Training with Explanations Alone

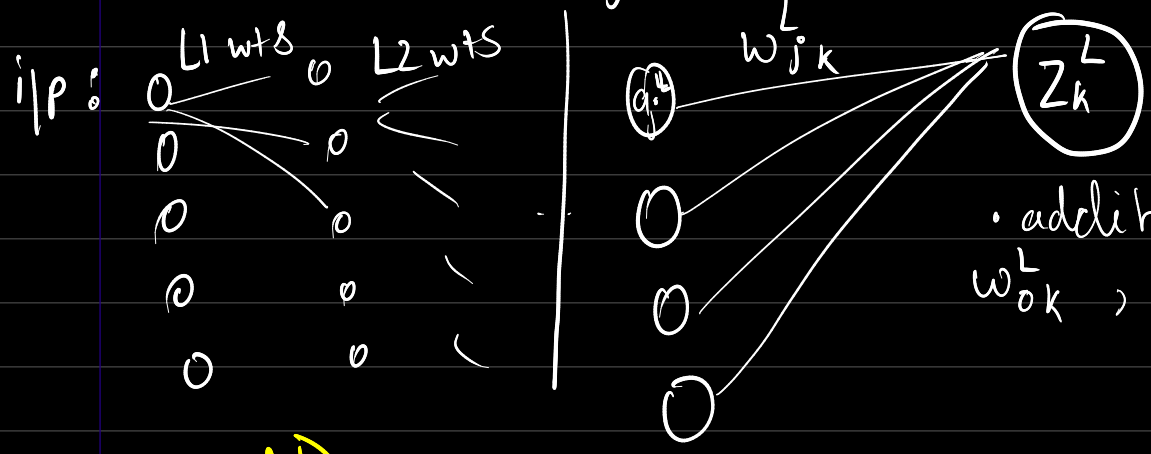
Appendix A

A.1 LRP fundamentals and tracing back bias

(1) (A2)

$$z_k^L = \sum_j w_{jk}^L a_j^L$$

- z_k^L is the output of the k^{th} neuron in the L^{th} fully connected layer BEFORE ReLU activation.
  - ↳ z_k^L is a function of the outputs of neurons j in the previous layer: a_j^L



- additive bias is w_{ok}^L, $a_o^L = 1$.

(2) (A1)

$$R_j^{L-1} = \sum_k \frac{w_{jk}^L a_j^L}{z_k^L + \text{sign}(z_k^L)\epsilon} R_k^L$$

- LRP-ϵ relevance back prop rule used to propogate relevance from neuron k in layer L to neurons j in layer $L-1$.

- R_k^L denotes how relevant the neuron k in layer L is to the output of the neural network (logits?)

* LRP back prop begins at one of the neurons in the final output layer say neuron c in layer N. R_c^N is the logit and $z_c^N = R_c^N$

- We begin at the neuron associated with the class we want the LRP heatmap to explain (C).
  ↳ relevances of other neurons $R_k^N = 0$ and $R_c^N = Z_c^N$.

- Backprop ends at the input where LRP explanation heatmap (relevance of each i/p pixel to neural net's output) is formed. $\overline{R}^0 = [R_k^0]$.

- In the explanation heatmap $\overline{R}^0 = [R_k^0]$.
  i) -ve values indicate areas that REDUCE(D) the classifier's confidence for the explained class C
  ii) +ve values are areas that increased this confidence
  iii) low abolute relevance indicates areas the classifier paid little attention to.

- eqn A1: ε is a small (10^{-4}, 10^{-2}) hyperparam to denoise the heatmap.
  ↳ it is added to Z_k^1 $(\sum_j w_{jk}^1 a_j)$.
  ∵ LRP prop. rule ÷ by Z_k^1, we add ε to prevent DR from being too close to 0 and the relevance exploding
  ↳ also ∵ this method is based on Taylor series approximations, ε represents the residuum in the left out terms of the series?

$$R_j^{L-1} = \sum_k \frac{w_{jk}^L a_j^L}{z_k^L + \text{sign}(z_k^L)\epsilon} R_k^L$$

where: $z_k^L = \sum_j w_{jk}^L a_j^L$

$$R_k^N = \begin{cases} z_k^N \text{ if } k = c \\ 0 \text{ otherwise} \end{cases}$$

- the amount of relevance back-propogated from neuron k to neuron j depends on the relative influence of neuron j on the output of neuron k (z_k^L)
  - ↳ $\frac{w_{jk}^L a_j^L}{z_k^L + \epsilon \frac{|z_k^L|}{z_k^L}}$
- For the bias to influence the classifier's output its influence must flow from its position in the i/p img through all layers.
  - ↳ ∴ ∃ a corresponding backward flow from the o/p to the bias location in the input image.
  - → going forwards now?
- The relevance of a neuron k in layer L (R_k^L) is a function of the outputs of neurons in layer L-1 ($\vec{a}^L = [a_j^L]$).

→ reference point x̃ for taylor's

Let $\vec{a}^L = \tilde{\vec{a}}^L + \vec{h}$

Then:

$$R_k^L(\vec{a}^L) = R_k^L(\tilde{\vec{a}}^L) + \sum_j (a_j^L - \tilde{a}_j^L) \frac{\partial R_k^L}{\partial a_j^L}\Big|_{\tilde{a}_L} + \rho$$

$R_k^L(\vec{a}^L)$ is broken up into one term for each neuron j.

each of these terms represents how much a change in the value of neuron j in layer L-1 would affect the relevance R_k^L of neuron k in layer L.

↳ ∴ it is these terms LRP backpropogates from neuron k to each neuron j.

Final Understanding:

- So basically, any bias to influence o/ps needs to go through every layer.

- Paper says we can see the relevance R_k^L as a function of o/ps of neurons in $L-1$
  ∴ $R_k^L = R_k^L(\bar{a}^L)$..
  ↳ can do taylor approximations etc... to see that $R_k^L(\bar{a}^L)$ is split up into each neuron in $\bar{a}^L$'s indicating influence of each neuron on R_k^L.
  ↳ this is what LRP back propogates.

In summary, LRP performs approximate Taylor expansions at each neuron, using them to determine how much the neuron's relevance was influenced by the neurons in previous layers. Relevance is back-propagated proportionally to this influence. TEA trains the student classifier to match the heatmaps of a teacher classifier. Accordingly, features in the images will influence the student and the teacher in a similar manner. I.e., the reasons behind the TEA student decisions will match the reasons behind the teacher decisions. If the teacher pays no attention to bias, no LRP relevance flows to bias in its heatmaps. Similarly, no LRP will flow to bias in the student's heatmap, making it also pay no attention to bias.

* Why training to match LRP o/ps guarantees correct outputs?

j k

Under full conservativeness condition, LRP redistributes the relevance R_k^L at neuron k (layer L) to the neurons j in layer L-1 with no loss or gain of relevance. I.e., when being back-propagated, relevance is never destroyed or created, only redistributed across neurons. Since relevance starts as the value of a classifier logit (Eq. A3), the sum of all neuron's relevances in any layer matches the classifier output (logit) where the back-propagation started:

$$\sum_k R_k^L = z_c^N \forall L \in [0, ..., N] \quad \text{(A11)}$$

Therefore, under perfect conservativeness, LRP provides a perfect guarantee of correct outputs: if the LRP heatmaps (LRP heatmaps are $\mathbf{R}^0 = [R_k^0]$ in Eq. A11) of two classifiers perfectly match, for all classes c, they will have the same logits z_c^N. However, setting ε to zero creates noisy heatmaps for deep neural networks and high-

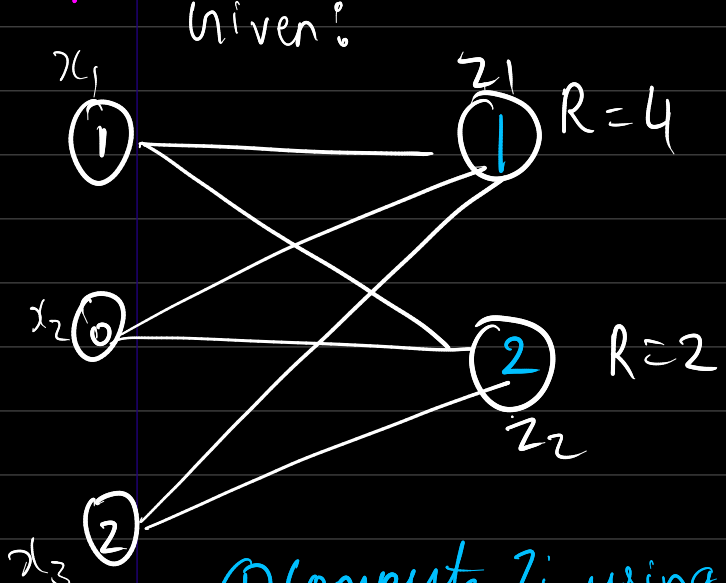
* Why TEA works better with LRP than GRAD-CAM

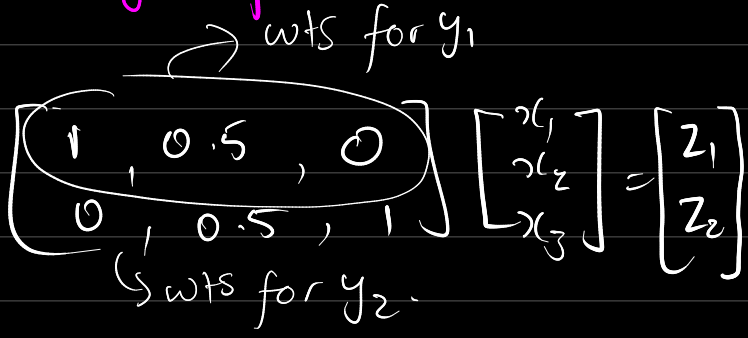
- GRAD-CAM has an issue called spurious-mapping: mapping i/p img features to deep activations not aligned with these features.

↳ the deep layers GRAD-CAM is applied on have huge receptive fields, can even take up the whole image.

* An Example Run through of LRP

Given:





① Compute z_i using a forward linear pass.
② if one of the z_i's is ≤ 0, set its relevance to 0.
∵ its relevance to o/p would have been killed by ReLU in forward pass.
③ add $z = z + \text{sign}(z)\epsilon$
④ $s = \frac{R\_out}{z} = \begin{bmatrix} 4/z_1 \\ 2/z_2 \end{bmatrix} = \begin{bmatrix} 3.96 \\ 0.995 \end{bmatrix}$
⑤ distribute back $\sum_k (w^l_{jk} q^l_j) \times s$ final step.

↳ do this vectorially using w^T?