

# Obtaining Faithful Interpretations from Compositional Neural Networks

Sanjay Subramanian<sup>\*1</sup> Ben Bogin<sup>\*2</sup> Nitish Gupta<sup>\*3</sup>  
Tomer Wolfson<sup>1,2</sup> Sameer Singh<sup>4</sup> Jonathan Berant<sup>1,2</sup> Matt Gardner<sup>1</sup>  
<sup>1</sup>Allen Institute for AI <sup>2</sup>Tel-Aviv University  
<sup>3</sup>University of Pennsylvania <sup>4</sup>University of California, Irvine

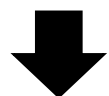
紹介者：豊田工業大学 知能数理研究室 博士3年 辻村 有輝

問題文：“*the llamas in both images are eating*”



“プログラム”に変換

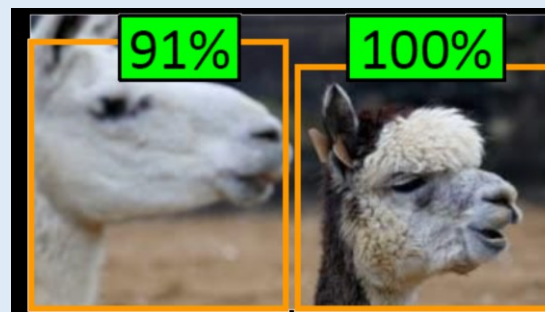
... ( **filter[eating]**( **find(llamas)** ) ) ...



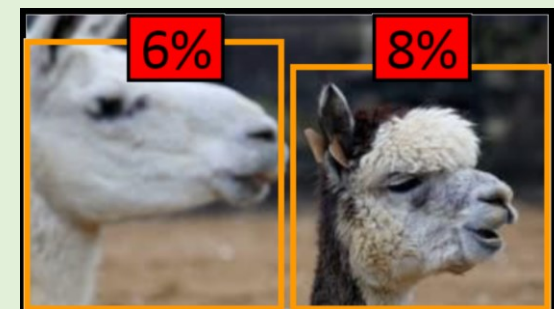
モジュール **find(llamas)** の実行



**理想的**なモジュール出力  
(このような出力が欲しい)



**意図しない**モジュール出力；  
**filter[eating]**も同時に  
解いてしまっている

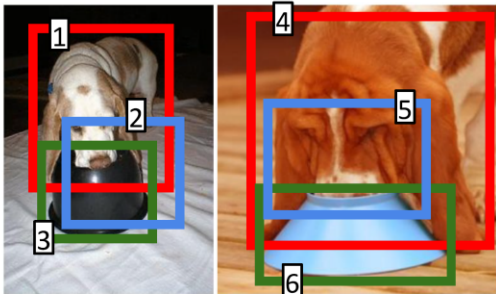


# 概要・貢献

- NMNs (Neural Module Networks) についての論文
  - 自然言語による文を**複数のモジュールの組み合わせ**で出来たプログラムに変換し各モジュールを順に実行することで出力を得るモデル
- NMNsの「**モジュール単位の正当性 (module-wise faithfulness)**」のコンセプトを提案
  - **定量的**な評価手法も整備
- モジュール単位の正当性を向上させる手法を提案
- モジュール単位の正当性を検証するために構築した追加アノテーションを公開
  - <https://github.com/allenai/faithful-nmn>

# NMNs (Neural Module Networks)とは

- 自然言語による文を**複数のモジュールの組み合わせ**で出来たプログラムに変換し各モジュールを順に実行していくことで出力を得るモデル

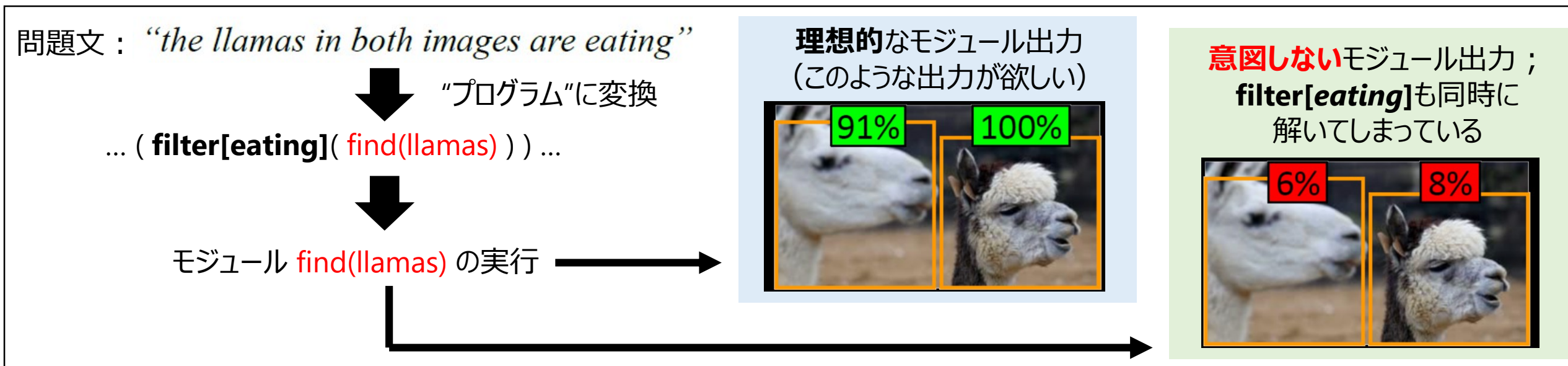
<i>two dogs are touching a food dish with their face</i>		
		
<b>Program</b>		<b>Output</b>
equal		<span style="border: 1px dashed black;">True</span>
count		2
with-relation [is touching]		[2, 5]
relocate [face]		[2, 5]
find [dog]		[1, 4]
find [food dish]		[3, 6]
number [two]		2

- プログラムへの変換を担う “**parser**” と実際に実行する “**executor**” の二つから成る
  - 本論文では主に **executor** に焦点を当てる
- 各モジュールは独立のネットワークで, End-task (+ 補助タスク) の損失から学習
- 各モジュールの入出力・ネットワーク構造はあらかじめ人手で設計

Module	Output	Implementation
$\text{find}[q_{att}]$	$p$	$W_1^T([x; v]) + b_1$
$\text{filter}[q_{att}](p)$	$p$	$p \odot (W_1^T([x; v]) + b_1)$

# モチベーション

- NMNsにより構築されたプログラム構造やそれぞれのモジュールの出力から、システムの推論プロセスの分析が可能とされる
- しかしながら、実際はモジュールが**意図した通りの役割にならない**ケースがある
  - End-taskの損失だけで学習ではモジュールの挙動を意図通りに導ききる保証なし
  - モジュールの役割が意図と違うと分析の前提が崩れてしまう



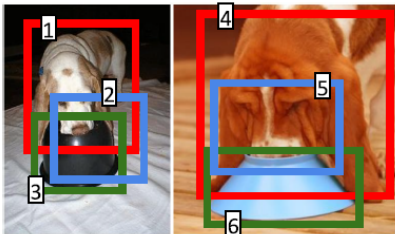
➡ **モジュール単体の正当性 (module-wise faithfulness) を定量的に検証したい**

# どうやって検証するか

## モジュールごとの正解とMetricを用意してスコアで比較する

- Metricはタスク・モジュール両方の設計に依存した定義になっている
- Metricは全モジュールに定義したわけではなく、各セットのうちの一部のみ

- データセット
  - NLVR2 : 画像を含む

<p><i>two dogs are touching a food dish with their face</i></p> 	<pre> Program equal count   with-relation [is touching]   relocate [face]   find [dog]   find [food dish] number [two] </pre>	<pre> Output True 2 [2, 5] [2, 5] [1, 4] [3, 6] 2 </pre>
--	---	--

- DROP : テキストのみ

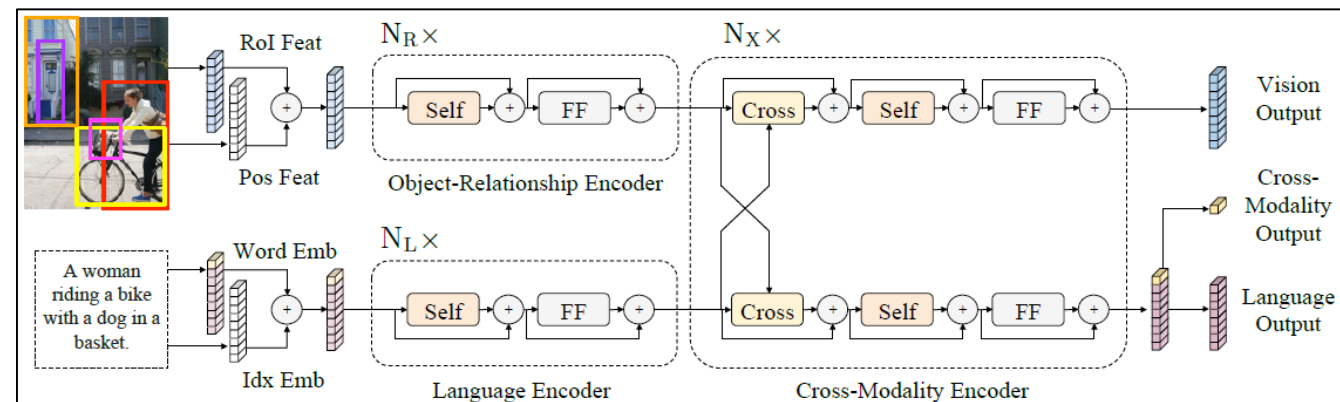
<p><i>Who threw the longest touchdown pass in the second half?</i></p> <p>In the first quarter, the Texans trailed early after QB Kerry Collins threw a 19-yard TD pass [1] to WR Nate Washington. Second quarter started with kicker Neil Rackers made a 37-yard field goal, and the quarter closed with kicker Rob Bironas hitting a 30-yard field goal. The Texans tried to cut the lead with QB Matt Schaub getting a 8-yard TD pass [2] to WR Andre Johnson, but the Titans would pull away with RB Javon Ringer throwing a 7-yard TD pass [3]. The Texans tried to come back into the game in the fourth quarter, but only came away with Schaub [4] throwing a 12-yard TD pass [5] to WR Kevin Walter.</p>	<pre> Program relocate[who threw] find-max-num filter [the second half] find [touchdown pass] </pre>	<pre> Output {Schaub [4]} [5] [2, 3, 5] [1, 2, 3, 5] </pre>
---	--	---

# Visual-NMN for NLVR2

- NLVR2
  - 画像ペアとそれらに関する説明文が与えられ，説明文の真偽を答える

<i>two dogs are touching a food dish with their face</i>		
<b>Program</b>		<b>Output</b>
equal		True
count		2
with-relation [is touching]		[2, 5]
relocate [face]		[2, 5]
find [dog]		[1, 4]
find [food dish]		[3, 6]
number [two]		2

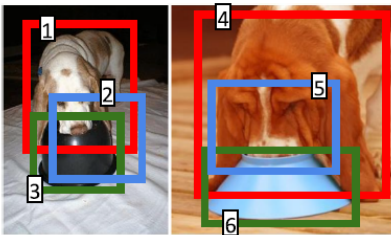
- Visual-NMN
  - LXMERTベースのモデル
  - マルチモーダルなTransformer
  - Visual-NMNではparserを利用せず常に正しいprogramに沿ってexecutorを実行





# Visual-NMN for NLVR2におけるMetric

- Metric : **モジュールの出力が正しいBounding boxを指しているかどうか**
  - 正しいboxとのIOUが0.5以上となるような候補boxを選べた時正解とする
  - Precision/Recall/F値で評価
- Metricが定義されるモジュールは find, filter, with-relation, relocateの4つ
  - 全て出力がbounding boxの形で出るモジュールたち
  - 各候補boxについて採用するかどうかを確率値で出力
  - 0.5を超えればそのboxを採用としIOUを計算する

<i>two dogs are touching a food dish with their face</i>		<b>Program</b>	<b>Output</b>
	equal	<div>True</div>	
	count	2	
	with-relation [is touching]	[2, 5]	
	relocate [face]	[2, 5]	
	find [dog]	[1, 4]	
	find [food dish]	[3, 6]	
	number [two]	2	

# Text-NMN for DROP

- DROP
  - Text only
  - 最終出力もテキスト

<i>Who threw the longest touchdown pass in the second half?</i> In the first quarter, the Texans trailed early after QB Kerry Collins threw a 19-yard TD pass [1] to WR Nate Washington. Second quarter started with kicker Neil Rackers made a 37-yard field goal, and the quarter closed with kicker Rob Bironas hitting a 30-yard field goal. The Texans tried to cut the lead with QB Matt Schaub getting a 8-yard TD pass [2] to WR Andre Johnson, but the Titans would pull away with RB Javon Ringer throwing a 7-yard TD pass [3]. The Texans tried to come back into the game in the fourth quarter, but only came away with Schaub [4] throwing a 12-yard TD pass [5] to WR Kevin Walter.		
<b>Program</b>		<b>Output</b>
relocate[who threw]		{Schaub [4]}
find-max-num		[5]
filter [the second half]		[2, 3, 5]
find [touchdown pass]		[1, 2, 3, 5]

- Text-NMN
  - 著者グループがICLR2020で発表したモデル
  - 出力はなんらかの確率分布の形式
    - 例えば find モジュールの出力は  
パラグラフの単語列に対する確率分布  
(アテンションの形になっている)
  - 元のモデルに加え addition, subtraction, extract-answer (!!!) の3モジュールを追加

Module	In	Out	Task
find	Q	P	For question spans in the input, find similar spans in the passage
filter	Q, P	P	Based on the question, select a subset of spans from the input
relocate	Q, P	P	Find the argument asked for in the question for input paragraph spans
find-num	P	N	} Find the number(s) / date(s) associated to the input paragraph spans
find-date	P	D	
count	P	C	Count the number of input passage spans
...			

- **Question (Q) and Paragraph (P) attentions:** soft subsets of relevant tokens in the text.
- **Number (N) and Date (D):** soft subset of unique numbers and dates from the passage. <sup>1</sup>
- **Count Number (C):** count value as a distribution over the supported count values (0 – 9).
- **Time Delta (TD):** a value amongst all possible unique differences between dates in the paragraph. In this work, we consider differences in terms of years.
- **Span (S):** span-type answers as two probability values (start/end) for each paragraph token.



# Text-NMN for DROPにおけるMetric

- Metric : **Negative Log-Likelihood**のようなスコア

$$I = - \sum_{i=1}^N \left( \log \sum_{j=t_s^i}^{t_e^i} p_{\text{att}}^j \right) \quad \begin{array}{l} p_{\text{att}}^j : \text{トークン } j \text{ の出力アテンション} \\ (t_s^i, t_e^i) : \text{あるモジュールの1つの正解単語列スパン} \end{array}$$

- 正しい単語に強いアテンションを与えると良スコア
- $p_{\text{att}}^j$  はアテンションなのでうまく正解スパンにアテンションを張れればsumして1
- 対象モジュール : find, filter, relocate, find-min/max-num, find-num/date
- DROPデータセットには正解programがアノテートされていない  
➡ 開発データのうち215問に手作業で正解programとその正解スパンをアノテートした

# ではどうすればFaithfulスコアが変動するか (1/3)

## Visual-NMNについて

- countモジュールはよく用いられる基本的なモジュールの一つだが、そのモジュールの**表現能力**がFaithfulnessに影響する可能性がある
  - 表現力の高いモデルは余計な処理を行ってしまう可能性がある
    - Layer-count module : 2層MLPによるcountモジュールの表現
  - 表現力の低いモデルは純粹なcountの体現ではあるものの、例えば与えられた bounding box がほとんど被っているときも愚直に加算するなど副作用もある
    - Sum-count module :  $\sum_i p_i$  で表現される素直な加算
  - 間を取った表現もある
    - Graph-count module : パラメータが300個未満程度のcountモジュール
- 仮説 : 表現力が高いほどFaithfulスコアは落ちやすい ➡ 実験で検証

## ではどうすればFaithfulスコアが変動するか (2/3)

### Visual-NMNについて

- コンテキスト情報の除外によるFaithfulスコア向上の可能性
  - モジュールの入力には質問文中のアテンションによる重み付きトークン表現も含まれる
  - アテンションにより纏められる直前の各トークン表現は, Transformerライクなモデルで文脈情報をエンコード済み
    - ➡ filter[**red**](find[**car**])のようなプログラムとなった場合, carに大きな重みを与えたモデルではあるものの, そのcarの表現の中にredの情報が含まれている
      - ➡ 実質的にfind[red car]であり, findが先走ってfilterの仕事まで行いうる
- ➡ モジュールに与える入力ベクトルとしては"car"だけのトークンから改めて計算した表現を与え, redを隠すことでモジュール単位の処理の分割が促進されるのではないか？
- モジュールにふさわしい入出力をもつデータセットを用意してpre-trainingしておく

# ではどうすればFaithfulスコアが変動するか (3/3)

Text-NMNについて

- モジュール単位の正解を用意して教師あり学習すれば上がりそう
  - 実験ではfind-num, find-dateモジュールについて正解を用意する
  - 著者グループの先行研究で実装済み（本論文での新規部分はFaithfulnessの評価）
- 必要となりそうなAtomicなモジュールが欠けているとFaithfulnessが下がりそう
  - 他のモジュールが本来担当しなくていい仕事を代わりにやってしまうと予想される
  - 実験ではfind-max-num, num-compare等のモジュールを除いて学習してみる
- いきなり答えを出すextract-answerモジュールを用意すればFaithfulnessが落ちそう
  - 他のモジュールの仕事を奪いそう

# 実験結果 (Visual-NMN)

Model	Performance (Accuracy)	Overall Faithful. ( $\uparrow$ )			Module-wise Faithfulness $F_1$ ( $\uparrow$ )			
		Prec.	Rec.	$F_1$	find	filter	with-relation	relocate
LXMERT	<b>71.7</b>							
Upper Bound		1	0.84	0.89	0.89	0.92	0.95	0.75
① NMN w/ Layer-count	71.2	0.39	0.39	0.11	0.12	0.20	0.37	<b>0.27</b>
② NMN w/ Sum-count	68.4	<b>0.49</b>	0.31	0.28	0.31	0.32	0.44	0.26
NMN w/ Graph-count	69.6	0.37	0.39	0.28	0.31	0.29	0.37	0.19
③ NMN w/ Graph-count + decont.	67.3	0.29	0.51	0.33	0.38	0.30	0.36	0.13
④ NMN w/ Graph-count + pretraining	69.6	0.44	0.49	0.36	0.39	0.34	0.42	0.21
NMN w/ Graph-count + decont. + pretraining	68.7	0.42	<b>0.66</b>	<b>0.47</b>	<b>0.52</b>	<b>0.41</b>	<b>0.47</b>	0.21

- ① 自由度が最も大きいLayer-countが最高Acc/最低Faithful
- ② 自由度が最も小さいSum-countは3者中最低AccだがLayer-countより高いFaithful
- ③ 入力から周辺文脈情報を落とすとAccは落ちるがFaithfulnessは向上
- ④ find, filterモジュールの事前学習はFaithfulnessに良い影響

# 実験結果 (Text-NMN)

Model	Performance (F <sub>1</sub> Score)	Overall Faithful. (cross-entropy* ↓)	Module-wise Faithfulness* (↓)					
			find	filter	relocate	min-max <sup>†</sup>	find-arg <sup>†</sup>	
Text-NMN w/o prog-sup								
③	w/ extract-answer	63.5	9.5	13.3	9.5	3.5	2.6	9.9
	w/o extract-answer	60.8	<b>6.9</b>	8.1	7.3	1.3	1.7	8.5
Text-NMN w/ prog-sup								
	no auxiliary sup	65.3	11.2	13.7	16.9	1.5	2.2	13.0
②	w/o sorting & comparison	63.8	8.4	9.6	11.1	1.6	1.3	10.6
①	w/ module-output-sup	<b>65.7</b>	<b>6.5</b>	<b>7.6</b>	<b>10.7</b>	<b>1.3</b>	<b>1.2</b>	<b>7.6</b>

Table 2: Faithfulness and performance scores for various NMNs on DROP. \*lower is better. <sup>†</sup>min-max is average faithfulness of find-min-num and find-max-num; find-arg of find-num and find-date.

- ① モジュール出力に対する教師あり学習(module-output-sup)がスコアにもfaithfulnessにも有効
- ② find-max-numやnum-compareモジュールの学習を行わないとその分faithfulスコアも低下
- ③ 直接答えを出すextract-answerモジュールはスコアを向上させる一方faithfulスコアは落とす



# Faithfulなモジュールは汎化性能が高いか？ (1/2)

Faithfulなモデルとそうでないモデルで汎化性能の違いがあるかどうかを評価

- 勝手に気持ちを推測：faithfulなモジュールを組めるモデルなら複雑な質問に対しても個々のモジュールの仕事をきっちりこなして汎化性能が高くなることを期待している？

Visual-NMNにて以下の3つの設定で評価

- (1) プログラム長が最大7の事例のみで訓練し，テスト時は最低長8の事例で評価
- (2) filterモジュールを最大で1つしか使わない事例で訓練，テストは必ず2つ以上出現
- (3) 訓練時filter, with-relationモジュールは同時出現しない，テスト時は必ず同時出現

結果：最もunfaithfulなモデルが最も高いF値 (!!!)

# Faithfulなモジュールは汎化性能が高いか？ (2/2)

一方でText-NMNでは逆の結果に ただし自分としては主張はあまり腑に落ちていない・・・  
設定

- アノテートした開発データを新たに訓練・テストデータに分割する
  - gold programを使うため
- maximum, greater-thanモジュールが必要なデータは必ず訓練データに入れる
- minimum, less-thanモジュールが必要なデータは必ずテストデータに入れる
- モジュールに対する教師あり学習を行う／行わない設定でそれぞれ訓練・評価
  - 教師あり学習**あり**：タスクF値 = 78.3%
  - 教師あり学習**なし**：タスクF値 = 32.3%

主張：モジュールに対する教師あり学習はFaithfulnessをあげるので、  
Faithfulnessの向上が汎化性能に貢献した

## まとめ

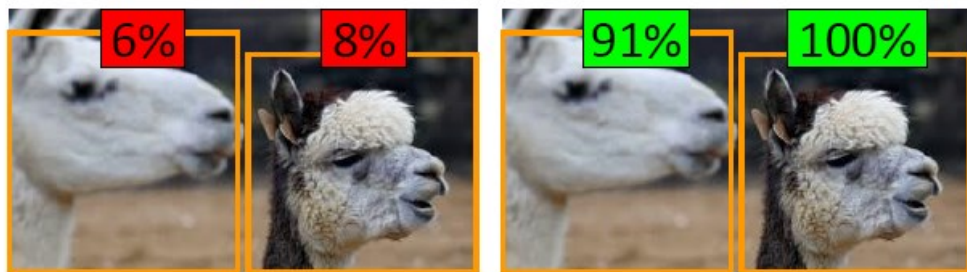
- NMNsの「**モジュール単位の正当性 (module-wise faithfulness)**」のコンセプトを提案
  - **定量的**な評価手法も整備：モジュールにおける正解率からスコアを出す
- モジュール単位の正当性を向上させる手法を提案・検証
  - 残念ながらFaithfulなら高スコアという結果にはならず，むしろタスク性能は落ちがち
- モジュール単位の正当性を検証するために構築した追加アノテーションを公開
  - <https://github.com/allenai/faithful-nmn>

# 所感

- Text-NMNの実験の記述が難しかった
  - w/o sorting & comparisonやgeneralizationの部分がまだよく分かっていない...
- metricがfaithfulnessを反映するという根拠は明確には示されていない
  - 感覚的には正しそうだが、実際の人間の感覚とスコアとの相関などを出してほしかった
- 人間が頑張っている感
  - モデルが自分で自分の役割が何なのか気付いてほしい
  - モデルやデータセットに依存している感じも強い

# 補助資料

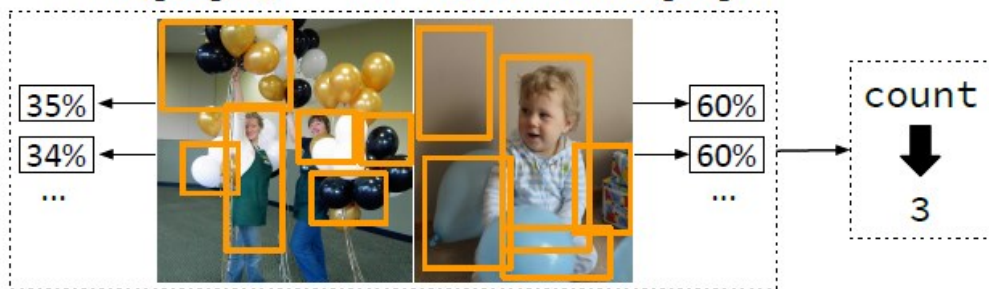
find[llamas] utt: "the llamas in both images are eating"



(a)

(b)

find[people] utt: "there are three people"



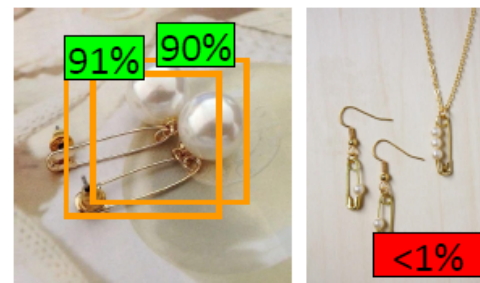
(c)

find[touchdown run]

The Redskins obtained an early lead when RB Clinton Portis scored on a 3-yard TD run. St. Louis scored again when free safety Oshiomogho Atogwe scored a 75 yards touchdown. Washington regained the lead with .... and a Clinton Portis 2-yard rushing TD. St. Louis would come back with a 49-yard field goal.

(d)

find[safety pin] utt: "at least one safety pin is not embellished."



(e)

Figure 3: Comparison of module outputs between NMN versions: (a) Visual-NMN with contextualized representations, (b) Visual-NMN with decontextualized representations, (c) model using a parameter-rich count layer (Layer-Count), (d) Text-NMN trained with-

out sorting module produces an incorrect find output (misses 2-yard rushing TD), and (e) Visual-NMN failure case with a rare object (of w/ Graph-count + decont. + pretraining)