbb{R}^n$ where its element a_i is the contribution of x_i to the prediction $F(x)$.
- The intention of this concept is to understand the input-output behavior of the deep network, which gives us the ability to improve it. In diagnostic captioning settings a DNN could help inform the doctor of the part of the image that resulted in the diagnosis.
- In a deep network, we model the absence of some cause using a single baseline input. For most deep networks, a natural baseline exists in the input space where the prediction is neutral.
  - In object recognition networ
  - When we assign blame to a consider its absence as a b

$c_i = \text{'cat'}$

$c_i = ???$

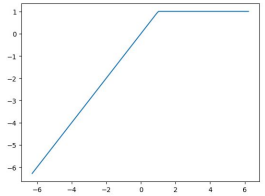
Attribution for Deep Networks

Let's identify two fundamental axioms for attribution methods:

1. **Sensitivity:**
   - An attribution method satisfies sensitivity if at least for every input x and baseline x' that differ in one feature i but have different predictions then the differing feature should be given a non-zero attribution $a_i \neq 0$.

$$\forall x, x' \text{ s.t. } \exists i\colon\ x_i \neq x_i', F(x_i) \neq F(x_i') \Rightarrow a_i \neq 0$$

   - Gradients violate sensitivity because the prediction function may flatten at the input x and thus have zero gradient despite the function value $F(x)$ at the input being different from that at the baseline.
   - Flattened function example: consider $F(x) = 1 - [1-x]_+$ (plotted). Suppose baseline is $x' = 0$ and the input is $x = 2$. As F becomes flat at $x = 1$, the gradient method gives attribution of 0 to x.



   - The lack of sensitivity causes gradients to focus on irrelevant features.
2. Implementation invariance.

Attribution for Deep Networks

Let's identify two fundamental axioms for attribution methods:

1. Sensitivity.
2. **Implementation invariance:**
   - A pair of NNs F_1, F_2 are *functionally equivalent* if their outputs are equal for all inputs, $F_1(x) = F_2(x) \ \forall x$ despite having very different implementations. If that is the case, *implementation invariance* holds if the attributions are also always identical.
   - DeepLift and LRP tackle the sensitivity issue by employing a baseline, and in some sense try to compute "discrete gradients" instead of gradients (that are instantaeneous in a sense) at the input but both break implementation invariance for this reason.
   - Chain rule does not hold, thus: $\frac{f(x_1)-f(x_0)}{g(x_1)-g(x_0)} \neq \frac{f(x_1)-f(x_0)}{h(x_1)-h(x_0)} \cdot \frac{h(x_1)-h(x_0)}{g(x_1)-g(x_0)}$
   - The attributions are potentially sensitive to unimportant aspects of a model.

Integrated Attribution for Deep Networks

- Formally, suppose we have a function $F: \mathbb{R}^n \rightarrow [0,1]$ that represents a deep network, and an input $x \in \mathbb{R}^n$. An **attribution** of the prediction at that input x relative to a baseline input x' is a vector $A_F(x, x') \in \mathbb{R}^n$ where its element a_i is the contribution of x_i to the prediction $F(x)$.
- Considering the straightline path from x' to x, integrated gradients are obtained by cumulating the gradients at all points along the path and are defined as the path integral of the gradients along it, for instance along the i^{th} dimension we compute the partial derivative of $F(x)$ w.r.t. x_i as:

$$\frac{\partial F(x)}{\partial x_i} = (x_i - x_i') \times \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha \times (x - x'))}{\partial x_i} \, d\alpha$$

- **Completeness:** the attributions add up to the difference $F(x)$ - $F(x')$ between the output at the input x, $F(x)$ and the baseline x', $F(x')$. This proves the method is somewhat comprehensive and should hold for $F: \mathbb{R}^n \rightarrow \mathbb{R}$ a differentiable function for most $x \in D_F$.

Integrated Attribution for Deep Networks

- Suppose a smooth function $\gamma: [0,1] \rightarrow \mathbb{R}^n$ specifying a path from the baseline x' to the input x. Then $\gamma(0) = x'$, $\gamma(1) = x$, whereas path integrated gradients are obtained along the path $\gamma(\alpha)$ for $\alpha \in [0, 1]$. Integrated gradients is a particular case of these *path methods*.

$$\frac{\partial F(x)}{\partial x_i} = \int_{\alpha=0}^{1} \frac{\partial F(\gamma(\alpha))}{\partial \gamma_i(\alpha)} \frac{\partial \gamma_i(\alpha)}{\partial \alpha} d\alpha$$

- There are several axioms that *path methods* **exclusively** satisfy:
  - **Sensitivity** (revisited)**:** if the function implemented by the deep network does not depend (mathematically) on some variable, then the attribution to that variable is always zero.
  - **Linearity:** if linearly compose two DNNs modeled by f_1, f_2 to form a third that models the function $f_3 = \alpha f_1 + \beta f_2$, the attributions for f_3 are the weighted sum of the attributions for f_1, f_2 with weights α and β respectively.
  - **Implementation Invariance.**

Integrated Attribution for Deep Networks

- Suppose a smooth function $\gamma: [0,1] \rightarrow \mathbb{R}^n$ specifying a path from the baseline x' to the input x. Then $\gamma(0) = x'$, $\gamma(1) = x$, whereas path integrated gradients are obtained along the path $\gamma(\alpha)$ for $\alpha \in [0, 1]$. Integrated gradients is a particular case of these *path methods*.

$$\frac{\partial F(x)}{\partial x_i} = \int_{\alpha=0}^{1} \frac{\partial F(\gamma(\alpha))}{\partial \gamma_i(\alpha)} \frac{\partial \gamma_i(\alpha)}{\partial \alpha} \, d\alpha$$

- There are several axioms that *path methods* ex
  - **Sensitivity (revisited):** if the function impl does not depend (mathematically) on som that variable is always zero.
  - **Linearity:** if linearly compose two DNNs m that models the function $f_3 = \alpha f_1 + \beta f_2$, the a sum of the attributions for f_1, f_2 with weights
  - **Implementation Invariance.**

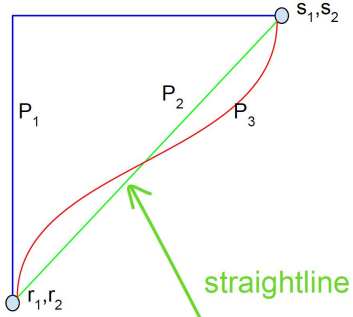


Figure 1. Three paths between an a baseline (r_1, r_2) and an input (s_1, s_2). Each path corresponds to a different attribution method. The path P_2 corresponds to the path used by integrated gradients.

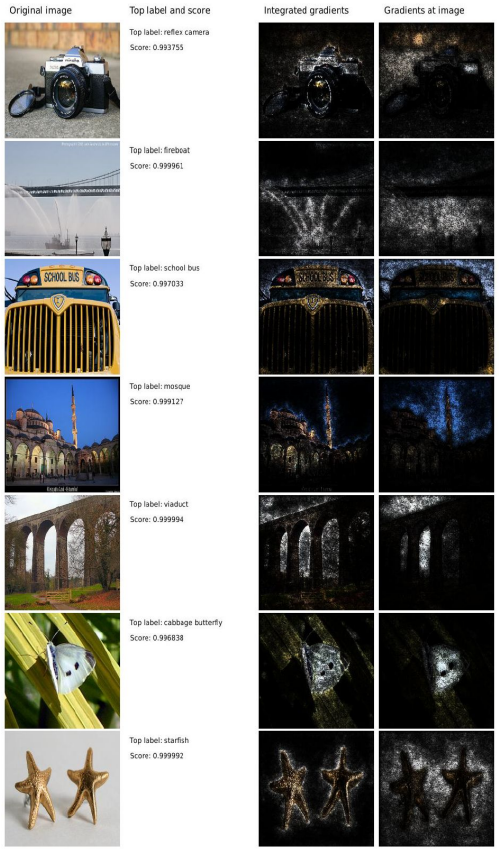
Integrated Attribution for Deep Networks

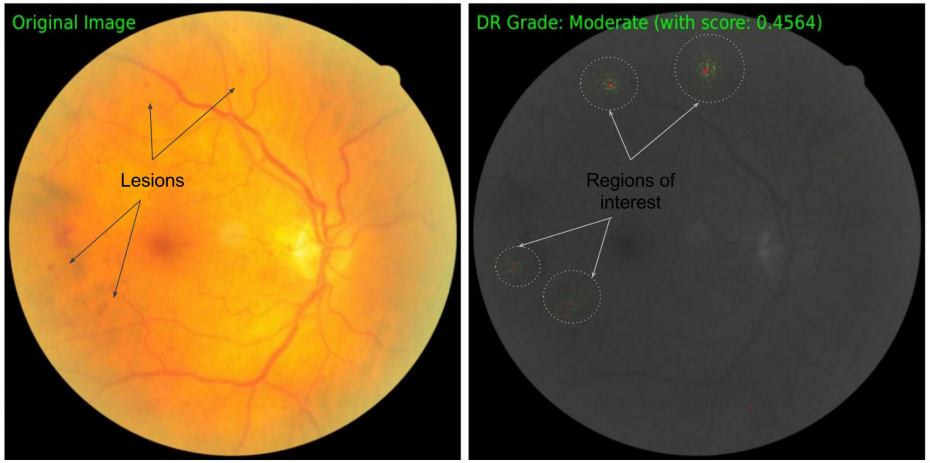
- Suppose a smooth function $\gamma: [0,1] \rightarrow \mathbb{R}^n$ specifying a path from the baseline x' to the input x. Then $\gamma(0) = x'$, $\gamma(1) = x$, whereas path integrated gradients are obtained along the path $\gamma(\alpha)$ for $\alpha \in [0, 1]$. Integrated gradients is a particular case of these *path methods*.

$$\frac{\partial F(x)}{\partial x_i} = \int_{\alpha=0}^{1} \frac{\partial F(\gamma(\alpha))}{\partial \gamma_i(\alpha)} \frac{\partial \gamma_i(\alpha)}{\partial \alpha} d\alpha$$

- There are several axioms that *path methods* **exclusively** satisfy (cont'ed):
  - **Symmetric variables property:** Two input variables x, y, are symmetric w.r.t. a function if swapping them does not change the function say F if and only if $F(x, y) = F(y, x) \ \forall x, y$.
  - **Symmetry perseverance:** An attribution method is symmetry preserving, if for all inputs x that have identical values for symmetric variables and for all baselines x' that have identical values for symmetric variables, these receive identical attributions.

Integrated Attribution for Deep Networks





Model goal: predict the severity grade for DR/Diabetic Retinopathy in retinal fundus images.

Integrated gradients aggregated in gray scale:

- in green we notice positive attributions
- in red we notice negative attributions

The **interior** of the lesions receive a negative attribution while the **periphery** receives a positive attribution indicating that the network focusses on the boundary of the lesion.

Integrated Attribution for Deep Networks

- **Question classification:** whether one could peek inside some machine learnt end-to-end model, provided there is enough training data, to derive new rules (like human-authored grammar rules). The goal of question classification is to identify the type of answer it is seeking.
- We use integrated gradients to attribute predictions down to the question terms in order to identify new trigger phrases for answer type. The baseline input is the all zero embedding vector.
  - The all zero input embedding vector is a good baseline.
  - The action of training causes unimportant words tend to have small norms, and so, in the limit, unimportance corresponds to the all-zero baseline.

$x' = [0, 0, 0, \ldots, 0]^{\mathrm{T}}$ ("meaningless" embedding *vs* context embeddings)

$$\ldots \quad \vec{e}_{i-1} = \begin{bmatrix} 1.8 \\ 2.3 \\ -1.4 \\ 3.7 \\ \ldots \\ -1.1 \end{bmatrix} \quad \vec{e}_i = \begin{bmatrix} 2.4 \\ -3 \\ 9.3 \\ 5.1 \\ \ldots \\ 3.9 \end{bmatrix} \quad \vec{e}_{i+1} = \begin{bmatrix} 2.2 \\ 3.8 \\ 1.2 \\ -6.4 \\ \ldots \\ 7.1 \end{bmatrix} \quad \ldots$$

- Attributions present several nice properties.

Integrated Attribution for Deep Networks

- **Question classification:** whether one could peek inside some machine learnt end-to-end model, provided there is enough training data, to derive new rules (like human-authored grammar rules). The goal of question classification is to identify the type of answer it is seeking.
- We use integrated gradients to attribute predictions down to the question terms in order to identify new trigger phrases for answer type. The baseline input is the all zero embedding vector.
- Attributions present several nice properties.

how many townships have a population above 50 ? [prediction: NUMERIC]
what is the difference in population between fora and masilo [prediction: NUMERIC]
how many athletes are not ranked ? [prediction: NUMERIC]
what is the total number of points scored ? [prediction: NUMERIC]
which film was before the audacity of democracy ? [prediction: STRING]
which year did she work on the most films ? [prediction: DATETIME]
what year was the last school established ? [prediction: DATETIME]
when did ed sheeran get his first number one of the year ? [prediction: DATETIME]
did charles oakley play more minutes than robert parish ? [prediction: YESNO]

Color indicates attribution strength or predictions (black):

- Red is positive,
- Blue is negative,
- Gray is neutral.

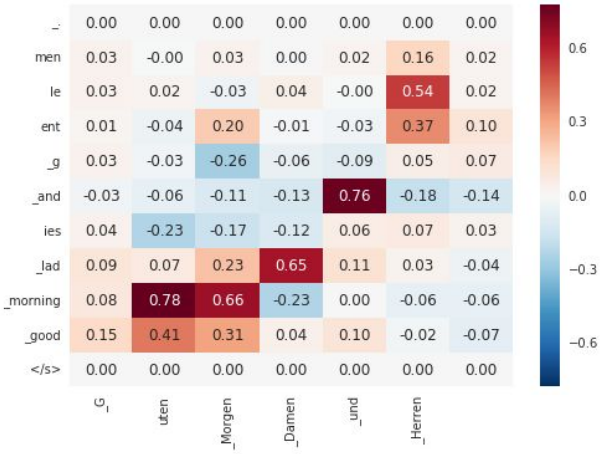
Integrated Attribution for Deep Networks

- **Question classification:** whether one could peek inside some machine learnt end-to-end model, provided there is enough training data, to derive new rules (like human-authored grammar rules). The goal of question classification is to identify the type of answer it is seeking.
- We use integrated gradients to attribute predictions down to the question terms in order to identify new trigger phrases for answer type. The baseline input is the all zero embedding vector.
- Attributions present several nice properties:
  a. Attributions largely agree with commonly used rules, for e.g., “how many” indicates a numeric seeking question.
  b. Attributions help identify novel question classification rules, e.g., questions containing “total number” are seeking numeric answers.
  c. Attributions also point out undesirable correlations, for e.g., “charles” is used as trigger for a yes/no question.

Integrated Attribution for Deep Networks

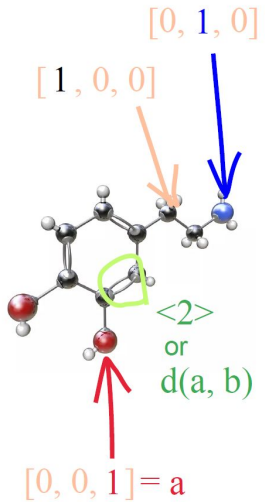
- **Question classification:** whether one could peek inside some machine learnt end-to-end model, provided there is enough training data, to derive new rules (like human-authored grammar rules). The goal of question classification is to identify the type of answer it is seeking.
- We use integrated gradients to attribute predictions down to the question terms in order to identify new trigger phrases for answer type. The baseline input is the all zero embedding vector.
- Attributions present several nice properties:
  a. Attributions largely agree with commonly used r
  indicates a numeric seeking question.
  b
  c. Attributions also point out undesirable correlatio
  used as trigger for a yes/no question.

They also yield meaningful associations!



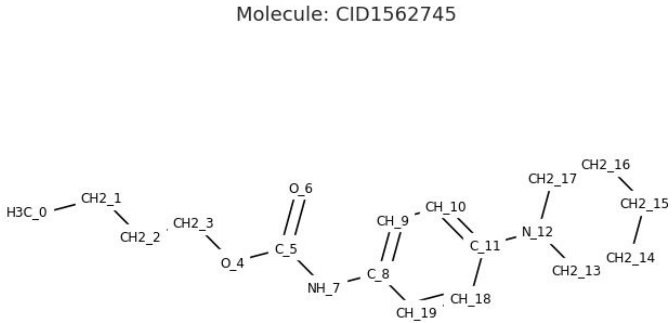
Integrated Attribution for Deep Networks

- **Next problem:** predicting whether an input molecule is active against a certain target (e.g., protein or enzyme) considering a network based on the molecular graph convolution architecture.
- The CNN requires an input molecule to be encoded by hand as a set of atom and atom-pair features describing the molecule as an undirected graph. Follow this particular model convention:
  - Atoms are featurized using a one-hot encoding specifying the atom type (e.g., C, O, S, etc.), and
  - atom-pairs are featurized by specifying either the type of bond (e.g., single, double, triple, etc.) between the atoms, or the graph distance between them.
- Then visualize integrated gradients as heatmaps over the atom and atom-pair features with the heatmap intensity depicting the strength of the contribution.



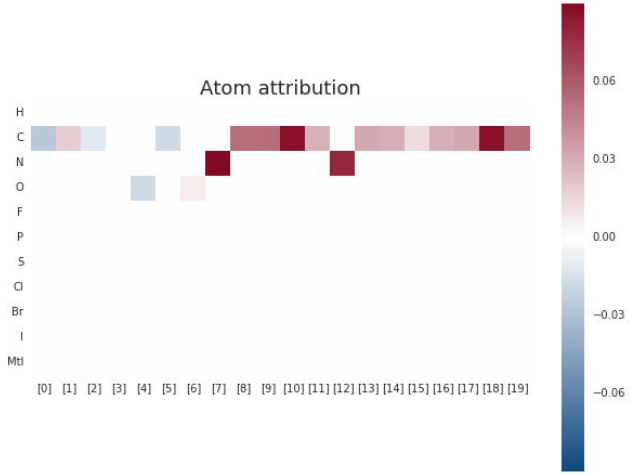
Integrated Attribution for Deep Networks

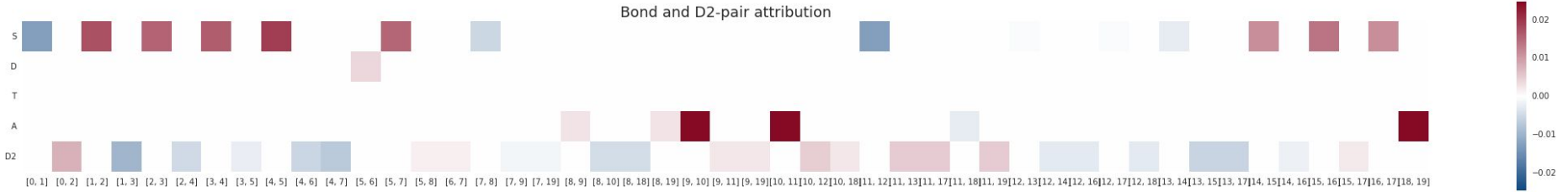
Attribution for a molecule under the W2N2 network. The molecules are active on task PCBA-58432.



Attribution summary

Softmax score for task PCBA-588342: 0.98
Atom attribution: 0.62 (63%)
Bond attribution: 0.45 (46%)
D2-pair attribution: -0.03 (-3%)





Additive feature attribution methods

- Let f be some prediction model to be explained and g the explanation model. We term as *interpretable model* any interpretable approximation of the original model. Explanation models typically use simplified inputs x' that map to the original inputs through a mapping function $x = h_x(x')$.
- Let's also focus on local methods designed to explain a prediction $f(x)$ based on a single input x. Local methods will try to ensure $g(z') \approx f(h_x(z'))$ whenever $z' \approx x'$. Note that $x = h_x(x')$ even though x' may contain less information than x because h_x is specific to the current input x.
- For these methods we derive: $g(z') = \varphi_0 + \sum_{i=1}^{M} \varphi_i z_i' , z' \in \{0,1\}^M, M$ and $\varphi_i \in \mathbb{R}$

  → Essentially, additive feature attribution (explanation) methods attribute an effect φ_i to each feature, and summing the effects of all feature attributions approximates the output $f(x)$ of the original model. Many current methods match this equation.

Additive feature attribution methods

- **LIME:** Uses the local linear explanation model and is an additive feature attribution method. LIME refers to simplified inputs x' as "interpretable inputs," and the mapping $x = h_x(x')$ converts a binary vector of interpretable inputs into the original input space.
  - Cost function to minimize: $\xi = \arg\min_{g \in \mathcal{G}} L(f, g, \pi_{x'}) + \Omega(g)$
  - Applications: BOW, CV, ...
- **DeepLIFT** attributes to each input x_i a value, $C_{\Delta x_i \Delta y}$, that represents the effect of that input being set to a reference value as opposed to its original value. The mapping $x = h_x(x')$ now converts binary values into the original inputs, where 1 indicates that an input takes its original value, and 0 indicates that it takes the reference value.

  "Summation-to-delta" property: $\Delta o = \sum_{i=1}^{n} C_{\Delta x_i \Delta o}$

  Set: $\varphi_0 = r_0$ and $\varphi_i = C_{\Delta x_i \Delta_o}$

$o = f(x)$ is the model output, $\Delta o = f(x) - f(r)$, $\Delta x_i = x_i - r_i$, and r is the reference input.

Additive feature attribution methods

- **LIME:** Uses the local linear explanation model and is an additive feature attribution method. LIME refers to simplified inputs x' as "interpretable inputs," and the mapping $x = h_x(x')$ converts a binary vector of interpretable inputs into the original input space.
  - Cost function to minimize: $\xi = \arg\min_{g \in \mathcal{G}} L(f, g, \pi_{x'}) + \Omega(g)$
  - Applications: BOW, CV, ...
- **DeepLIFT** a… ents the effect of that input… ginal value. The mapping x = … puts, where 1 indicates th… at it takes the reference va… model output, "Summation…), $\Delta x_i = x_i - r_i$, and …ence input.

Set: $\varphi_0 = r_0$ …

Explanation of LIME cost function

Faithfulness of the explanation model $g(z')$ to the original model $f(h_x(z'))$ is enforced through loss L over a set of samples $\{x_i\}_i$ in the simplified input space weighted by the local kernel $\pi_{x'}$. Then Ω penalizes the complexity of g.

Additive feature attribution methods

- **LIME:** Uses the local linear explanation model and is an additive feature attribution method. LIME refers to simplified inputs x' as "interpretable inputs," and the mapping $x = h_x(x')$ converts a binary vector of interpretable inputs into the original input space.
  - Cost function to minimize: $\xi = \arg\min_{g \in \mathcal{G}} L(f, g, \pi_{x'}) + \Omega(g)$
  - Applications: BOW, CV, ...
- **DeepLIFT** attributes to each input x_i a value, $C_{\Delta x_i \Delta y}$, that represents the effect of that input being set to a reference value as opposed to its original value. The mapping $x = h_x(x')$ now converts binary values into the original inputs, where 1 indicates that an input takes its original value, and 0 indicates that it takes the reference value.

  "Summation-to-delta" property: $\Delta o = \sum_{i=1}^{n} C_{\Delta x_i \Delta o}$

  Set: $\varphi_0 = r_0$ and $\varphi_i = C_{\Delta x_i \Delta_o}$.

$o = f(x)$ is the model output, $\Delta o = f(x) - f(r)$, $\Delta x_i = x_i - r_i$, and r is the reference input.

Additive feature attribution methods

- **LRP:** Equivalent to DeepLIFT with the reference activations of all neurons fixed to zero. Thus, $x = h_x(x')$ now converts binary values into the original input space, where 1 means that an input takes its original value, and 0 means an input takes the 0 value.
- **Classic Shapley Value Estimation:**
  - *Sharpley regression values.*
  - *Sharpley sampling values.*
  - *Quantitative input influence.*

- All additive feature attribution methods have several desirable properties.
  a. Local accuracy.
  b. Missingness.
  c. Consistency.

Additive feature attribution methods

- **Classic Shapley Value Estimation:**
  - *Sharpley regression values:* Feature importance values for linear models in the presence of multicollinearity. Requires training the model on all feature subsets $S \subseteq F_{\text{all}} \backslash \{i\}$, where F_{all} is the set of all features.
    - A model $f_{S \cup \{i\}}$ is trained with that feature present, and another model f_S is trained with the feature withheld.
    - Predictions from the two models are compared on the current input that is $f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_S(x_S)$, where x_S represents the input features $x_S \in S$. Attributions (differences) are computed for each possible subset before contributing to a weighted average φ_i.

$$\varphi_i = \sum_{S \subseteq F_{\text{all}} \{i\}} \frac{|S|!\,(|F_{\text{all}}| - |S| - 1)!}{|F_{\text{all}}|!} \left[f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_S(x_S) \right]$$

  - *Sharpley sampling values.*
  - *Quantitative input influence.*

Additive feature attribution methods

- **Classic Shapley Value Estimation:**
  - *Sh… …r models in the… …all feature su…*
    - …model f_S is …
    - …nput that is …S.
    - …t before contributing to a weighted average φ_i.

Interpretation of h_x for *Sharpley regression values*

The function h_x maps 1 or 0 to the original input space, where 1 indicates the input is included in the model, and 0 indicates exclusion from the model. If we let $\varphi_0 = f_\varnothing(\varnothing)$, then the Shapley regression values match the definition for the general case $g(z')$.

$$\varphi_i = \sum_{S \subseteq F_{\text{all}}\{i\}} \frac{|S|!\,(|F_{\text{all}}| - |S| - 1)!}{|F_{\text{all}}|!} \left[f_{S\cup\{i\}}(x_{S\cup\{i\}}) - f_S(x_S) \right]$$

  - *Sharpley sampling values.*
  - *Quantitative input influence.*

Additive feature attribution methods

- **Classic Shapley Value Estimation:**
  - *Sharpley regression values.*
  - *Sharpley sampling values:*
    - Apply sampling approximations to weighted averages φ_i.
    - Approximate the effect of removing a variable from the model by integrating over samples from the training dataset $S \in \mathcal{D}$.
    - No need for retraining the model for every $S \subseteq F_{\text{all}} \backslash \{i\}$.
    - Less than $2^{|S|}$ differences to be computed.
  - *Quantitative input influence:*
    - Broader framework that addresses more than feature attributions.
    - However, as part of its method it independently proposes a sampling approximation to Shapley values that is nearly identical to Shapley sampling values.

  All the aforementioned are additive feature attribution methods.

Additive feature attribution methods

- **Classic Shapley Value Estimation:**
  - *Sharpley regression values.*
  - *Sharpley sampling values:*
    - Apply sampling approximations to weighted averages φ_i.
    - Approximate the effect of removing a variable from the model by integrating over samples from the training dataset $\mathcal{S} \in \mathcal{D}$.
    - No need for retraining the model for every $S \subseteq F_{\text{all}} \setminus \{i\}$.
    - Less than $2^{|S|}$ differences to be computed.
  - *Quantitative input influence:*
    - Broader framework that addresses more than feature attributions.
    - However, as part of its method it independently proposes a sampling approximation to Shapley values that is nearly identical to Shapley sampling values.

  All the aforementioned are additive feature attribution methods.

Additive feature attribution methods

- All additive feature attribution methods have several desirable properties.
  a. Local accuracy.
  When approximating the original model f for a specific input x, $f(x)$, local accuracy requires the explanation model to at least match the output of $f(x)$ for the simplified input x', $g(x')$ (which corresponds to the original input x).

  Recall per definition: $g(z') = \varphi_0 + \sum_{i=1}^{M} \varphi_i z_i', z' \in \{0,1\}^M, M$ and $\varphi_i \in \mathbb{R}$

  b. Missingness.
  If the simplified inputs represent feature presence, missingnes requires features missing in the original input to have no impact.

  $$x_i' = 0 \Longrightarrow \varphi_i = 0$$

  c. Consistency.

Additive feature attribution methods

- All additive feature attribution methods have several desirable properties.
  a. Local accuracy.
     When approximating the original model f for a specific input x, $f(x)$, local accuracy requires the explanation model to at least match the output of $f(x)$ for the simplified input x', $g(x')$ (which corresponds to the original input x).
     Recall per definition: $g(z') = \varphi_0 + \sum_{i=1}^{M} \varphi_i z_i', z' \in \{0,1\}^M, M$ and $\varphi_i \in \mathbb{R}$
  b. Missingness.
     If the simplified inputs represent feature presence, missingnes requires features missing in the original input to have no impact.

$$x_i' = 0 \Rightarrow \varphi_i = 0$$

  c. Consistency.

Additive feature attribution methods

- All additive feature attribution methods have several desirable properties.
  a. Local accuracy.
  b. Missingness.
  c. Consistency.

  Let $f_x(z') = f(h_x(z'))$ and $z' \setminus i$ denote setting $z' = 0$. For any two models f, f':

  $$f_x'(z') = f_x'(z' \setminus i) \geq f_x(z') = f_x(z' \setminus i) \;\forall\; z' \in \{0, 1\}^M \Rightarrow \varphi_i(f', x) \geq \varphi_i(f, x)$$

- **Theorem:** The only possible explanation model g that follows the definition and the aforementioned properties is described by $\varphi_i(f, x)$, where $|z'|$ is the number of non-zero entries in z', and $z' \subseteq x'$ represents all z' vectors where the non-zero entries are a subset of the non-zero entries in x'.

$$\varphi_i(f, x) = \sum_{z' \subseteq x'} \frac{|z'|!\,(M - |z'| - 1)!}{M!} [f_x(z') - f_x(z' \setminus i)]$$

Additive feature attribution methods

- All additive feature attribution methods have several desirable properties.
  a. Local accuracy.
  b. Missingness.
  c. Consistency.

  Let $f_x(z') = f(h_x(z'))$ and $z' \setminus i$ denote setting $z' = 0$. For any two models f, f':

  $$f_x'(z') = f_x'(z' \setminus i) \geq f_x(z') = f_x(z' \setminus i) \;\forall\; z' \in \{0, 1\}^M \Rightarrow \varphi_i(f', x) \geq \varphi_i(f, x)$$

- **Theorem:** The only possible explanation model g that follows the definition and the aforementioned properties is described by $\varphi_i(f, x)$, where $|z'|$ is the number of non-zero entries in z', and $z' \subseteq x'$ represents all z' vectors where the non-zero entries are a subset of the non-zero entries in x'.

$$\varphi_i(f, x) = \sum_{z' \subseteq x'} \frac{|z'|!\,(M - |z'| - 1)!}{M!} [f_x(z') - f_x(z' \setminus i)]$$

SHAP (SHapley Additive exPlanation) values

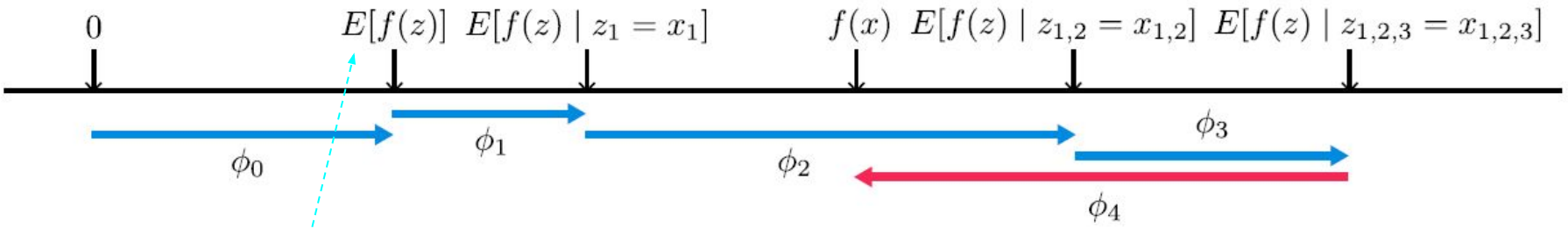


Figure 1: SHAP (SHapley Additive exPlanation) values attribute to each feature the change in the expected model prediction when conditioning on that feature. They explain how to get from the base value $E[f(z)]$ that would be predicted if we did not know any features to the current output $f(x)$. This diagram shows a single ordering. When the model is non-linear or the input features are not independent, however, the order in which features are added to the expectation matters, and the SHAP values arise from averaging the ϕ_i values across all possible orderings.

SHAP (SHapley Additive exPlanation) values

- SHAP values provide the unique **additive feature importance** measure that adheres to all properties and uses conditional expectations to define simplified inputs. They are obtained by setting $f_x(z') = f(h_x(z')) = \mathbb{E}[f(z) \mid z_S]$, where S is the set of non/zero indices in z'.
- Implicit in this definition of SHAP values is a simplified input mapping, $h_x(z') = z_S$, where z_S has **missing values for features** not in the set S. Since most models cannot handle arbitrary patterns of **missing input values**, we approximate $f(z_S)$ with $\mathbb{E}[f(z) \mid z_S]$. This definition of SHAP values is designed to closely align with the Classic Shapley Value Estimation.
  - Model-agnostic approximation methods:
    - Shapley sampling values
    - Kernel SHAP.
  - Model-type-specific approximation methods:
    - Max SHAP
    - Deep SHAP

Assuming model independence and feature linearity we get:

$$
\begin{aligned}
f(h_x(z')) &= E[f(z) \mid z_S] \\
&= E_{z_{\bar{S}} \mid z_S}[f(z)] \\
&\approx E_{z_{\bar{S}}}[f(z)] \quad \text{model ind.} \\
&\approx f([z_S, E[z_{\bar{S}}]]). \quad \text{feat. lin.}
\end{aligned}
$$

SHAP (SHapley Additive exPlanation) values

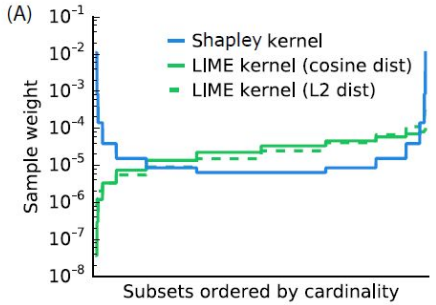
- Remember LIME objective: $\xi = \arg\min_{g \in \mathcal{G}} L(f, g, \pi_{x'}) + \Omega(g)$
- Since linear LIME is an additive feature attribution method, the Shapley values are the only possible solution to this minimization although it looks dissimilar to the desired explanation model g. Whether the solution ξ recovers the Sharpley values depends on the choice of loss function L, weighting kernel $\pi_{x'}$ and the regularization term Ω.
- To guarantee **convergence to Sharpley values** we may not choose LIME parameters heuristically; but use the loss function L, weighting kernel $\pi_{x'}$, and the regularization term Ω defined by a **Sharpley Kernel**.

$$\Omega(g) = 0,$$

$$\pi_{x'}(z') = \frac{(M-1)}{(M \text{ choose } |z'|)|z'|(M-|z'|)},$$

$$L(f, g, \pi_{x'}) = \sum_{z' \in Z} \left[f(h_x^{-1}(z')) - g(z') \right]^2 \pi_{x'}(z')$$

(assuming $|z'|$ is the number of non-zero elements in z')



SHAP (SHapley Additive exPlanation) values

- Remember LIME objective: $\xi = \arg\min_{g \in G} L(f, g, \pi_{x'}) + \Omega(g)$
- Since I… ley values are the… ssimilar to the des… Sharpley values … and the regular…
- To gua… LIME parame… $\pi_{x'}$, and the regularization term Ω defined by a **Sharpley Kernel**.

Interpretation of the Shapley kernel choice

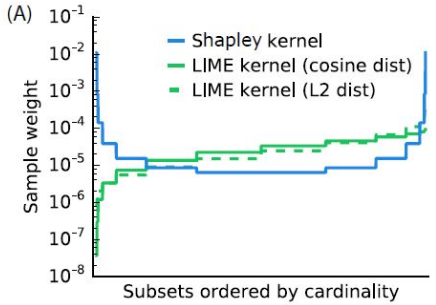
Compared to previous heuristically chosen kernels, since the mean is also the best least squares point estimate for a set of data points, it is natural to search for a weighting kernel that causes linear least squares regression to recapitulate the Shapley values.

$$\Omega(g) = 0,$$

$$\pi_{x'}(z') = \frac{(M-1)}{(M \text{ choose } |z'|)|z'|(M-|z'|)},$$

$$L(f, g, \pi_{x'}) = \sum_{z' \in Z} \left[f(h_x^{-1}(z')) - g(z') \right]^2 \pi_{x'}(z')$$

(assuming $|z'|$ is the number of non-zero elements in z')



SHAP (SHapley Additive exPlanation) values

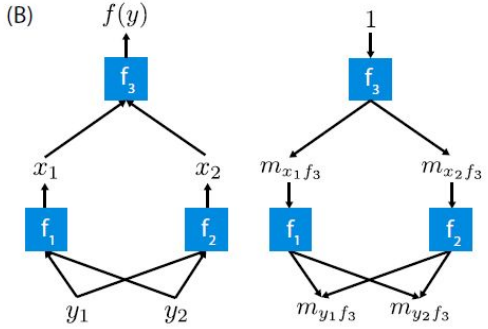
- **Linear SHAP**
  For linear models, if we assume input feature independence, SHAP values can be approximated directly from the model's weight coefficients:

$$f(x) = \sum_{j=1}^{M} w_j x_j + b \Rightarrow \varphi_0(f, x) = b, \varphi_i(f, x) = w_j\left(x_j - \mathbb{E}[x_j]\right)$$

- **Low-Order SHAP**
  Since linear regression using Shapley kernel has complexity $O(2^M + M^3)$, it is efficient for small values of M if we choose an approximation of the conditional expectations assuming model independence and feature linearity.
- **Max-SHAP**
  Using a permutation formulation of Shapley values, we can calculate the probability that each input will increase the maximum value over every other input. Doing this on a sorted order of input values lets us compute the Shapley values of a max function with M inputs in $O(M^2)$ time instead of $O(M2^M)$.

SHAP (SHapley Additive exPlanation) values

- DeepLIFT approximates SHAP values assuming that the i**nput features are independent of one another and the deep model is linear**; using its linear composition rule, which is equivalent to linearizing the non-linear components of a neural network.
- Its back-propagation rules defining how each component is linearized are intuitive but were heuristically chosen; however that Shapley values represent the only attribution values that satisfy consistency. DeepLIFT could become a compositional approximation of SHAP values.
- **Deep SHAP:** It embodies this idea by combining SHAP values computed for smaller components of the network into SHAP values for **the whole network *f***. It recursively passes DeepLIFT's multipliers, defined in terms of SHAP values, backwards through the network.



$$f_3(y) = f_1(y_1) + f_2(y_2), y = \{y_1, y_2\}$$

SHAP (SHapley Additive exPlanation) values

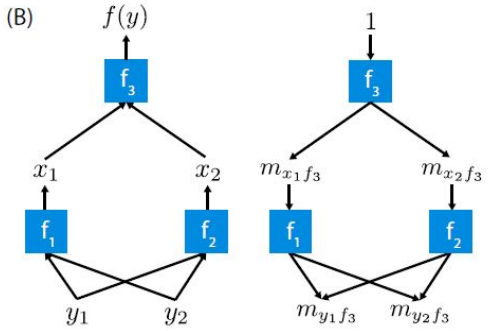
- DeepLIFT a… **features are independe**… sing its linear composition… r components of a neural…
- Its back-pro… arized are intuitive but… lues represent the only att… uld become a composition…

Explanation of DeepLIFT's multipliers

Compositional models such as deep neural networks are comprise of simple components. Given analytic solutions for the Shapley values of these, fast approximations for the model can be made using DeepLIFT's back-propagation.

- **Deep SHAP:** It embodies this idea by combining SHAP values computed for smaller components of the network into SHAP values for **the whole network** f. It recursively passes DeepLIFT's multipliers, defined in terms of SHAP values, backwards through the network.

$$f_3(y) = f_1(y_1) + f_2(y_2), y = \{y_1, y_2\}$$



SHAP (SHapley Additive exPlanation) values

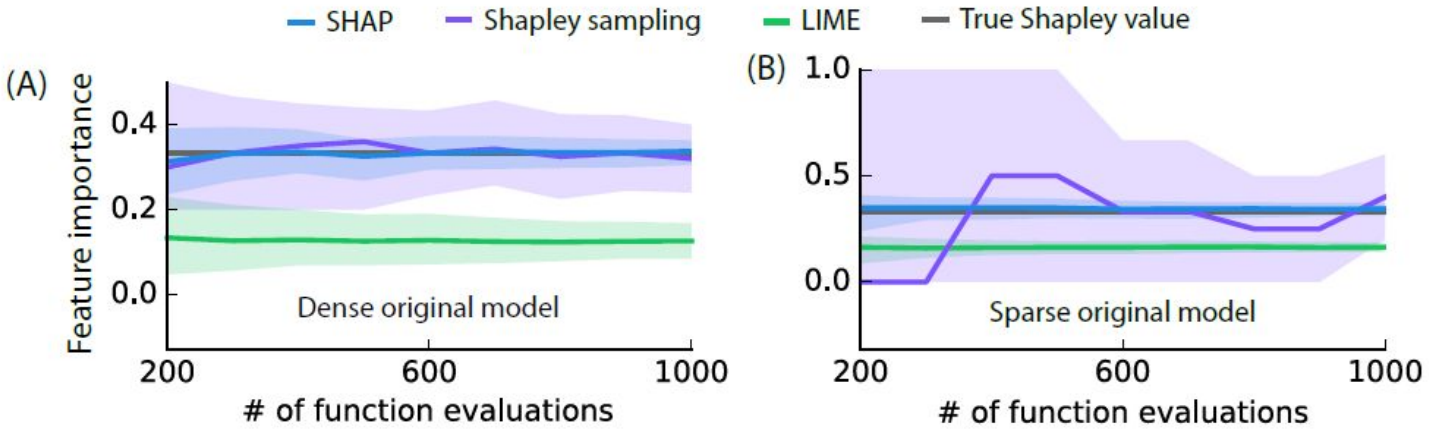


Figure 3: Comparison of three additive feature attribution methods: Kernel SHAP (using a debiased lasso), Shapley sampling values, and LIME (using the open source implementation). Feature importance estimates are shown for one feature in two models as the number of evaluations of the original model function increases. The 10th and 90th percentiles are shown for 200 replicate estimates at each sample size. (A) A decision tree model using all 10 input features is explained for a single input. (B) A decision tree using only 3 of 100 input features is explained for a single input.

SHAP (SHapley Additive exPlanation) values

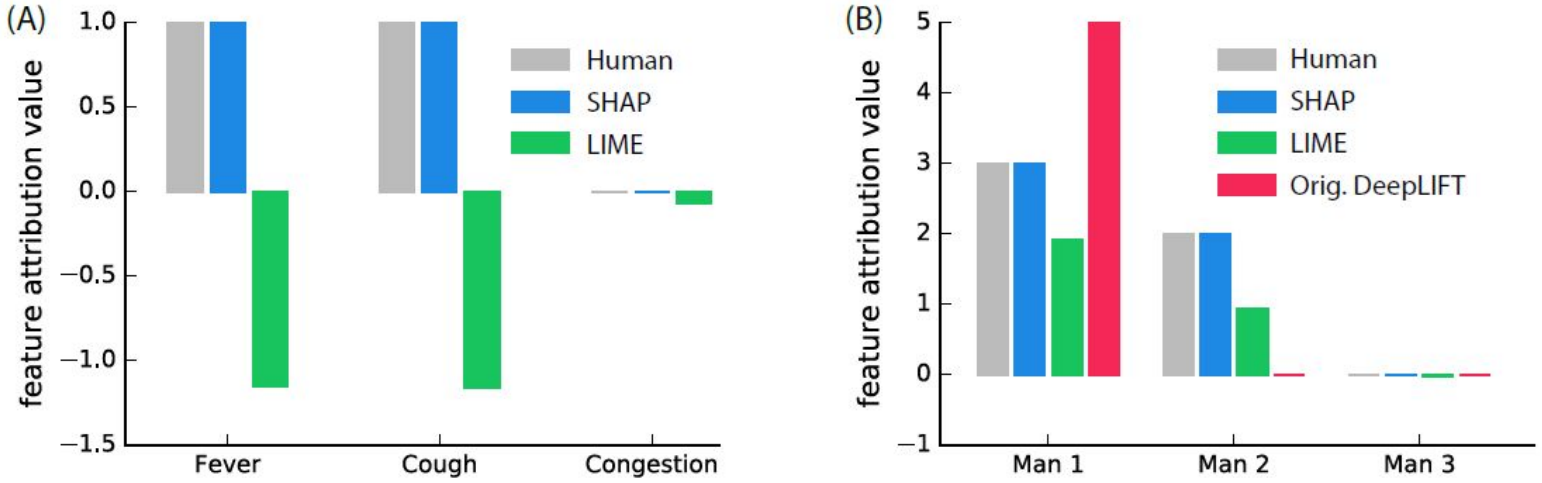


Figure 4: Human feature impact estimates are shown as the most common explanation given among 30 (A) and 52 (B) random individuals, respectively. (A) Feature attributions for a model output value (sickness score) of 2. The model output is 2 when fever and cough are both present, 5 when only one of fever or cough is present, and 0 otherwise. (B) Attributions of profit among three men, given according to the maximum number of questions any man got right. The first man got 5 questions right, the second 4 questions, and the third got none right, so the profit is $5.

SHAP (SHapley Additive exPlanation) values

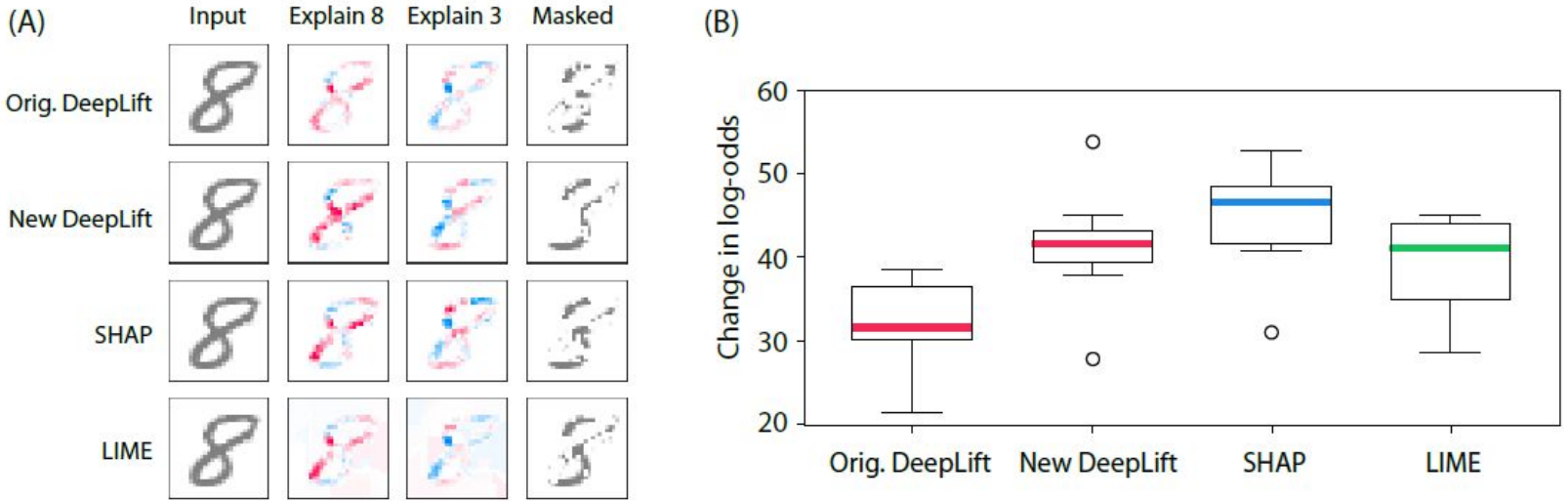


Figure 5: Explaining the output of a convolutional network trained on the MNIST digit dataset. Orig. DeepLIFT has no explicit Shapley approximations, while New DeepLIFT seeks to better approximate Shapley values. (A) Red areas increase the probability of that class, and blue areas decrease the probability. Masked removes pixels in order to go from 8 to 3. (B) The change in log odds when masking over 20 random images supports the use of better estimates of SHAP values.