A Tale of Two Oracles: Defining and Verifying when AI Systems are Safe

Edoardo Manino

University of Manchester (UK)
This work is funded by the EPSRC grant EP/T026995/1 entitled "EnnCore: End-to-End Conceptual Guarding of Neural Architectures" under *Security for all in an AI enabled society*

EnnCore Project: an overview

Overview

- Vision: End-to-End Conceptual Guarding of Neural Architectures
- 4 Packages: Explainability, Symbolic Verification, Software Safety, Case Studies
- Large project: 6 academics, 5 postdocs, 2 academic institutions, 2 industrial partners, £1.7M funding from EPSRC

EnnCore

Outcomes up to January 2023

- 30+ publications, 1 workshop at AAAI'22, 1 software tool, 3 awards

EnnCore Project: a personal view

My profile

- Studied computer engineering
- PhD in theoretical ML
- Postdoc in NN verification

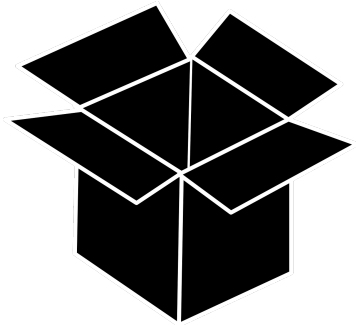
EnnCore

This talk is based on:

- Manino et al., *Systematicity, Compositionality and Transitivity of Deep NLP Models: a Metamorphic Testing Perspective*, Findings of the ACL, 2022
- Batista et al., *CEG4N: Counter-Example Guided Neural Network Quantization Refinement*, IEEE Transactions on Computer-Aided Design, 2023
- Manino et al., *Towards Global Neural Network Abstractions with Locally-Exact Reconstruction*, Neural Network Journal, 2023

The Oracle Problem

Testing a Black-Box System Requires



- Many test cases (inputs)
- Their ground-truth (outputs)

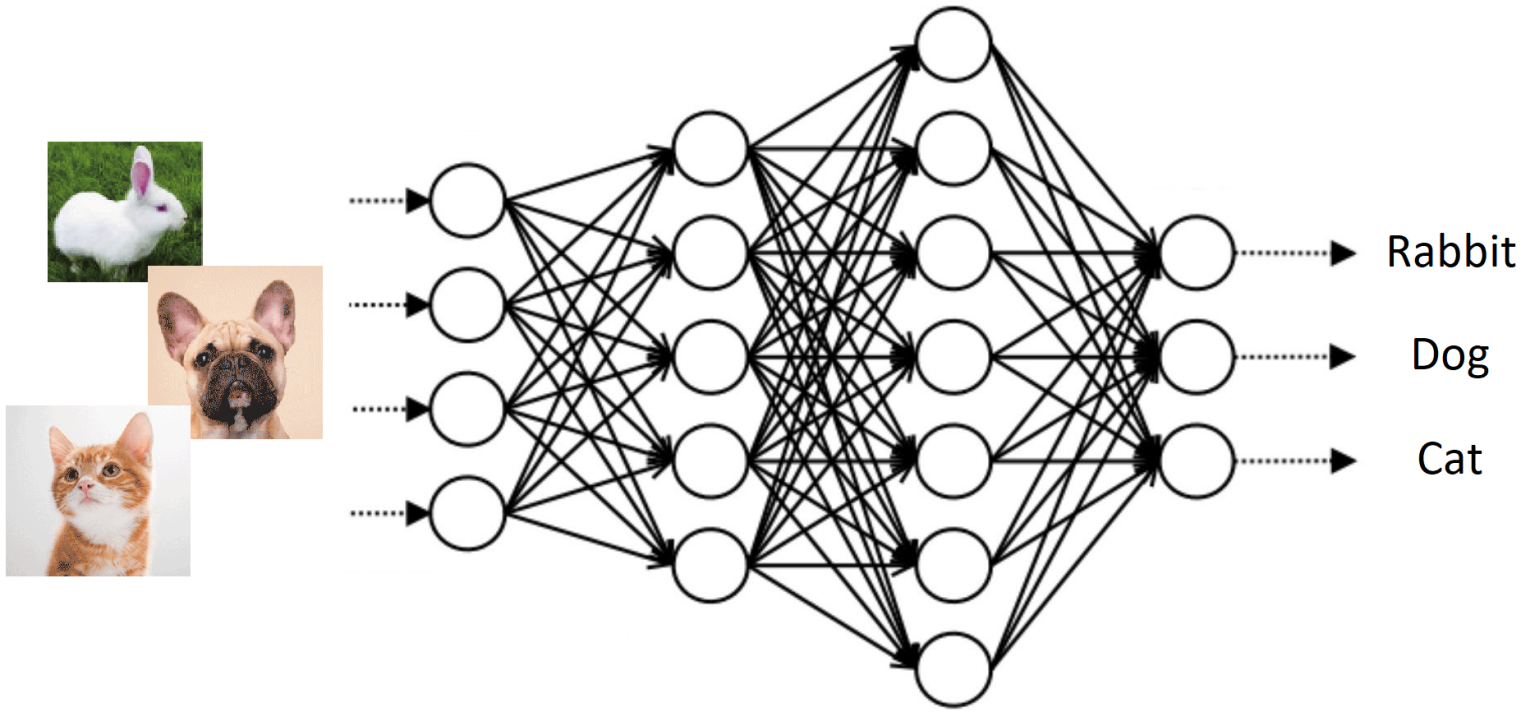
"Exhaustive" Testing Would Require

- The presence of an oracle
- That can gives us the ground-truth
- For **any** possible input

A Safety Paradox

- If such oracle exists, we do not need the black box system!
- This talk: two ML-specific variants of this paradox

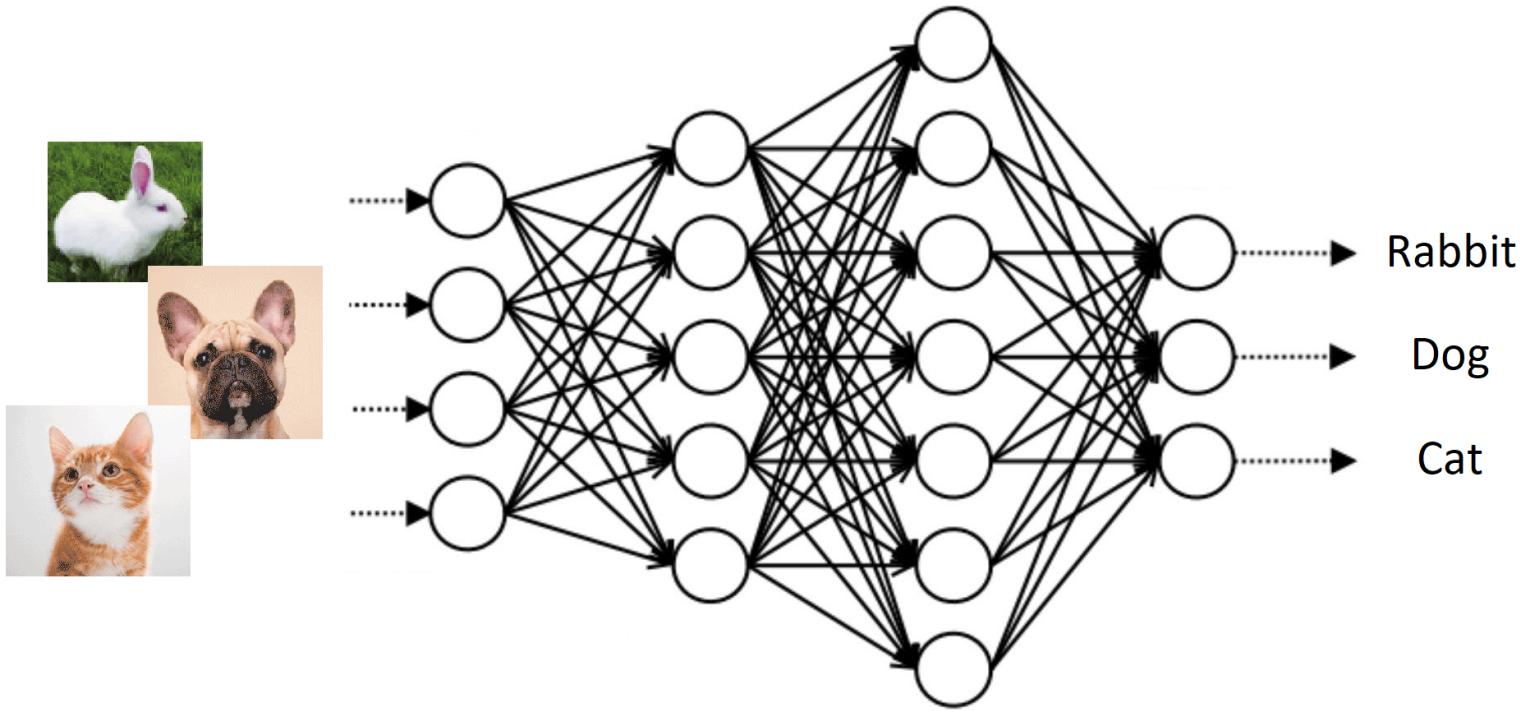
Back to the Basics: The Data Scientist's View



ML "Ingredients"

- A (possibly large) dataset of examples
- A ML model and an algorithm to train it

Back to the Basics: Empirical Risk Minimisation



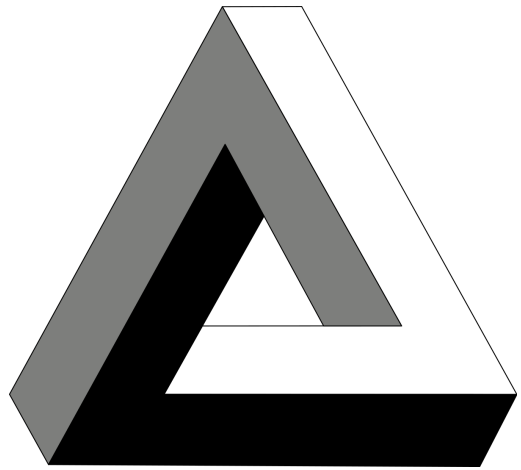
What's The Requirement?

- Minimise the empirical loss $\frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(f(x_i), y_i)$
- That is, mimic the training set in some statistical sense

The Requirements Paradox

No Formal Requirements in ML

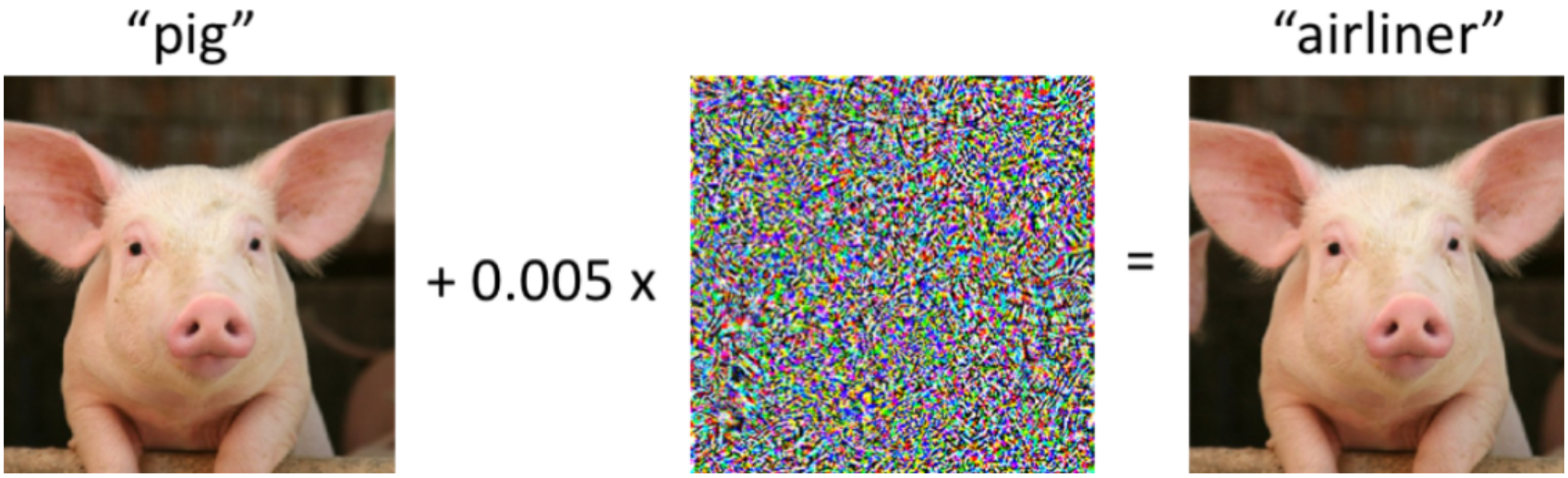
- Minimise the loss function
- Perform "well" on test set
- No constraints on OOD behaviour



A ML Safety Paradox (1)

- If we have a full set of requirements we do not need ML at all
- I.e., just use the oracle

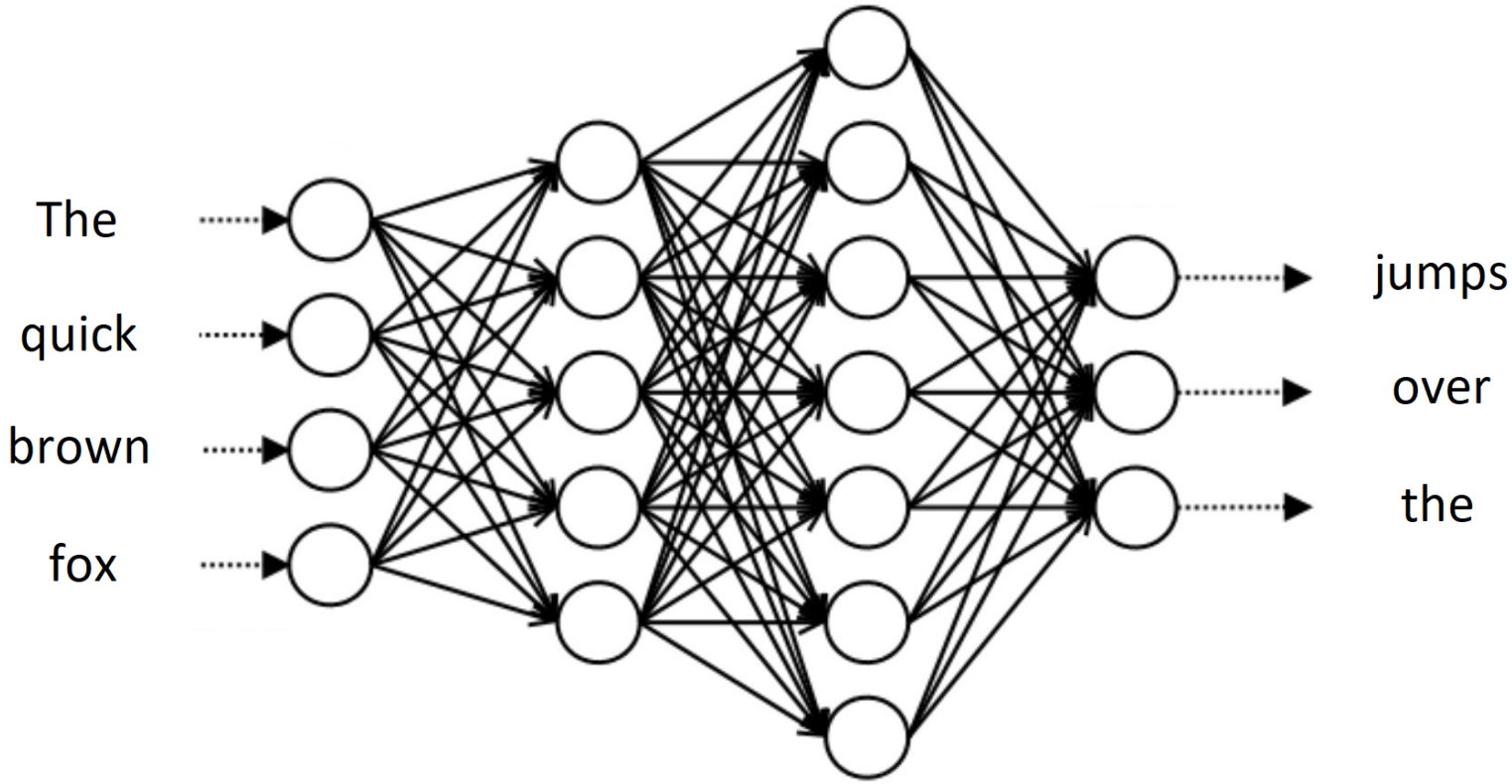
Image robustness



Safety property

- For each input image x with output y
- Any perturbation $x' : ||x' - x|| < \epsilon$ still outputs y
- This property is broken in the example above!

NLP Safety Properties



A Few Crucial Differences

- NLP inputs (tokens) are discrete not continuous
- Rich tradition of linguistic analysis, often grounded in logic
- Recent successes suggest the presence of shallow reasoning

Text robustness (1)

Use the adversarial image paradigm with text?

- Discrete input $\neq$ "continuous" image
- What is an imperceptible noise/perturbation for text?
- Literature between 2017-2018 focuses on this

Example: mizpelling vs sentiment analysis

- Input: *The encore was nice* $\rightarrow$ Positive Review
- Noise: natural typos, synthetic typos, character shuffle
- Result: *Ze EnnCore was niec* $\rightarrow$????

Safety property

- For each sentence x with output sentiment y
- Any semantic-preserving mutation of x still outputs y

Text robustness (2)

More semantic-preserving mutations

- Semantic noise: replace words with synonyms
- Syntactic noise: parse and reorder a sentence
- Sentence compression: parse a sentence, delete sub-tree
- Positive tautology: add "*and true is true*" at the end
- Jabberwocky words: replace peripheral words with nonsense
- ...

Example: Jabberwockies and synonyms vs entailment

- *The French band gave an encore. The musicians played*
  $\rightarrow$ Entailment
- *The Messazovian band gave an encore. The ensemble played*
  $\rightarrow$???? [RoBERTa: Entailment]

Other properties

Beyond robustness

- We can define more complex safety properties
- See Ribeiro *et al*, 2020, *Behavioral Testing of NLP Models*

Example: semantic monotonicity vs sentiment analysis

- Input: *The orchestra was cool, the music not so much*
- Adding "*and you are lame!*" must make it more negative

Example: gender equality vs comprehension

- *John is not the harpist, Mary is. Who is a harpist?*
  $\rightarrow$ Answer: Mary
- *Mary is not the harpist, John is. Who is a harpist?*
  $\rightarrow$ Answer: ????

Popular Safety Requirements

Research Challenge

- Empirical risk minimisation is not strong enough
- We need to augment it with additional requirements

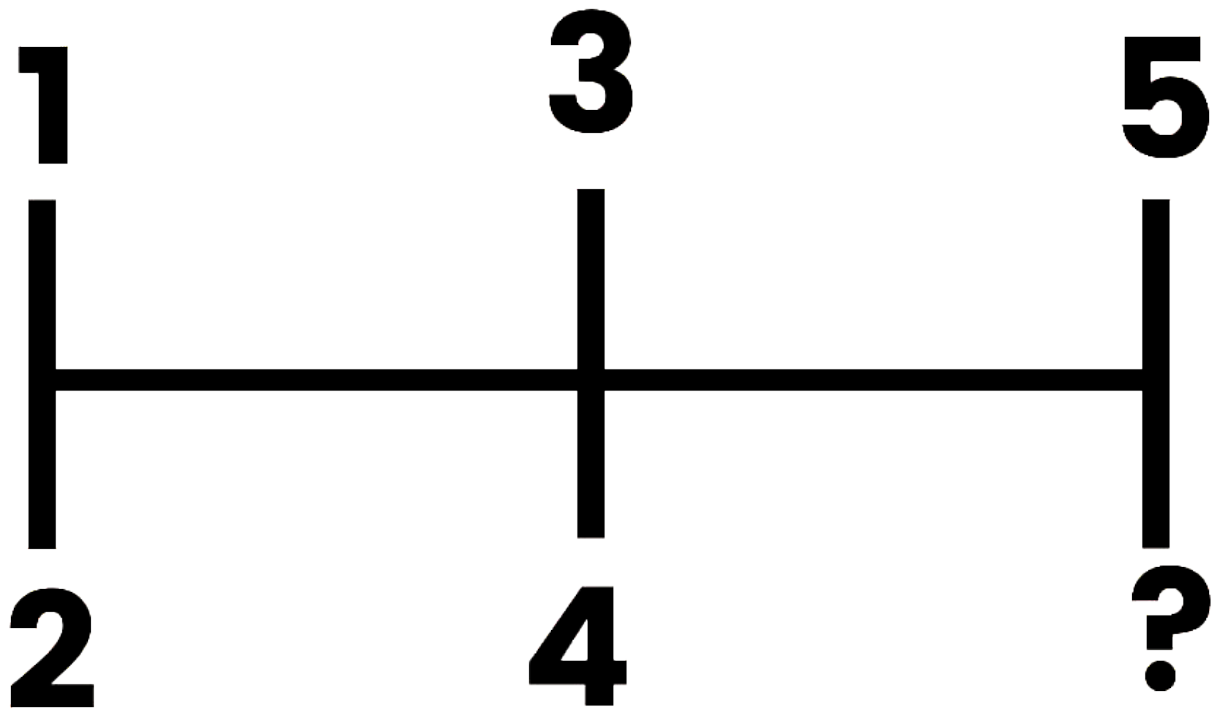
Popular Safety Properties

- Deterministic: robustness*, monotonicity, equivalence, stability
- Probabilistic: robustness*, fairness
- System-Level: privacy-preserving ML, absence of backdoors

A Property of ML Safety Properties (1)

- We only tell the ML system what **not** to do

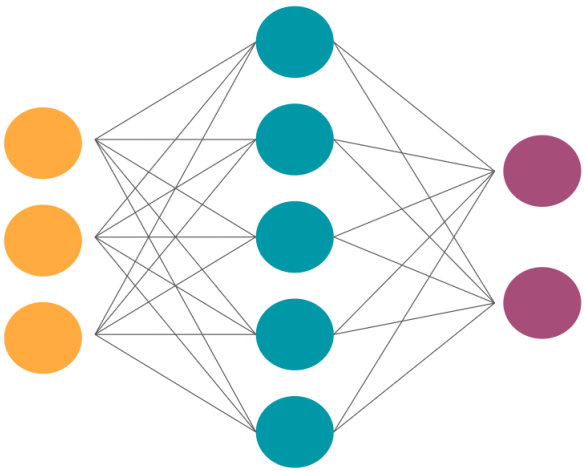
WHAT IS MISSING ?



A software security view (1)

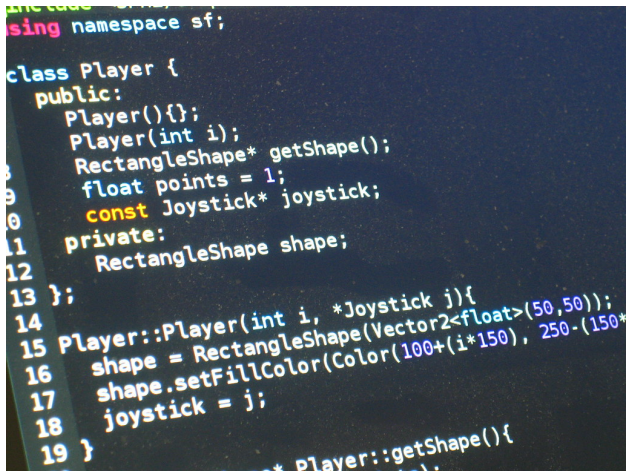
ML testing

- Test set accuracy
- Mutations/perturbations
- Wrong predictions
- Adversarial example
- Adversarial training



Software testing

- Unit testing
- Fuzzed input
- Exceptions/crashes
- Vulnerability
- Debugging



A software security view (2)

Trend towards learning from unlabelled data

- Unsupervised, semi-supervised, self-supervised
- No need for costly dataset annotation

Testing without ground-truth?

- Current paradigms need ground-truth annotations
- In-distribution testing: train-validate-test split
- More recent: out-of-distribution testing, probing

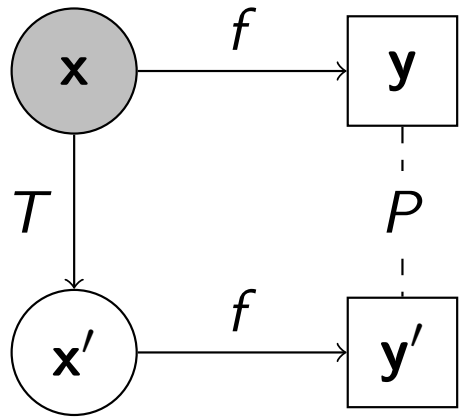
Metamorphic testing!

- Formal definition of input-output behaviour
- Checks whether the NLP model satisfies it
- Less reliance on ground-truth $\implies$ large number of test cases

A formalisation of robustness properties

Notation (input)

- x: The encore was nice
- x': Ze EnnCore was niec
- T: add input noise



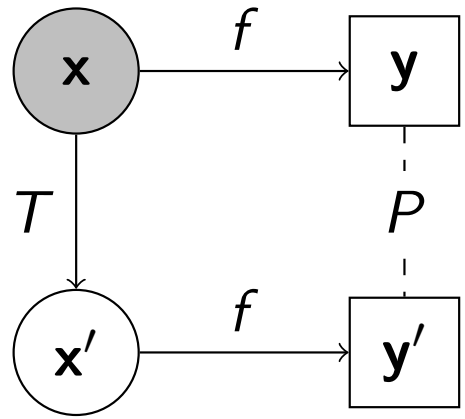
A formalisation of robustness properties

Notation (input)

- x: The encore was nice
- x': Ze EnnCore was niec
- T: add input noise

Notation (ML and output)

- f: neural network
- y: positive/negative sentiment
- y': positive/negative sentiment



A formalisation of robustness properties

Notation (input)

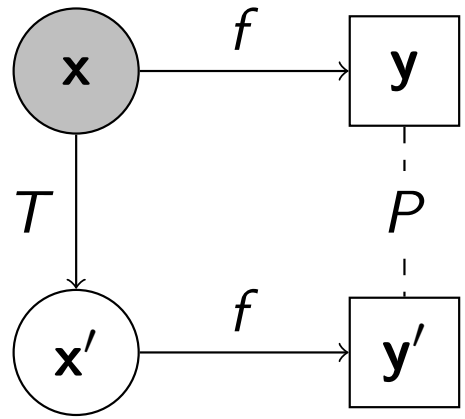
- x: The encore was nice
- x': Ze EnnCore was niec
- T: add input noise

Notation (ML and output)

- f: neural network
- y: positive/negative sentiment
- y': positive/negative sentiment

Notation (relation)

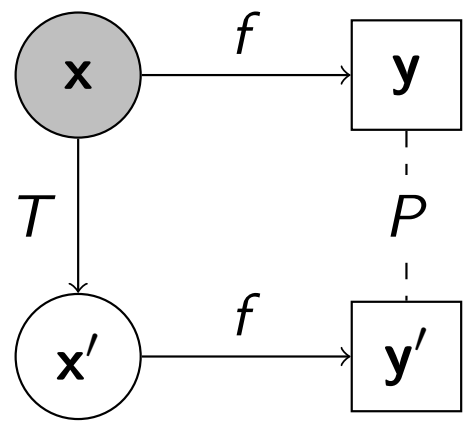
- P: equivalence



Beyond robustness properties

Example: semantic monotonicity vs sentiment analysis

- Input: *The orchestra was cool, the music not so much*
- Adding "*and you are lame!*" must make it more negative



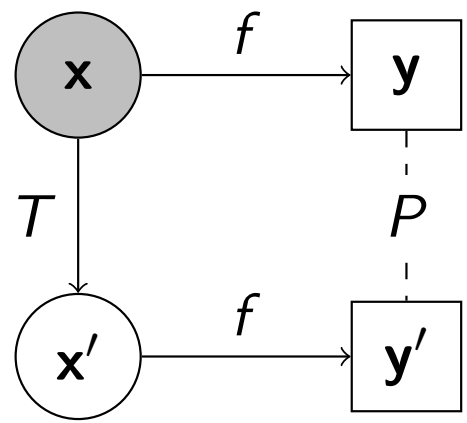
Beyond robustness properties

Example: semantic monotonicity vs sentiment analysis

- Input: *The orchestra was cool, the music not so much*
- Adding "*and you are lame!*" must make it more negative

Example: gender equality vs comprehension

- *John is not the harpist, Mary is. Who is a harpist?*
  $\rightarrow$ Answer: Mary
- *Mary is not the harpist, John is. Who is a harpist?*
  $\rightarrow$ Answer: John



Beyond robustness properties

Example: semantic monotonicity vs sentiment analysis

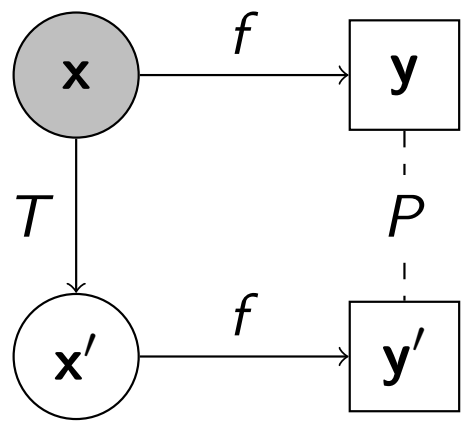
- Input: *The orchestra was cool, the music not so much*
- Adding "*and you are lame!*" must make it more negative

Example: gender equality vs comprehension

- *John is not the harpist, Mary is. Who is a harpist?*
  $\rightarrow$ Answer: Mary
- *Mary is not the harpist, John is. Who is a harpist?*
  $\rightarrow$ Answer: John

Reference

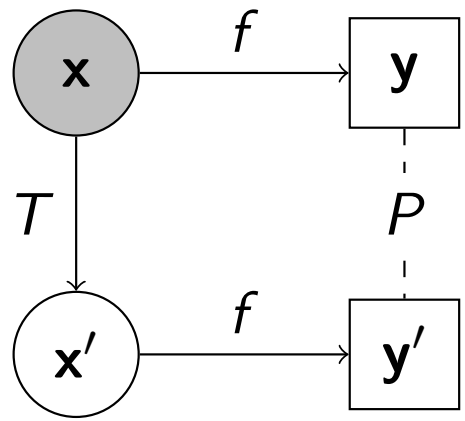
- Ribeiro *et al*, 2020, *Behavioral Testing of NLP Models*



Robustness-like properties (recap)

Main characteristic

- A user-defined transformation T
- A relation P on the (softmax) output
- It must hold for every input x



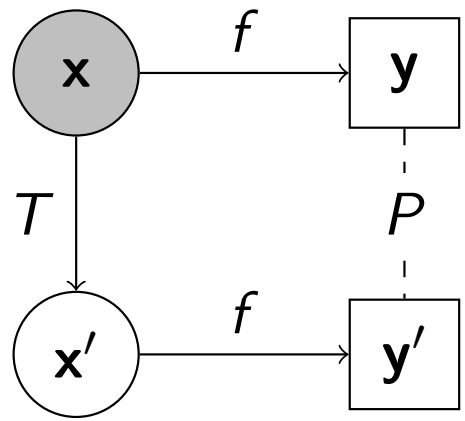
Robustness-like properties (recap)

Main characteristic

- A user-defined transformation T
- A relation P on the (softmax) output
- It must hold for every input x

Verification/testing challenge

- Find inputs x that break the relation P
- aka "counterexamples"



Robustness-like properties (recap)

	Single-input metamorphic relations
Input:	$\mathbf{x}$ = The cat sat on the mat.
	$\mathbf{x}'$ = The pet stood onto the mat.
T:	*replace any word of the input with a synonym.*
P:	$\mathbf{y} = f(\mathbf{x}) \wedge \exists i \, \forall j \neq i \, (y_i > y_j) \wedge (y'_i > y'_j)$

Table: Example of robustness relations from the literature [Li 2017]. Robustness relations belong to the class of single-input relations.

Our claim

- We reviewed all existing metamorphic testing for NLP
- They all test the same basic metamorphic relation
- We name it the *single-input* relation

More NLP safety properties?

A new idea

- What if we consider **pairs** of inputs?

More NLP safety properties?

A new idea

- What if we consider **pairs** of inputs?

Example (step 1): hyponymy relation

- Input x_1: "a tree is a type of plant"
- Input x_2: "a car is a type of vehicle"

More NLP safety properties?

A new idea

- What if we consider **pairs** of inputs?

Example (step 1): hyponymy relation

- Input x_1: "a tree is a type of plant"
- Input x_2: "a car is a type of vehicle"

Example (step 2): context change $\ell(\cdot)$

- Input x_1': "we know that tree is a subset of plant"
- Input x_2': "we know that car is a subset of vehicle"

More NLP safety properties?

A new idea

- What if we consider **pairs** of inputs?

Example (step 1): hyponymy relation

- Input x_1: "a tree is a type of plant"
- Input x_2: "a car is a type of vehicle"

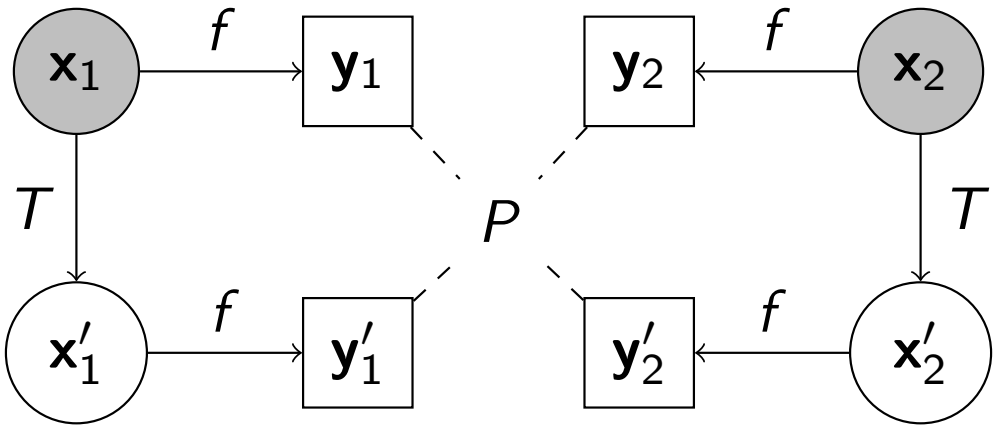
Example (step 2): context change $\ell(\cdot)$

- Input x_1': "we know that tree is a subset of plant"
- Input x_2': "we know that car is a subset of vehicle"

Example (step 3): safety property

- Pick a neural net $f(\cdot)$ that predicts the truth of a statement
- If x_1 happens to be "truer" than x_2
- Then we want x_1' to be "truer" than x_2' as well

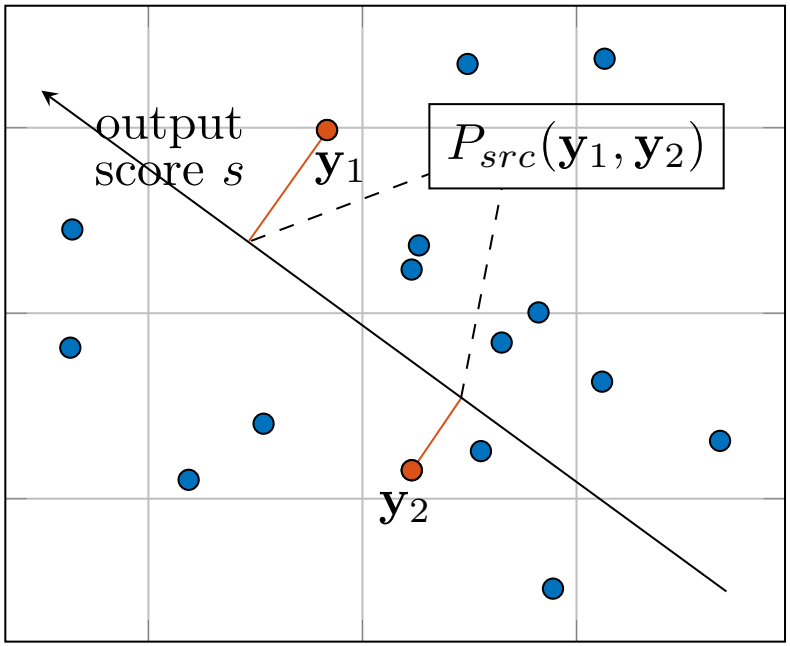
A formalisation of the new NLP properties

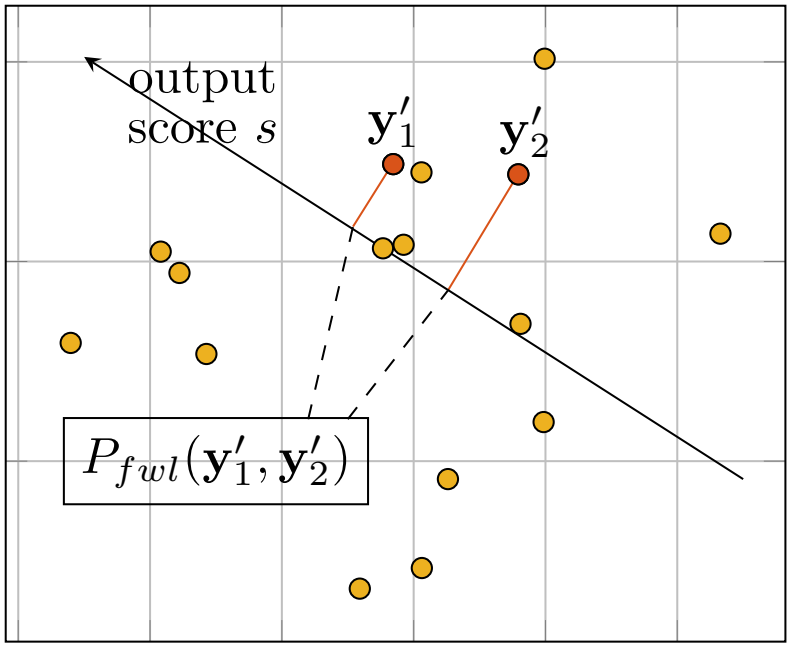


Notation

- T: "a <q> is a type of <r>" becomes "We know that <q> is a subset of <r>"
- P: if $y_1 \geq y_2$ then $y'_1 \geq y'_2$, where $\geq$ means "truer"
- P is an implication $P_{src} \implies P_{fwl}$

Pairwise systematicity: a geometric view

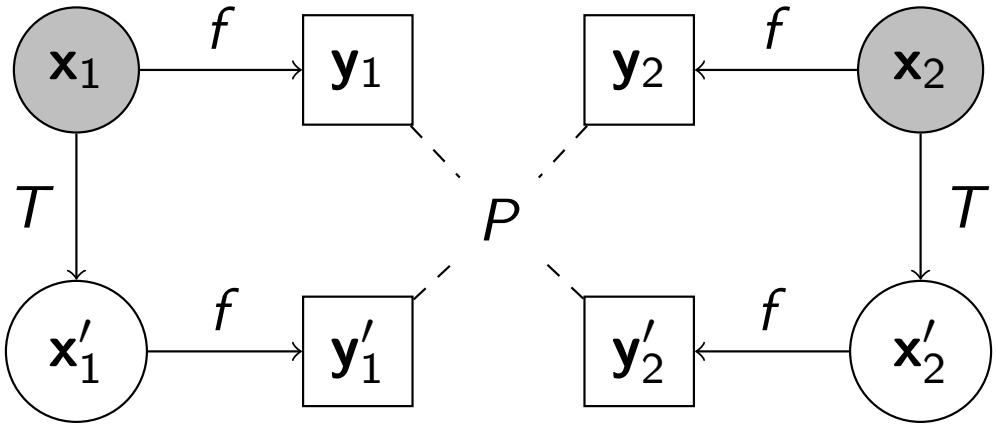




What happens in the embedding space?

- Pairwise systematicity impose implicit constraints!
- The relation on the left needs to match the one on the right

Pairwise systematicity (recap)



Intuition

- Pick two unrelated source inputs $\mathbf{x}_1, \mathbf{x}_2$
- Read the relation between their outputs $\mathbf{y}_1, \mathbf{y}_2$
- Check whether the relation holds after transformation T

Pairwise systematicity: experiment 1

Insertion	Label	Context C	Context D
(pumpkin, vegetable)	leq	We know that pumpkin belongs to the set of vegetable	Pumpkin is a type of vegetable
(animal, shoe)	none	We know that animal belongs to the set of shoe	Animal is a type of shoe
(building, house)	geq	We know that building belongs to the set of house	Building is a type of house

Table: Examples of insertion pairs with hyponymy (leq), hypernymy (geq) and no relation (none), and two contexts C and D .

Binary Target	geq	none	leq	rand
Training Accuracy	0.931	1.000	0.990	0.591
Satisfied Properties	0.881	0.867	0.861	0.639

Table: Ratio of consistent relationships across different contexts vs training accuracy. These were computed out of 1M random pairs.

Pairwise systematicity: experiment 2

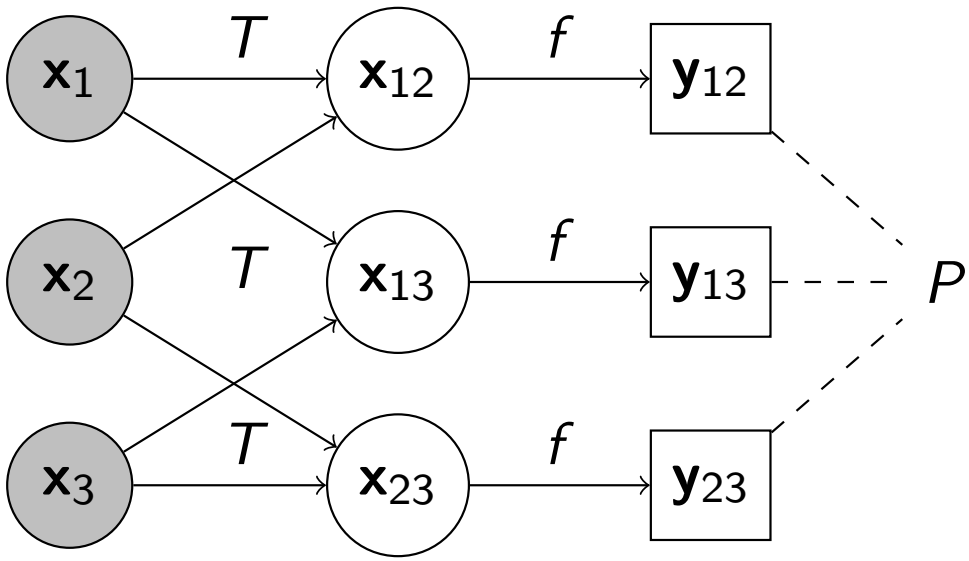
Pairwise systematicity metamorphic relations		
Input:	$\mathbf{x}_1 =$	Light, cute and forgettable.
	$\mathbf{x}_2 =$	A masterpiece four years in the making.
	$\mathbf{x}'_1 =$	Thank you. Light, cute and forgettable.
	$\mathbf{x}'_2 =$	Thank you. A masterpiece four years in the making.
T:	*concatenate the text*	Thank you. *at the beginning of the input.*
P:	$s_{pos}(f(\mathbf{x}_1)) > s_{pos}(f(\mathbf{x}_2)) \iff s_{pos}(f(\mathbf{x}'_1)) > s_{pos}(f(\mathbf{x}'_2))$	

Table: Example of pairwise systematicity relations for sentiment analysis.

Empirical results

- State-of-the-art RoBERTa model for sentiment analysis
- 112M+ relations from a dataset with 11K+ unlabelled entries!
- From 5% to 10% violations depending on T

Contribution: three-way transitivity



Intuition

- Pick three unrelated source inputs $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$
- Create all possible pairs $\mathbf{x}_{ij} = (\mathbf{x}_i, \mathbf{x}_j)$
- Check whether $v(y_{12}) \wedge v(y_{23}) \Rightarrow v(y_{13})$, with boolean v

Contribution: three-way transitivity

Three-way transitivity metamorphic relations	
	$\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3 =$ arrangement | symmetrical | together
Input:	$\mathbf{x}_{12} =$ (arrangement | symmetrical)
	$\mathbf{x}_{23} =$ (symmetrical | together)
	$\mathbf{x}_{13} =$ (arrangement | together)
T:	*choose two words from the source triplet* $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$
P_{syn}:	$v_{syn}\big(f(\mathbf{x}_{12})\big) \wedge v_{syn}\big(f(\mathbf{x}_{23})\big) \implies v_{syn}\big(f(\mathbf{x}_{13})\big)$
P_{hyp}:	$v_{hyp}\big(f(\mathbf{x}_{12})\big) \wedge v_{hyp}\big(f(\mathbf{x}_{23})\big) \implies v_{hyp}\big(f(\mathbf{x}_{13})\big)$

Table: Example of three-way transitivity relations for the lexical relations of synonymy and hypernymy.

Empirical results

- State-of-the-art RoBERTa model for lexical relations
- Cubic number of relations, we pick a sample of them
- From 60% to 80% violations depending on the input language

Summary and future work

Contributions

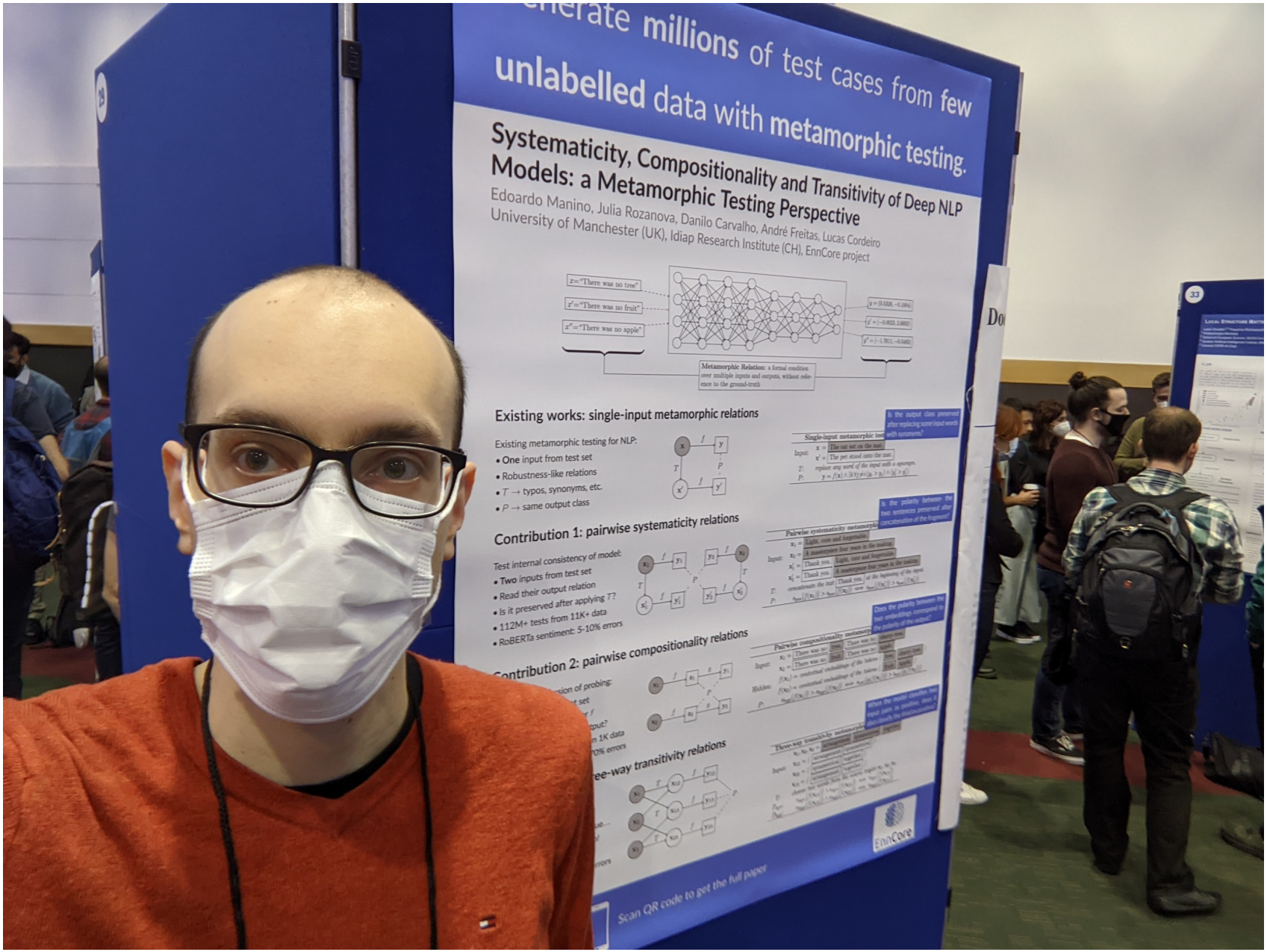
- Taxonomy of existing work (single-input relations)
- Novel graphical notation for metamorphic relations
- Pairwise systematicity metamorphic relation
- Pairwise compositionality metamorphic relation
- Three-way transitivity metamorphic relation

Practical impact

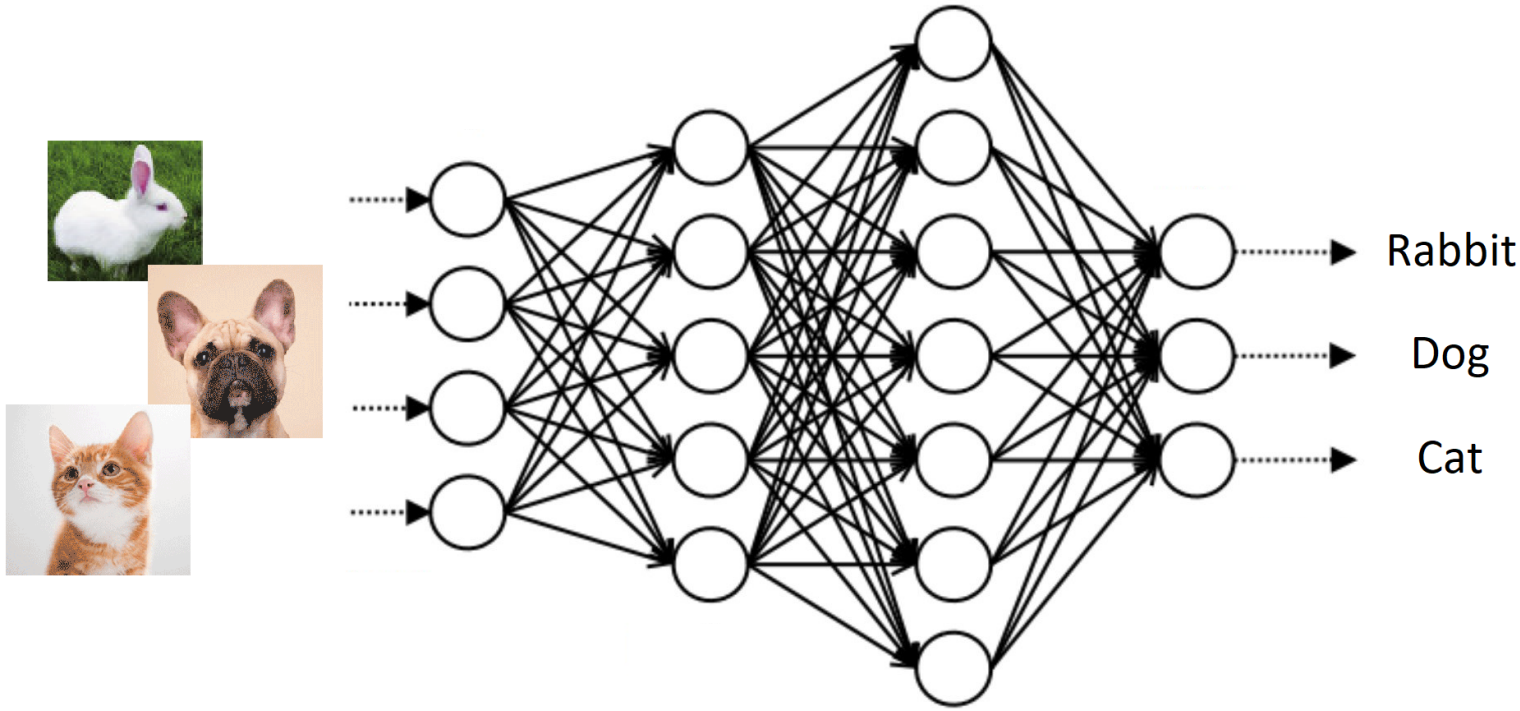
- Metamorphic testing can work with unlabelled test sets
- Our relations generate a quadratic/cubic number of test cases

Future work

- *Montague Semantics and Modifier Consistency Measurement in Neural Language Models*, submitted to EACL'23


millions of test cases from few
unlabelled data with metamorphic testing.
Systematicity, Compositionality and Transitivity of Deep NLP Models: a Metamorphic Testing Perspective
Edoardo Manino, Julia Rozanova, Danilo Carvalho, André Freitas, Lucas Cordeiro
University of Manchester (UK), Idiap Research Institute (CH), EnnCore project
Existing works: single-input metamorphic relations
Existing metamorphic testing for NLP:
• One input from test set
• Robustness-like relations
• $T \rightarrow$ typos, synonyms, etc.
• $P \rightarrow$ same output class
Contribution 1: pairwise systematicity relations
Test internal consistency of model:
• Two inputs from test set
• Read their output relation
• Is it preserved after applying T?
• 112M+ tests from 11K+ data
• RoBERTa sentiment: 5-10% errors
Contribution 2: pairwise compositionality relations
Three-way transitivity relations
Scan QR code to get the full paper
EnnCore

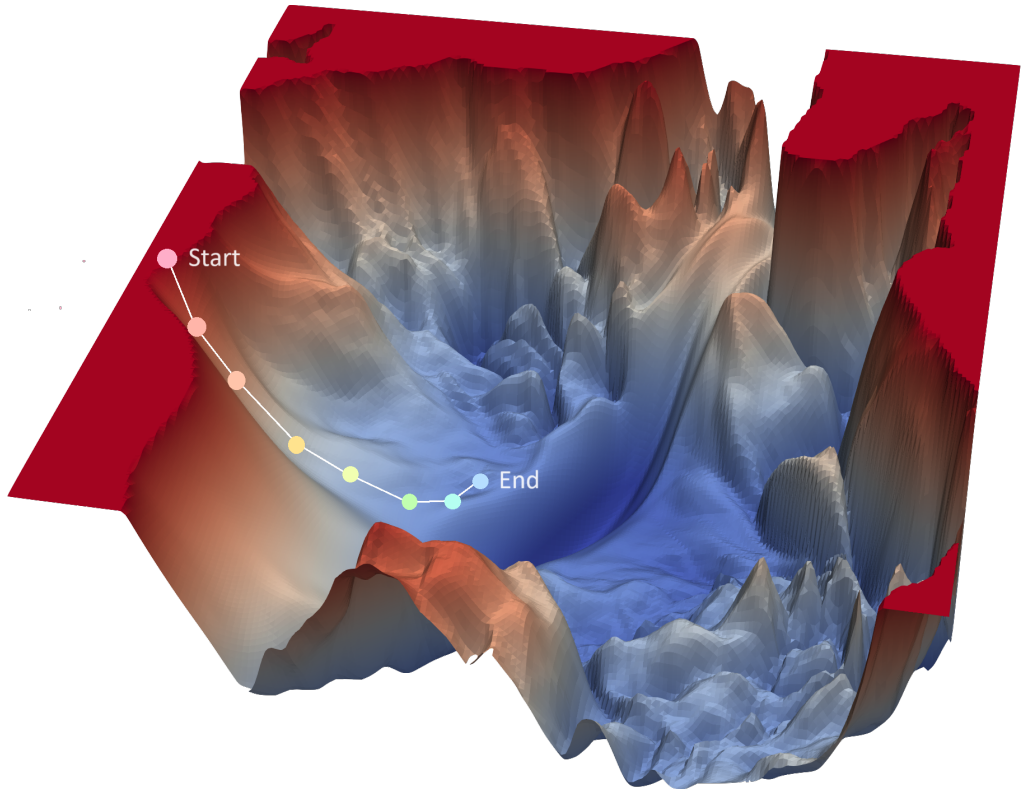
Back to the Basics: Universal Approximation



Why do neural networks perform well?

- In most cases, they are *universal approximators*
- That is, there exists a set of parameters (weights, biases)
- Such that the network is able to fit arbitrary training data

Back to the Basics: Gradient Descent



There is a catch though...

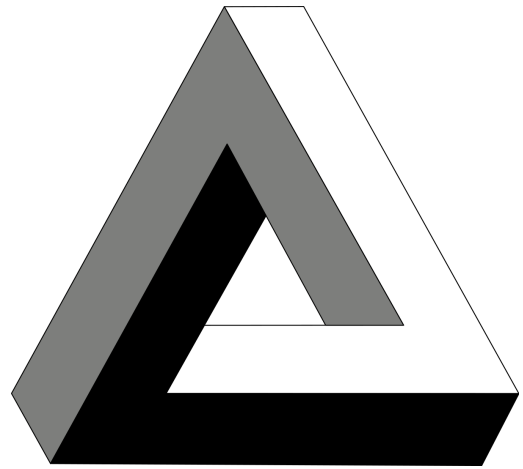
- There exist an optimal set of parameters (weights, biases)
- But gradient descent might never find it!


THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

The Equivalence Paradox

NNs have High Redundancy

- In order to train well
- We use (very) large nets
- To maximize capacity



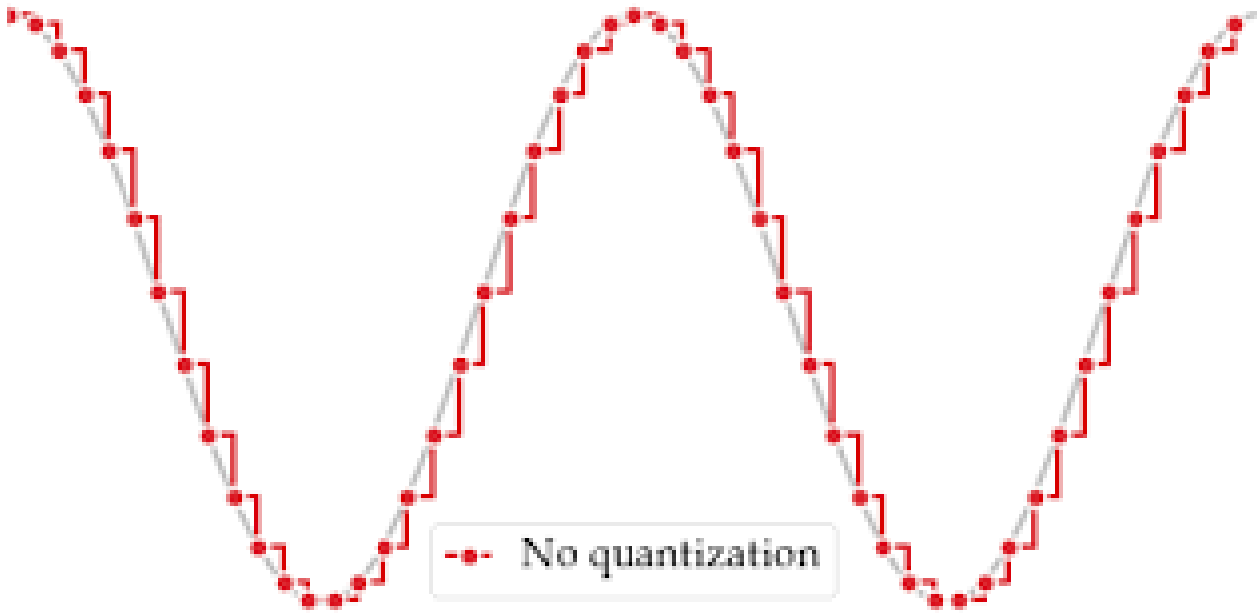
Compression techniques

- After training we want to reduce the network size
- E.g., pruning, quantisation, distillation

A ML Safety Paradox (2)

- Inference with the original NN (the oracle!) is expensive
- The compressed network may introduce unwanted behaviour

Neural Network Quantization (1)



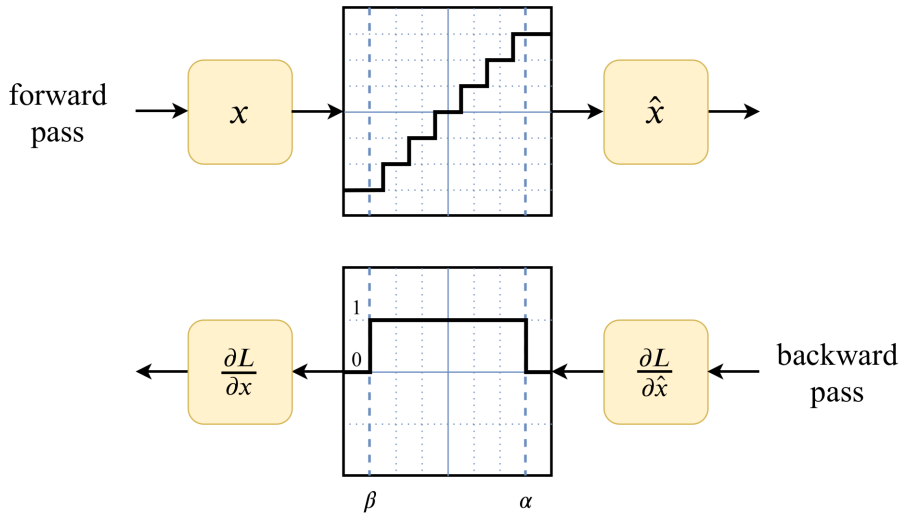
Why quantization?

- Old technique from signal processing/information theory
- Reduce memory footprint (e.g., store 8-bit weights)
- Reduce latency/power (full integer computation)

Neural Network Quantization (2)

Many Strategies

- Dynamic
- Post-Training
- Q-Aware Training
- Non-Uniform
- ...



Main differences

- Whether the weights and/or the activations are quantized
- Whether the weights are fine-tuned after quantization
- Whether the quantization is uniform (e.g., int 8-bit)

Quantisation and NN Equivalence

		Number of bits												
Safety Prop.		6	7	8	9	10	11	12	13		28	29	30	31	32
Set.	R_{40}	S	S	F	S	S	S	S	S	...	S	S	S	S	S
	R_{50}	S	S	F	F	F	F	F	F	...	F	F	F	F	S
Vers.	R_{20}	S	F	S	S	S	S	S	S	...	S	S	S	S	S
	R_{30}	S	F	S	S	S	S	S	S	...	S	S	S	S	S
	R_{40}	S	F	S	F	F	F	S	S	...	S	S	S	S	S
	R_{50}	S	F	F	F	F	F	F	F	...	F	F	F	F	F
Virg.	R_{20}	S	F	S	S	S	S	S	S	...	S	S	S	S	S
	R_{30}	S	F	S	S	S	S	S	S	...	S	S	S	S	S
	R_{40}	S	F	S	S	F	S	S	S	...	S	S	S	S	S
	R_{50}	S	F	F	F	F	F	F	F	...	F	F	F	F	F

Table: Effects of quantization on the safety of a NN trained on Iris data.

Effects of Quantisation

- Even if the accuracy does not drop, the behaviour may change
- Can we deploy *safe* quantized network?

CEG4N: Counterexample-Guided NN Quantisation

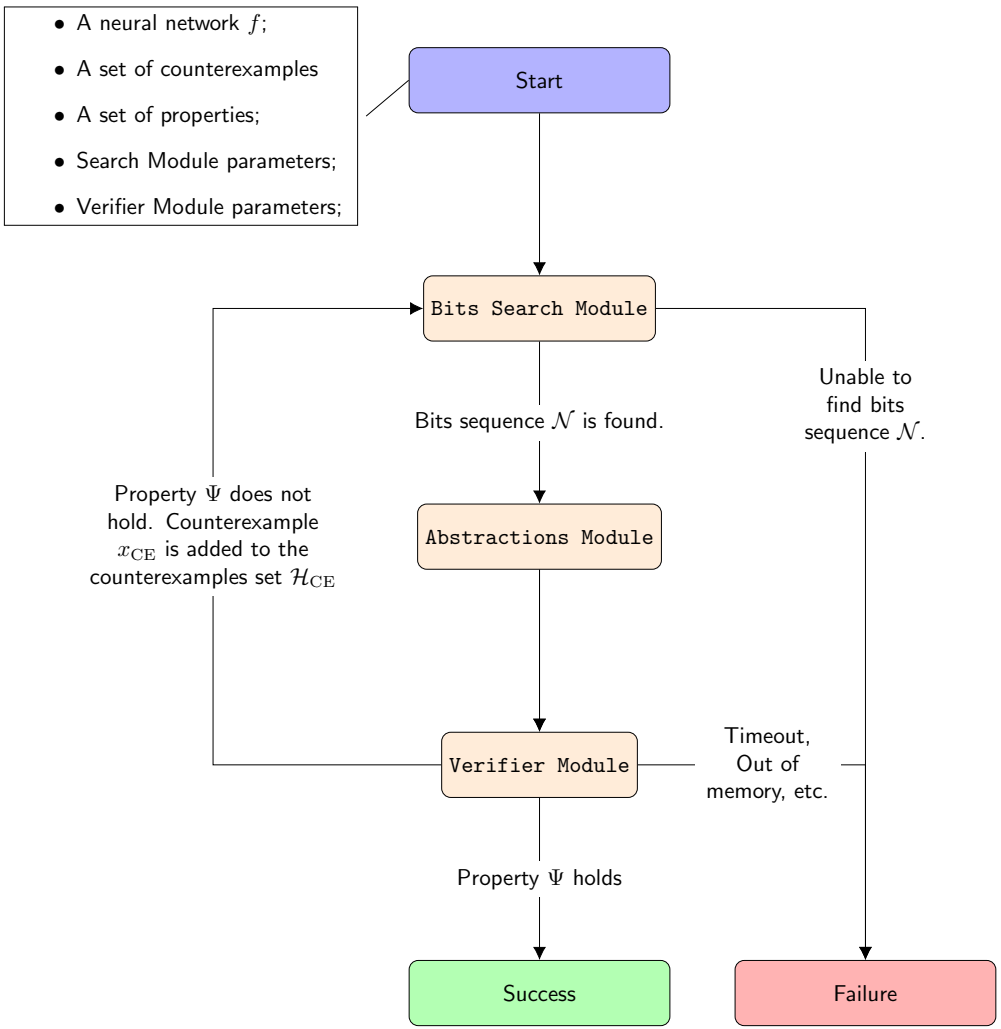
- in Batista et al., TCAD 2023

Quantisation

- Genetic algorithm
- Minimise bits
- Test equivalence

Verification

- Verify equivalence
- If not, generate counterexample
- Augment testset
- Repeat



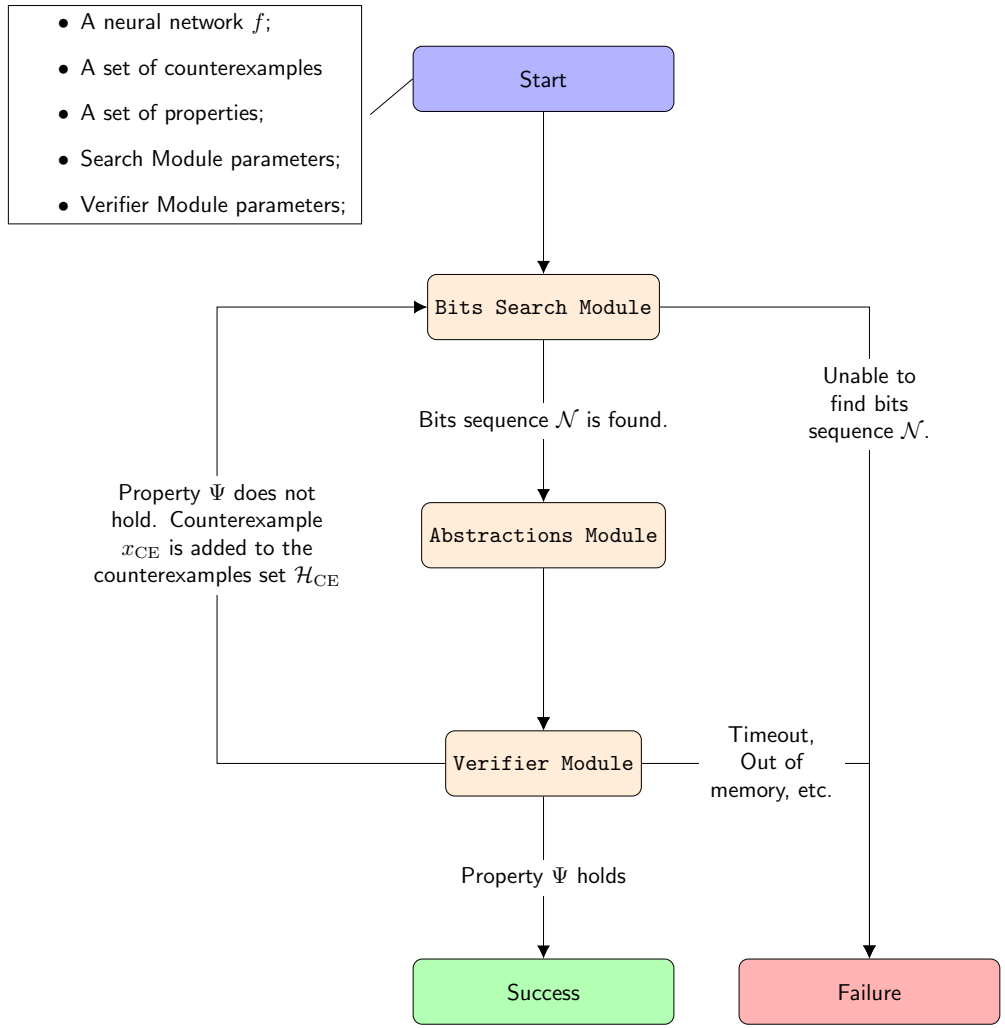
CEG4N: Lessons Learned

Equivalence

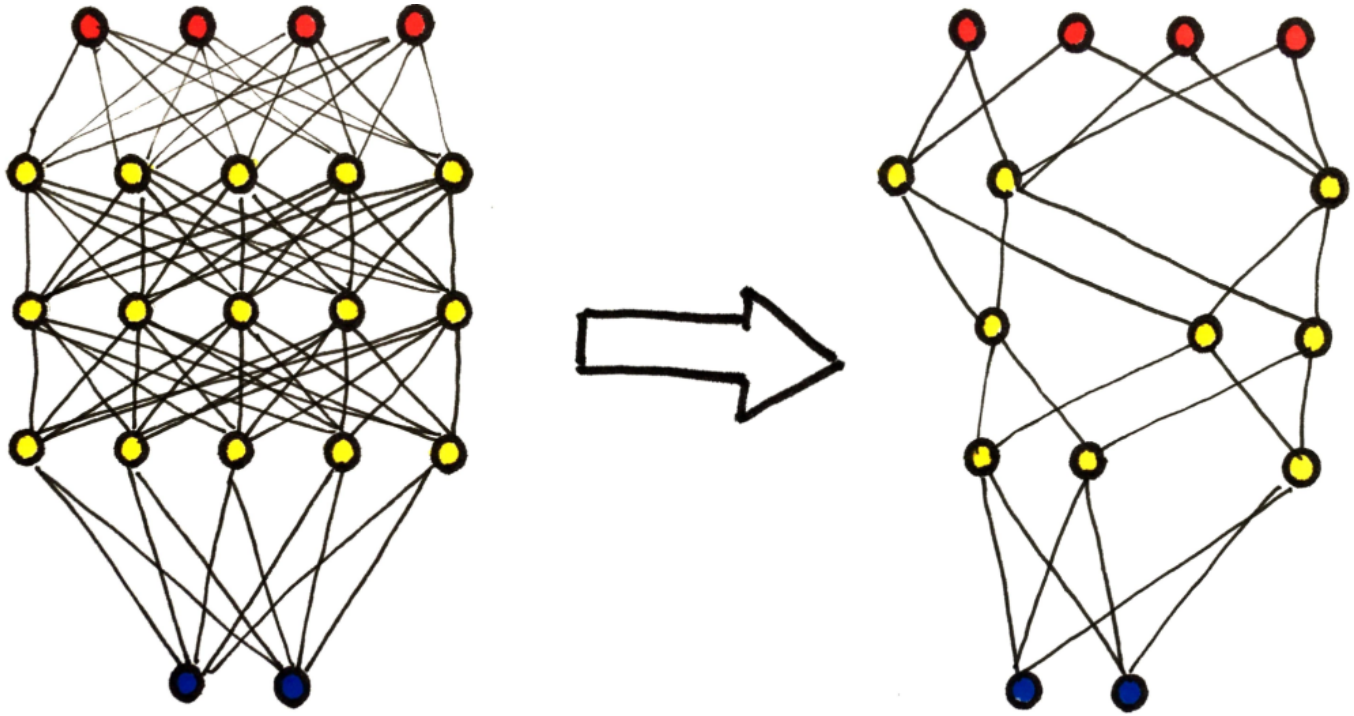
- Different definitions
- Same output class or error bound?
- No correlation with accuracy and robustness

Scalability

- Verification is slow
- But only few iterations are needed



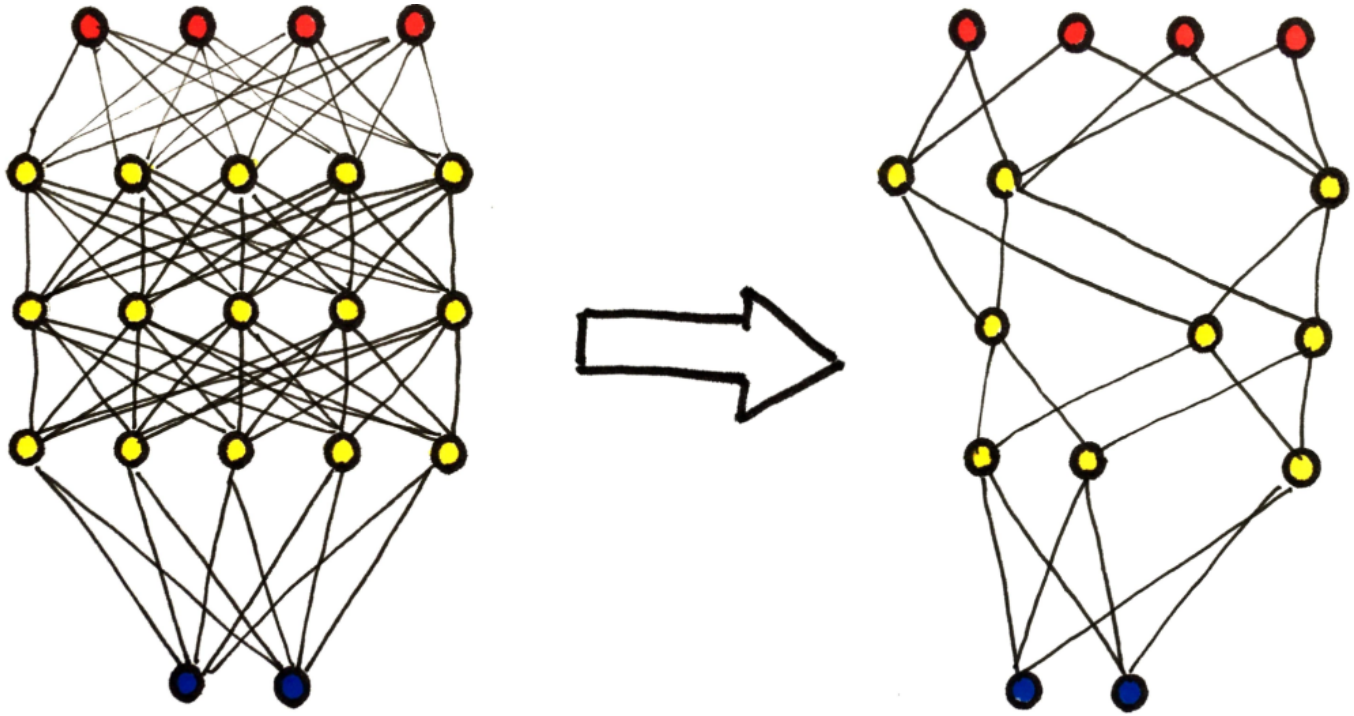
Pruning (1)



Why Pruning?

- Neural networks are highly redundant
- Remove whole neurons and connections
- Original goal: reduce latency/power

Pruning (2)



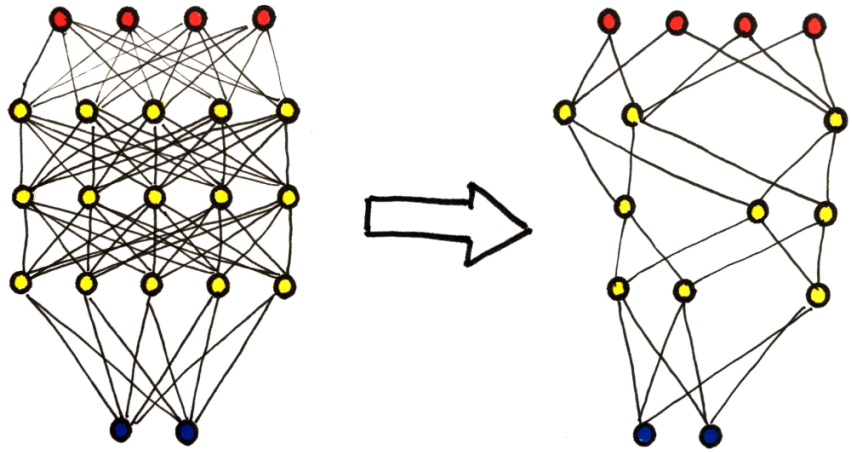
Pruning for verification

- Neural network verification does not scale well
- Can we use pruning to reduce the size of the problem?
- Only if the smaller model is an *abstraction* of the original one

Pruning ≡ Global NN Abstraction

Our "Pruning" Plan

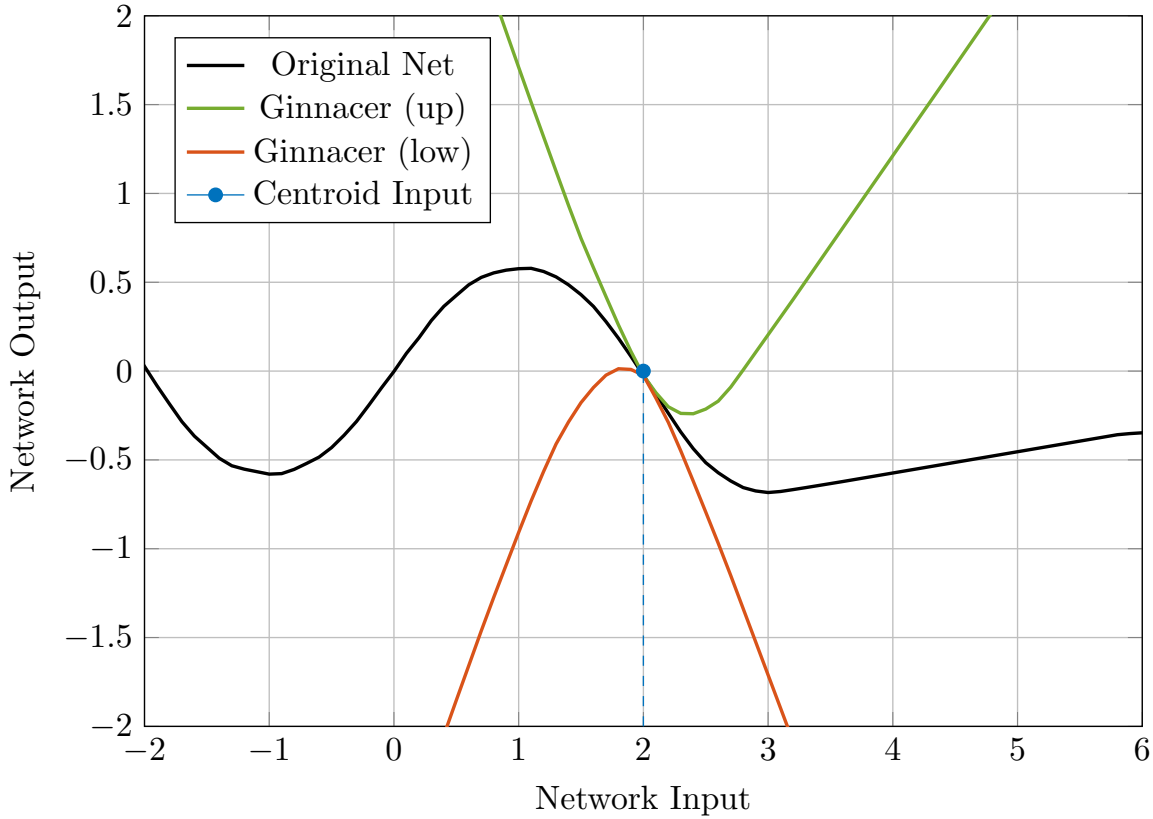
- Make network smaller
- Verify smaller model
- "Transfer" result to original model



Global neural network abstractions

- Our plan works if "pruning" keeps certified error bounds
- Key trick: merge similar neurons, keep max/min weights
- Literature: Prabhakar (NeurIPS 2019), Elboher (CAV 2020)
- Problem: error bounds are huge

Towards Global Abstractions with Local Reconstruction



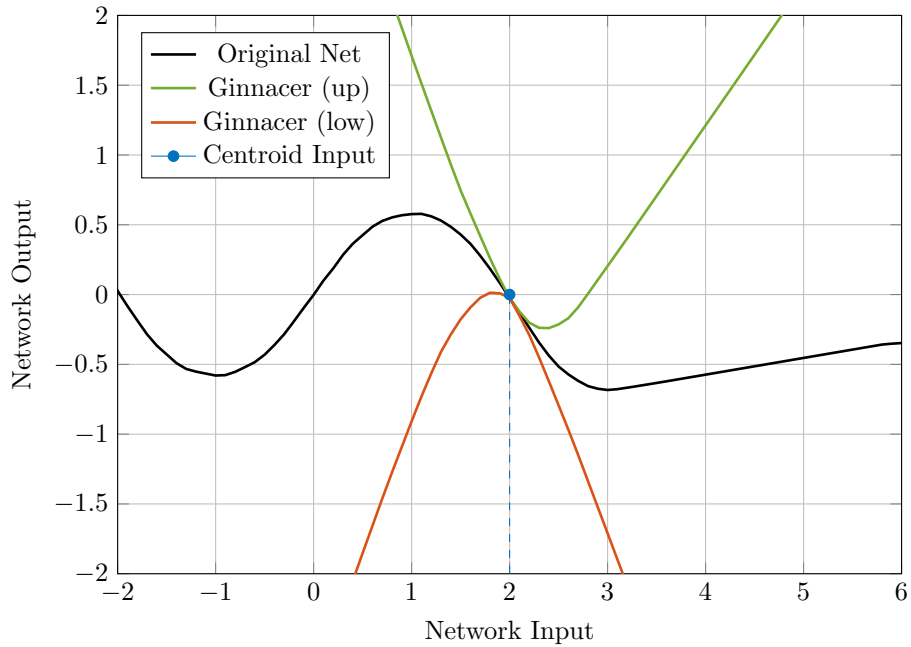
Our GINNACER Algorithm

- Do not merge if the activation state changes at the centroid
- The upper and lower bounds are ReLU NNs themselves!
- in Manino et al., Neural Network Journal, 2023

GINNACER: Lessons Learned

Tightness

- Beats existing global abstractions
- Competitive with local ones
- Is it enough?



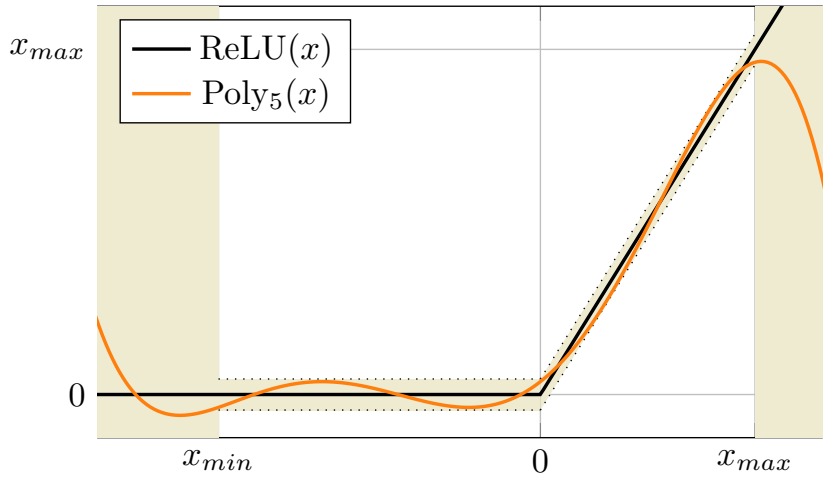
Future

- Simple rules like pruning and merging are limited
- Required to reason about multiple layers
- Subtle trade-off between abstraction and solving

Other NN Transformations

Private Inference

- Run neural networks
- On encrypted data!
- Uses poly activations
- Equivalence problem



- in Manino et al., FoMLAS 2023 (CAV workshop)

Convert neural networks to C code

- Microcontrollers benefit from standalone, compilable code
- Requires off-the-shelf software verification
- But it is not as easy as it sounds. . .
- in Manino et al., AFRiTS 2023 (SBMF workshop)

Summary

Requirements Paradox

- Formalise expectations of NLP system users
- Contribution: metamorphic definition of linguistic properties

Equivalence Paradox

- Compressed NN may exhibit unwanted behaviour
- Contribution (1): safe design of quantized NNs
- Contribution (2): global NN abstraction (formal "pruning")
- Contribution (3): other transformations

My Collaborators

- João Batista, Iury Bessa, Danilo Carvalho, Lucas C. Cordeiro, Eddie de Lima Filho, André Freitas, Bernardo Magri, Rafael Sá Menezes, Mustafa Mustafa, Julia Rozanova, Fedor Shmarov, Xidan Song