Layer-Wise Relevance Propogation: An Overview

Authors: Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek, and Klaus-Robert Müller

Presenters: Sarinda Samarasinghe and Nick Meeker

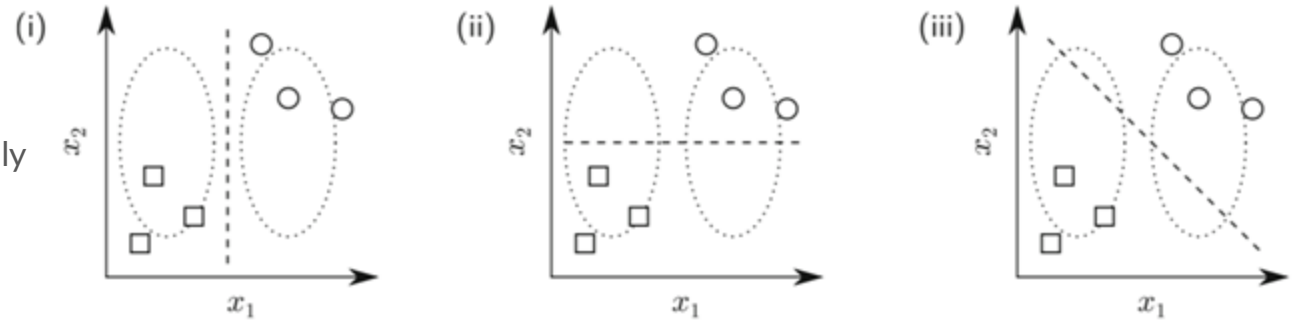
Contents

Introduction

Background & Motivation

- Rise of **large datasets** is a main driver for the success of machine learning techniques in both industrial and scientific applications
- Large datasets can be plagued by **spurious correlations**; leads to **"Clever Hans"** predictors

All classify correctly, but only **(i)** generalizes



Explainable Machine Learning

- **Feature selection** is one solution: only present the model with "good" input features
  - This can be difficult to apply in practice
  - Consider image recognition, where individual pixels do not have fixed roles
- **Explainable machine learning** takes opposite approach: train the model, then examine which features the model actually learned
  - We do not care about feature selection during training
  - "Bad" features can be removed later, and the model can be retrained on cleaned data
- **Taylor Decomposition** is a foundational explainable ML technique related to LRP

Taylor Decomposition

- Produce explanations by performing a **Taylor expansion** of the prediction $f(x)$ at some nearby reference point
- First-order terms (the elements of the sum) quantify the relevance of each input feature, **forming the explanation**

$$f(\boldsymbol{x}) = f(\widetilde{\boldsymbol{x}}) + \sum_{i=1}^{d} (x_i - \widetilde{x}_i) \cdot [\nabla f(\widetilde{\boldsymbol{x}})]_i + \dots$$

Reference point

Problems with Taylor Decomposition

- **Unstable** when applied to DNN
  - **Shattered gradients:** while $f(x)$ is generally accurate, the gradient is often very noisy, containing little meaningful information
  - **Adversarial examples:** small perturbations of the input can cause dramatic changes to the function value
- It can be difficult to choose a **meaningful reference point**
  - **0** → may be far away from real input

Alternative Explanation Techniques

- Integrate a large number of local gradient estimates
- Replace the gradient with a coarser estimate of effect, such as model response to patch-like perturbations
- Optimization techniques involving a local surrogate model or the explanation itself
- **Common Problem:** these techniques are all **computationally expensive**, involving multiple network evaluation

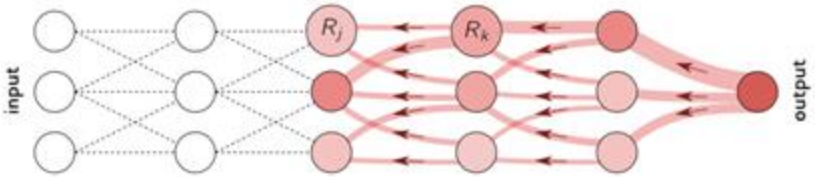
Four Properties of Good Explanation Techniques

- **Conservation:** if we find explainable evidence in the output, it **must** show up somewhere in the input features (no loss of evidence)
- **Positivity:** either a feature is relevant (positive) or irrelevant (zero)
- **Continuity:** if two inputs are almost the same, and the prediction is almost the same, then the explanation should be almost the same
- **Selectivity:** models must **agree** with explanation; removing evidence from input should reduce confidence in the output

LRP satisfies all of these properties; the previous techniques do not

Layer-Wise Relevance Propogation (LRP)

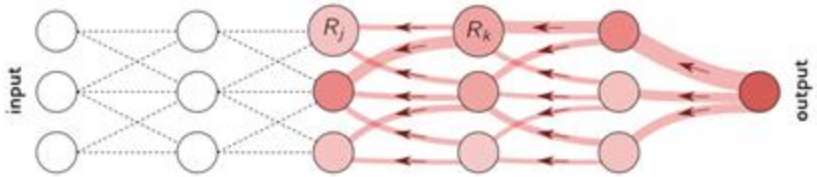
LRP Explained



- LRP is an explanation technique which propagates the prediction backwards using purposely designed local propagation rules
- LRP is subject to the **conservation** property
  - What has been received by a neuron must be redistributed to the lower layer in **equal amount**
  - (It's also subject to the other properties, but this one is explicitly mentioned)

$$R_j = \sum_k \frac{z_{jk}}{\sum_j z_{jk}} R_k$$

LRP Explained (ii)



Referring to the previous equation:

- z_{jk}: the quantity which models the extent to which neuron *j* has contributed to make neuron *k* relevant
- $\sum_j z_{jk}$: the denominator which serves to enforce the conservation property

$$R_j = \sum_k \frac{z_{jk}}{\sum_j z_{jk}} R_k$$

LRP Rules for Deep Rectifier Networks

- **Question:** How do we determine z_{jk} (the contribution of neuron j to R_k)?
- **Answer:** With LRP rules!
- We'll talk about three in the following slides:
  - Basic (**LRP-0**)
  - Epsilon (**LRP-ϵ**)
  - Gamma (**LRP-γ**)
- Note that we're working in the context of ReLU activations:

$$a_k = \max\left(0, \sum_{0,j} a_j w_{jk}\right)$$

LRP Rules: LRP-0

- Redistribute relevance in proportion to the contributions of each input to the neuron activation
- Note $z_{jk} = a_j w_{jk}$
- Properties:
  - If $a_j = 0$ or $w_{jk} = 0$, then $R_j = 0$, which allows for compatibility with concepts such as zero weight, deactivation, or absent connections
  - Uniform application produces an explanation equivalent to $(Gradient \times Input)$, which is undesirable since gradients are noisy

$$R_j = \sum_k \frac{z_{jk}}{\sum_j z_{jk}} R_k \longrightarrow R_j = \sum_k \frac{a_j w_{jk}}{\sum_{0,j} a_j w_{jk}} R_k$$

LRP Rules: LRP-ϵ

- This rule aims to solve the problem of gradient noise by introducing a small positive term, ϵ, to the denominator
- ϵ diminishes relevance scores, aiming to absorb some relevance when contributions to neuron k are weak, contradictory, etc.
- As ϵ becomes larger, only the most salient explanation factors are preserved
- **Result:** sparser explanations in terms of input features, and less noise

$$R_j = \sum_k \frac{a_j w_{jk}}{\sum_{0,j} a_j w_{jk}} R_k \longrightarrow R_j = \sum_k \frac{a_j w_{jk}}{\epsilon + \sum_{0,j} a_j w_{jk}} R_k$$

LRP Rules: LRP-γ

- This rule aims to reduce noise and improve stability by favoring the effect of positive contributions over negative ones with the introduction of a γ parameter applied to w_{jk}.
- As γ increases, negative contributions disappear
- Limits how large positive and negative contributions can grow during propagation, improving stability

$$R_j = \sum_k \frac{a_j w_{jk}}{\sum_{0,j} a_j w_{jk}} R_k \longrightarrow R_j = \sum_k \frac{a_j \cdot (w_{jk} + \gamma w_{jk}^+)}{\sum_{0,j} a_j \cdot (w_{jk} + \gamma w_{jk}^+)} R_k$$

Bonus LRP Rule: LRP-$\alpha\beta$

- Like **LRP-γ**, this rule aims to treat positive and negative contributions asymmetrically
- Applies two parameters, α and β, to positive and negative contributions, respectively
- Subject to conservation constraint $\alpha = \beta + 1$
- Using **LRP-γ** where $\gamma = \infty$ causes **LRP-γ** to become equivalent to LRP-$\alpha\beta$ where $\alpha = 1$ and $\beta = 0$ (among other rules not covered in this paper)

$$R_j = \sum_k \left(\alpha \frac{(a_j w_{jk})^+}{\sum_{0,j} (a_j w_{jk})^+} - \beta \frac{(a_j w_{jk})^-}{\sum_{0,j} (a_j w_{jk})^-} \right) R_k$$

Implementing LRP Efficiently

Apply LRP rule to weights

$$R_j = \sum_k \frac{a_j \cdot \rho(w_{jk})}{\epsilon + \sum_{0,j} a_j \cdot \rho(w_{jk})} R_k$$

- Consider the generic LRP rule (pictured right)
- For any layer j, R_j can be computed in four steps:

$$\begin{aligned}
\forall_k: \quad & z_k = \epsilon + \textstyle\sum_{0,j} a_j \cdot \rho(w_{jk}) && \text{(forward pass)} \\
\forall_k: \quad & s_k = R_k / z_k && \text{(element-wise division)} \\
\forall_j: \quad & c_j = \textstyle\sum_k \rho(w_{jk}) \cdot s_k && \text{(backward pass)} \\
\forall_j: \quad & R_j = a_j \, c_j && \text{(element-wise product)}
\end{aligned}$$

- Note the third step is equivalent to a gradient computation, where $\boldsymbol{a}$ is the vector of lower-layer activations:

$$c_j = \left[\nabla\left(\textstyle\sum_k z_k(\boldsymbol{a}) \cdot s_k\right)\right]_j$$

Implementing LRP Efficiently (in code)

$$\forall_k: \; z_k = \epsilon + \sum_{0,j} a_j \cdot \rho(w_{jk})$$

$$\forall_k: \; s_k = R_k / z_k$$

$$\forall_j: \; c_j = \sum_k \rho(w_{jk}) \cdot s_k$$

$$\forall_j: \; R_j = a_j \, c_j$$

```
def relprop(a,layer,R):
    z = epsilon + rho(layer).forward(a)
    s = R/(z+1e-9)
    (z*s.data).sum().backward()
    c = a.grad
    R = a*c
    return R
```

LRP as a Deep Taylor Decomposition

- **Deep Taylor Decomposition** views LRP as a succession of Taylor expansions performed at each neuron
- Treat the relevance score R_k as a function of lower-level activations $(a_j)_j$ denoted by the vector $\boldsymbol{a}$, and then perform a first-order Taylor expansion of $R_k(\boldsymbol{a})$ at some reference point in the space of activations:

$$R_k(\boldsymbol{a}) = R_k(\widetilde{\boldsymbol{a}}) + \sum\nolimits_{0,j}(a_j - \widetilde{a}_j) \cdot [\nabla R_k(\widetilde{\boldsymbol{a}})]_j + \dots$$

LRP as a Deep Taylor Decomposition (ii)

- DTD requires a **closed-form** expression for the terms of the previous equation
- Substitute the true relevance function with a model that is easier to analyze:

$$\widehat{R}_k(\boldsymbol{a}) = \max\left(0, \sum_{0,j} a_j w_{jk}\right) \cdot c_k$$

- Modulation term c_k is a constant set in such a way that $\hat{R}_k(\boldsymbol{a}) = R_k(\boldsymbol{a})$ at the current data point
- Then the Taylor expansion becomes:

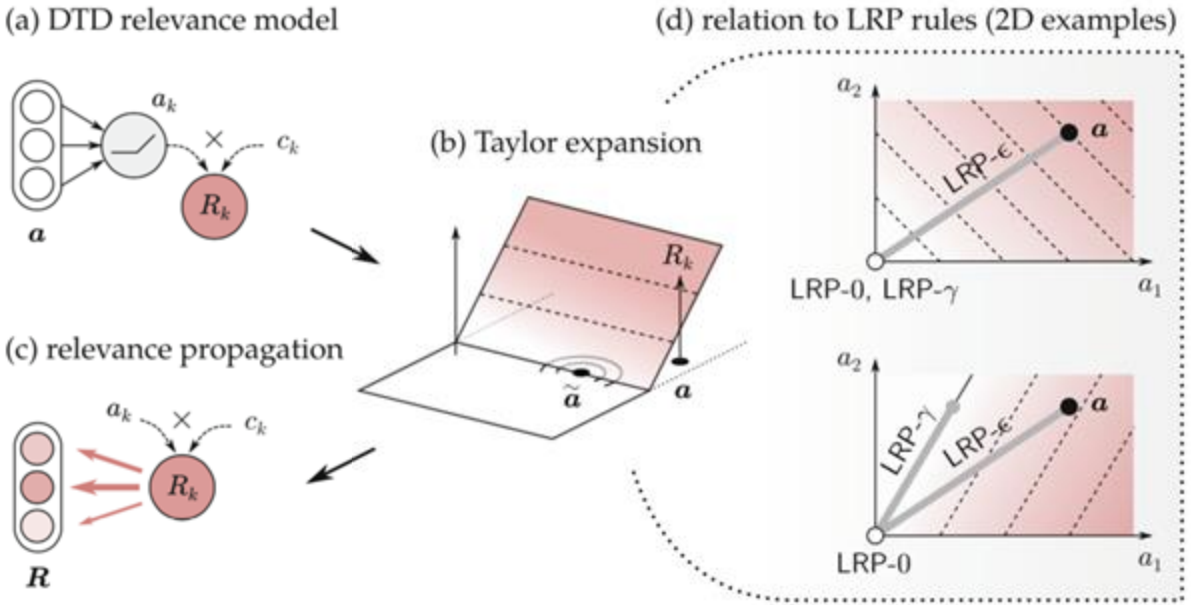
$$\widehat{R}_k(\boldsymbol{a}) = \widehat{R}_k(\widetilde{\boldsymbol{a}}) + \sum_{0,j}(a_j - \widetilde{a}_j) \cdot w_{jk}\, c_k$$

LRP as a Deep Taylor Decomposition (iii)

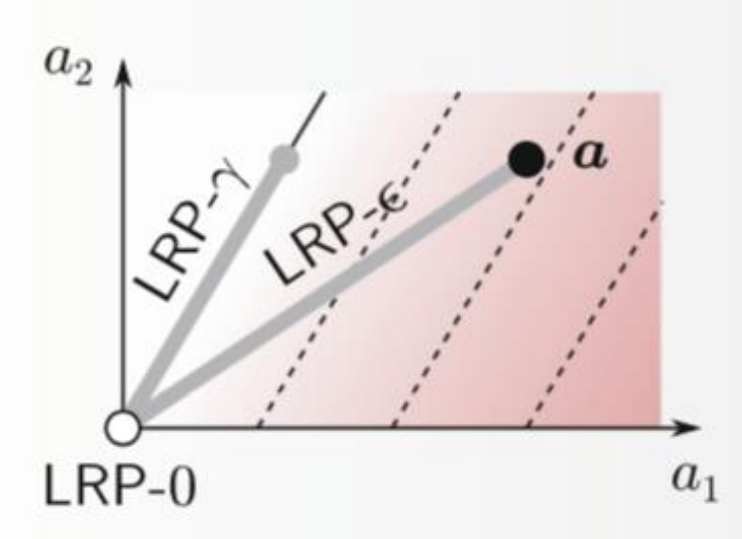
- **Relation to LRP-0/ϵ/γ:** LRP rules can be recovered within the DTD framework by changing the reference point:
  - **LRP-0:** 0
  - **LRP-ϵ:** $\epsilon \cdot (a_k + \epsilon) - 1 \; \boldsymbol{a}$
  - **LRP-γ**:

$$\{\boldsymbol{a} - t \cdot \boldsymbol{a} \odot (\mathbf{1} + \gamma \cdot \mathbf{1}_{\boldsymbol{w}_k \succeq \mathbf{0}}) \mid t \in \mathbb{R}\}$$

LRP as a Deep Taylor Decomposition (iv)



LRP as Deep Taylor Decomposition (v)



- LRP-0: $\tilde{a}$=0

$$R_j = \sum_k \frac{a_j w_{jk}}{\sum_{0,j} a_j w_{jk}} R_k$$

- LRP-ϵ: $\tilde{a}$=$\epsilon \cdot (a_k + \epsilon) - 1 \; a$

$$R_j = \sum_k \frac{a_j w_{jk}}{\epsilon + \sum_{0,j} a_j w_{jk}} R_k$$

- LRP-γ: $\tilde{a}=\left\{a - t \cdot a \odot \left(1 + \gamma \cdot 1_{w_k \succeq 0}\right) \mid t \in \mathbb{R}\right\}$

$$R_j = \sum_k \frac{a_j \cdot (w_{jk} + \gamma w_{jk}^+)}{\sum_{0,j} a_j \cdot (w_{jk} + \gamma w_{jk}^+)} R_k$$

Which LRP Rule for Which Layer?

Properties of Explanations

- LRP is a general framework for propagation, leaving flexibility for different rules at each layer, and for the parameters ε and γ
  - Optimal selection for parameters requires a measure of explanation quality, which is still being researched

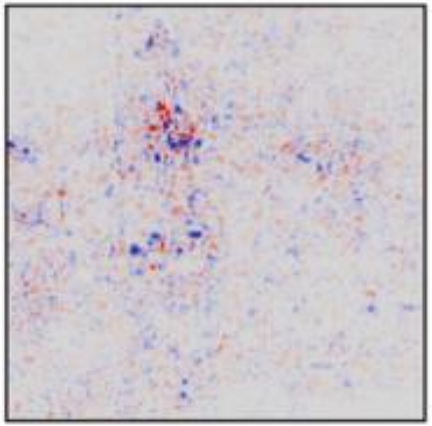
- Focus on 2 main explanation properties: **fidelity** and **understandability**
  - **Fidelity** is the accuracy of the explanation's representation of the output neuron
    - To visually assess fidelity, we must assume the network properly solved the task (is using correct visual features and avoiding distracting elements)
  - **Understandability** is the interpretability of the explanation to a human

Properties of Explanations (ii)

Input

Model: VGG-16

LRP-0



red = positively relevance
blue = negatively relevance

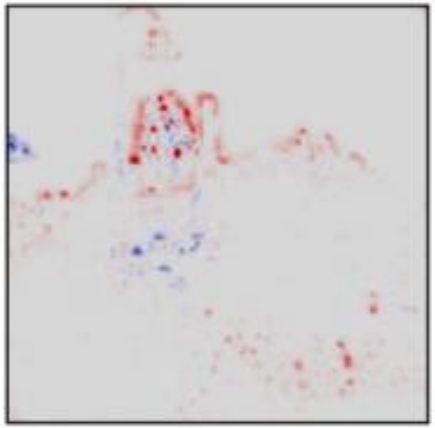
- Explanation is complex
  - Lacks understandability
- Fails to focus on castle
  - Lacks fidelity

Properties of Explanations (ii)

Input

Model: VGG-16

LRP-ϵ



red = positive relevance
blue = negative relevance

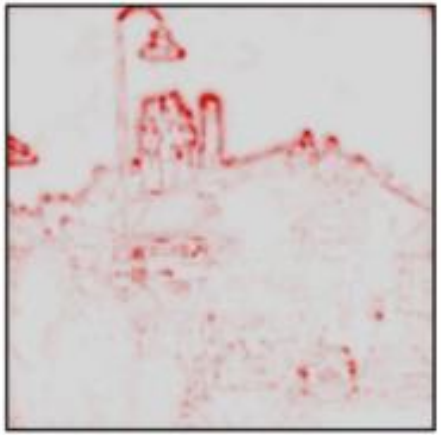
- Explanation is very sparse
  - Lacks understandability
- Much of the noise is removed
  - Has fidelity

Properties of Explanations (ii)

Input

Model: VGG-16

LRP-γ



red = positive relevance
blue = negative relevance

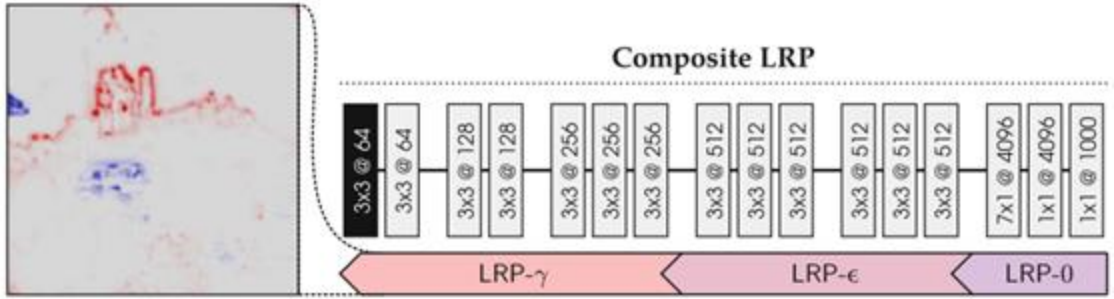
- Explanation is very clearly outlined
  - Has understandability
- Too much is highlighted (Ex. lamp post)
  - Lacks fidelity

Properties of Explanations (ii)

Input

Model: VGG-16

- Well outlined explanation
  - Has understandability
- Castle is correctly identified
  - Has fidelity



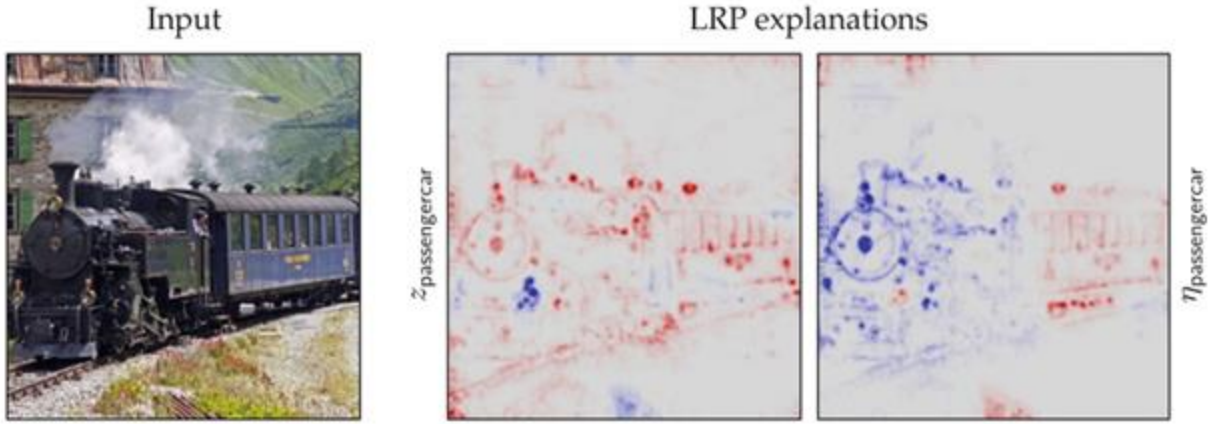
red = positive relevance
blue = negative relevance

Rule Choices with VGG-16

LAYER	RULE	EXPLANATION

Handling the Top Layer

$$z_c = \sum_{0,k} a_k w_{kc}$$
$$\mathrm{P}(\omega_c) = \exp(z_c) / \sum_{c'} \exp(z_{c'})$$
$$\eta_c = \log[\mathrm{P}(\omega_c)/(1-\mathrm{P}(\omega_c))]$$



Classification Task: Passenger Car

red = positive relevance
blue = negative relevance

Conclusion

Conclusion

- Layer-wise Relevance Propagation (LRP) can explain SOTA predictions in terms of their input features by propagating the prediction backwards through the model with various rules
- These can be implemented efficiently and modularly (in most modern neural net softwares)
- Through parameter tuning even complex models can have high quality explanations
- With Neuralization-Propagation (NEON), LRP can be applied beyond DNNs to other model types, increasing its scope to help many other scenarios that require explainable machine learning solutions

For

- LRP satisfies several properties of good explanatory ML techniques, and produces faithful and understandable explanations
- LRP can be extended to a broad range of ML models beyond just DNN
- LRP can be implemented efficiently compared to other explanation techniques
- LRP can be easily modified to fit a variety of use cases via different rules

Against

- Little empirical evidence presented, with no comparison to other SOTA methods
- Many types of LRP rules were left out and it is unclear why this is the case
- LRP itself only applies to ReLU activations
- Explanation quality still requires human assessment, which can be time-consuming and error-prone
- No formal evaluation criteria, only *fidelity* and *understandability*, which are subject to bias
    - Fidelity requires the assumption that the model is functioning exactly as intended as well
- Authors offer heuristics for applying LRP rules (i.e., “use epsilon for middle layers”), but only support these with intuition rather than with more rigorous forms of evidence
- The relationship to the DTD framework is clear, but it is not clear why this relationship is valuable

Questions?