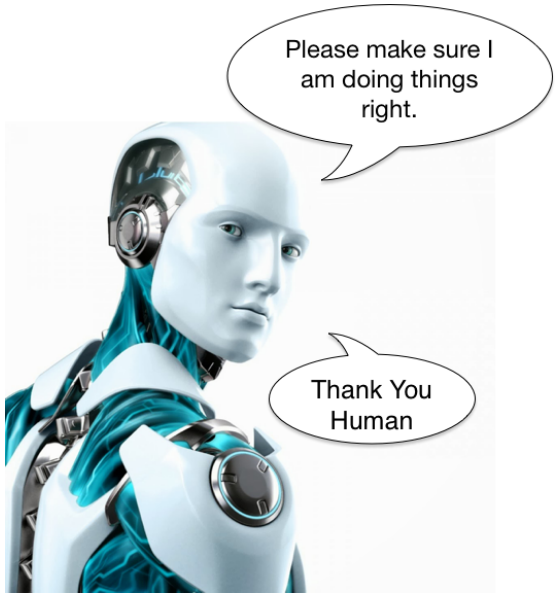
MANCHESTER 1824
The University of Manchester

Deep Neural Networks: Verification

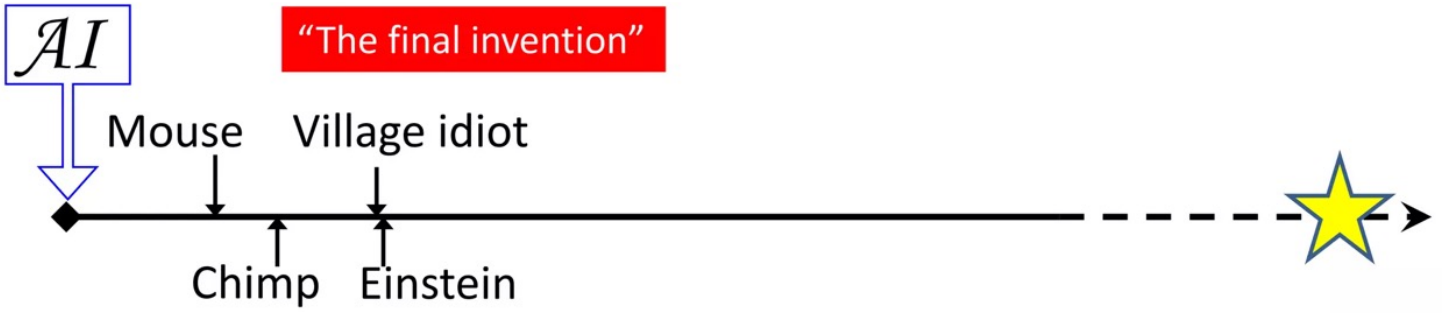


MANCHESTER 1824
The University of Manchester

What is AI?

"Theorem-proving and equation-solving are by now so well established that they are hardly regarded as AI anymore."

— Superintelligence: Paths, Dangers, Strategies

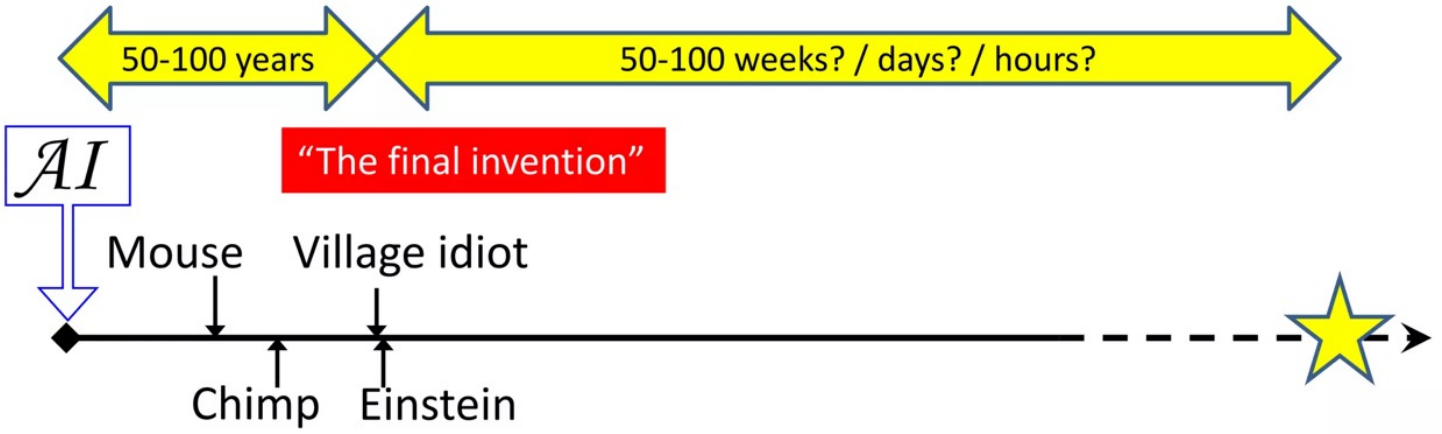


MANCHESTER 1824
The University of Manchester

What is AI?

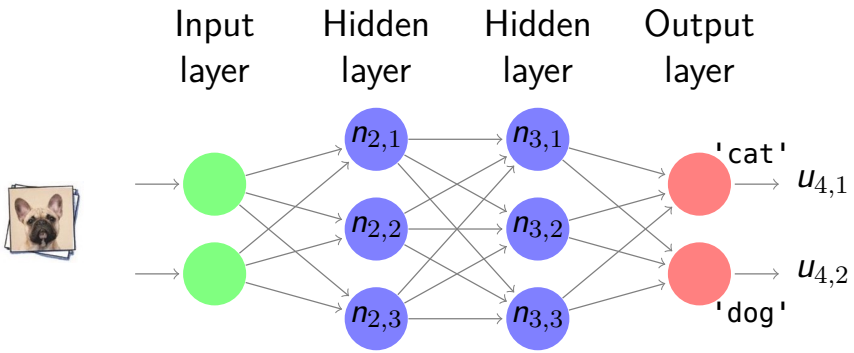
"Theorem-proving and equation-solving are by now so well established that they are hardly regarded as AI anymore."

— Superintelligence: Paths, Dangers, Strategies



MANCHESTER 1824
The University of Manchester

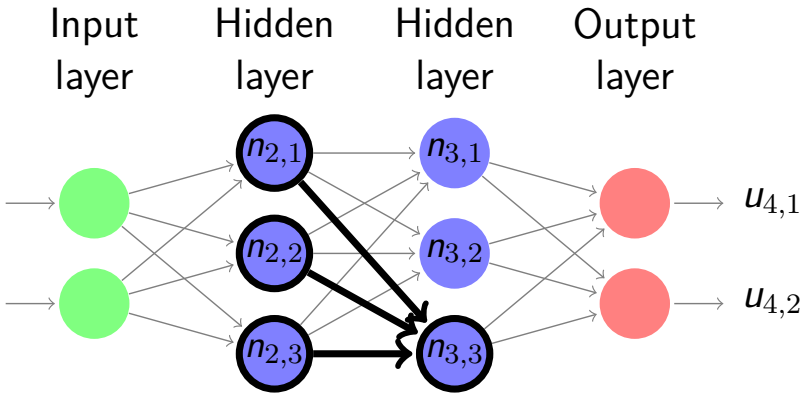
Deep Neural Networks (DNNs)



$$label = \text{argmax}_{1 \leq l \leq s_K} u_{K,l}$$

MANCHESTER 1824
The University of Manchester

Deep Neural Networks (DNNs)



$$label = \text{argmax}_{1 \le l \le s_K} u_{K,l}$$

1) neuron activation value

$$u_{k,i} = b_{k,i} + \sum_{1 \le h \le s_{k-1}} w_{k-1,h,i} \cdot v_{k-1,h}$$

weighted sum plus a bias;

w,b are parameters learned

2) rectified linear unit (ReLU):

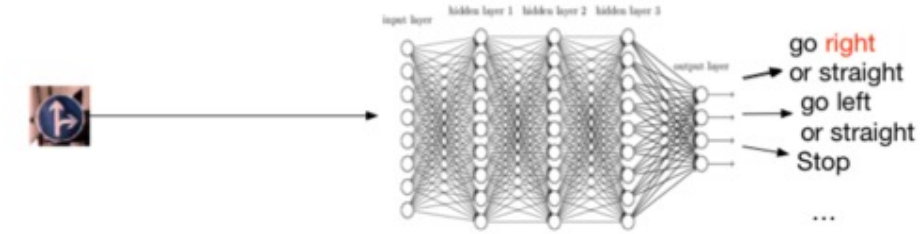
$$v_{k,i} = \max\{u_{k,i}, 0\}$$

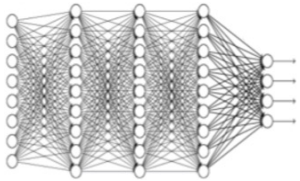
MANCHESTER 1824
The University of Manchester

The Good, Bad and the Ugly

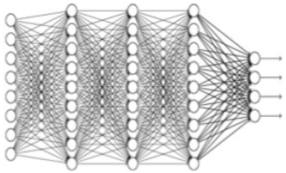
MANCHESTER 1824
The University of Manchester

The Good





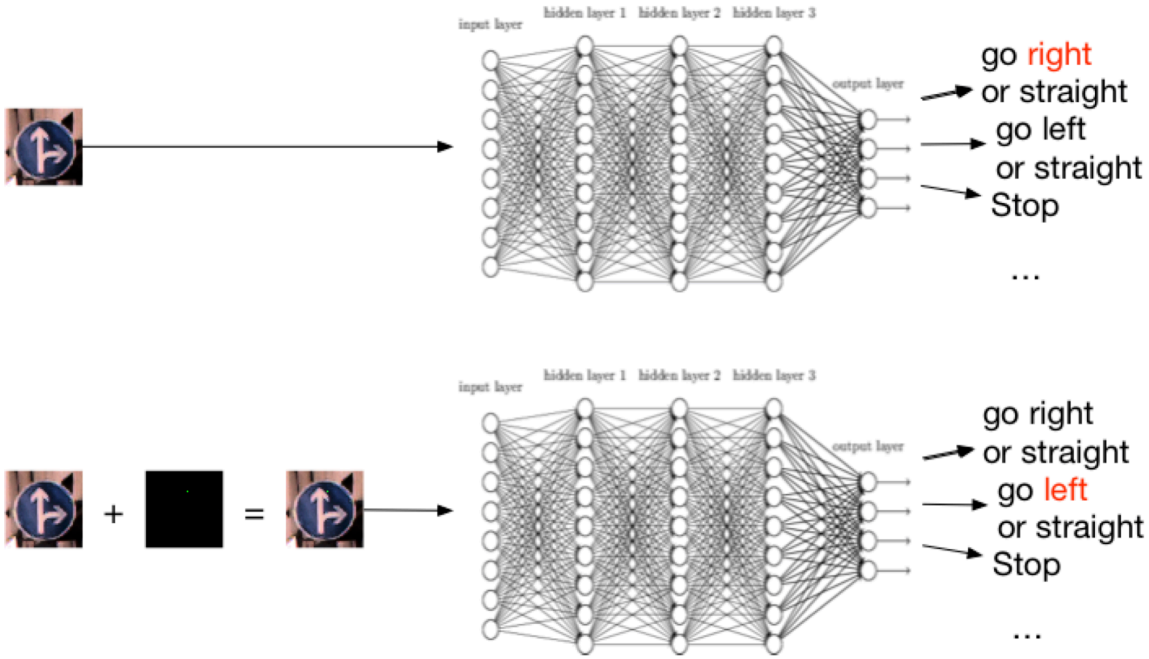
'red panda'



‘11-45-G’

MANCHESTER 1824
The University of Manchester

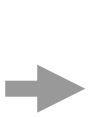
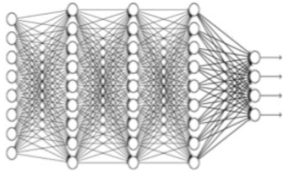
Adversarial Examples



- An adversarial example refers to specially crafted input which is designed to look "normal" to humans but causes misclassification to a machine learning model.

MANCHESTER 1824
The University of Manchester

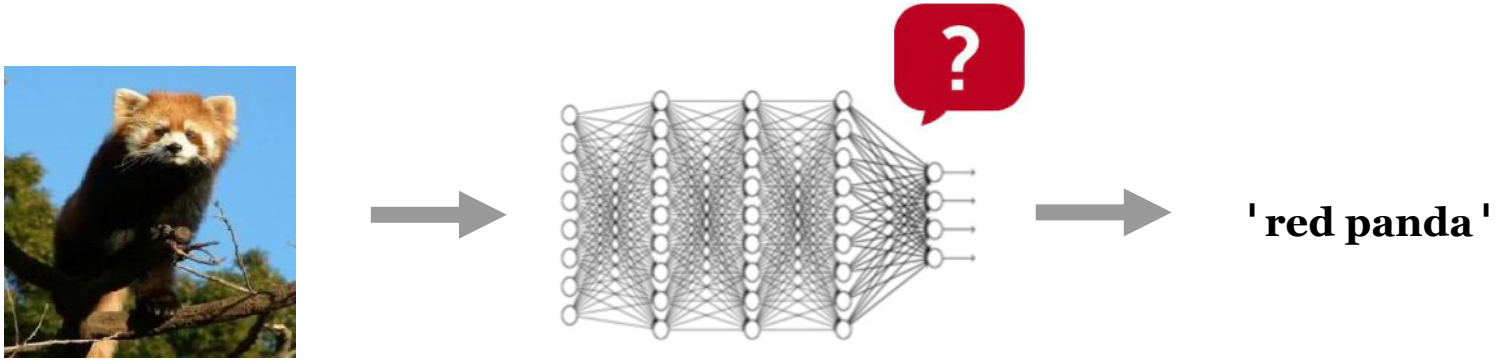
Backdoor

- Performant models, with backdoors that produce inference errors when presented with input containing a trigger defined by the adversary

MANCHESTER 1824
The University of Manchester

Explainability



MANCHESTER 1824
The University of Manchester

Security in DNNs

- How to verify that a DNN is robust enough to adversarial examples?
- How to verify that a DNN is free of backdoor?
- How to explain a DNN?

MANCHESTER 1824
The University of Manchester

Adversarial Robustness

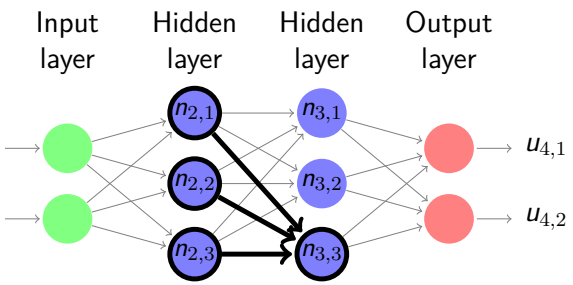
- Let ***N*** be a neural network and ***N(x)*** be the prediction on an input **x**.
- Given a neural The neural network is said to be adversarial robust, subject to a perturbation upper bound ***r***, if for any ***0<δ<=r***:

$$N(x+\delta) = N(x)$$

MANCHESTER 1824
The University of Manchester

ESBMC

DNN as a program



$$label = \text{argmax}_{1 \le l \le s_K} u_{K,l}$$

1) neuron activation value

$$u_{k,i} = b_{k,i} + \sum_{1 \le h \le s_{k-1}} w_{k-1,h,i} \cdot v_{k-1,h}$$

weighted sum plus a bias;

w,b are parameters learned

2) rectified linear unit (ReLU):

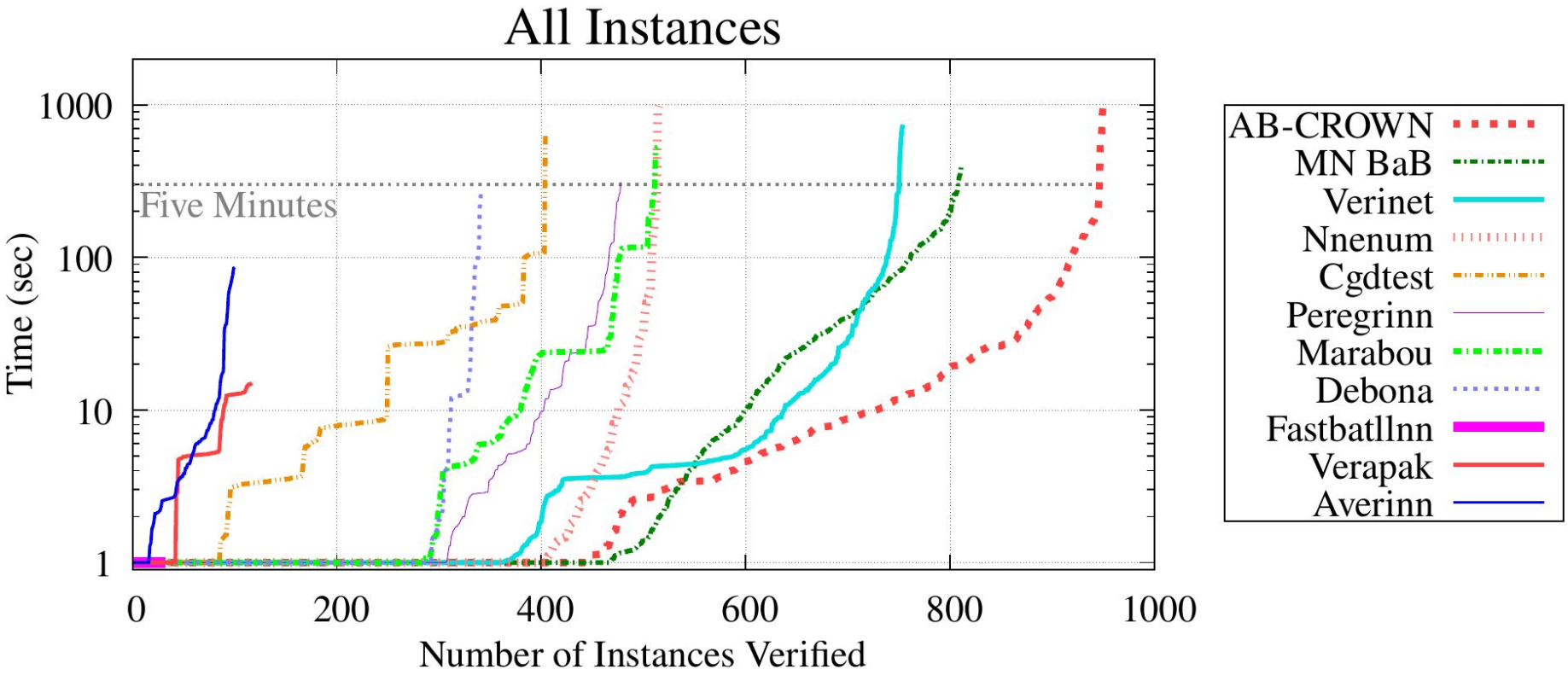
$$v_{k,i} = \max\{u_{k,i}, 0\}$$

```
...

// 1) neuron activation value
double u_{k,i} = b_{k,i};
for (unsigned h = 1; h <= s_{k-1}; h += 1)
{
   u_{k,i} += w_{k-1,h,i} * v_{k-1,h};
}

double v_{k,i} = 0;

// 2) ReLU
if (u_{k,i} > 0)
{
    v_{k,i} = u_{k,i};
}

...
```

https://github.com/theyoucheng/DLTT

MANCHESTER 1824
The University of Manchester

VNN-COMP: Verification of Neural Networks Competition

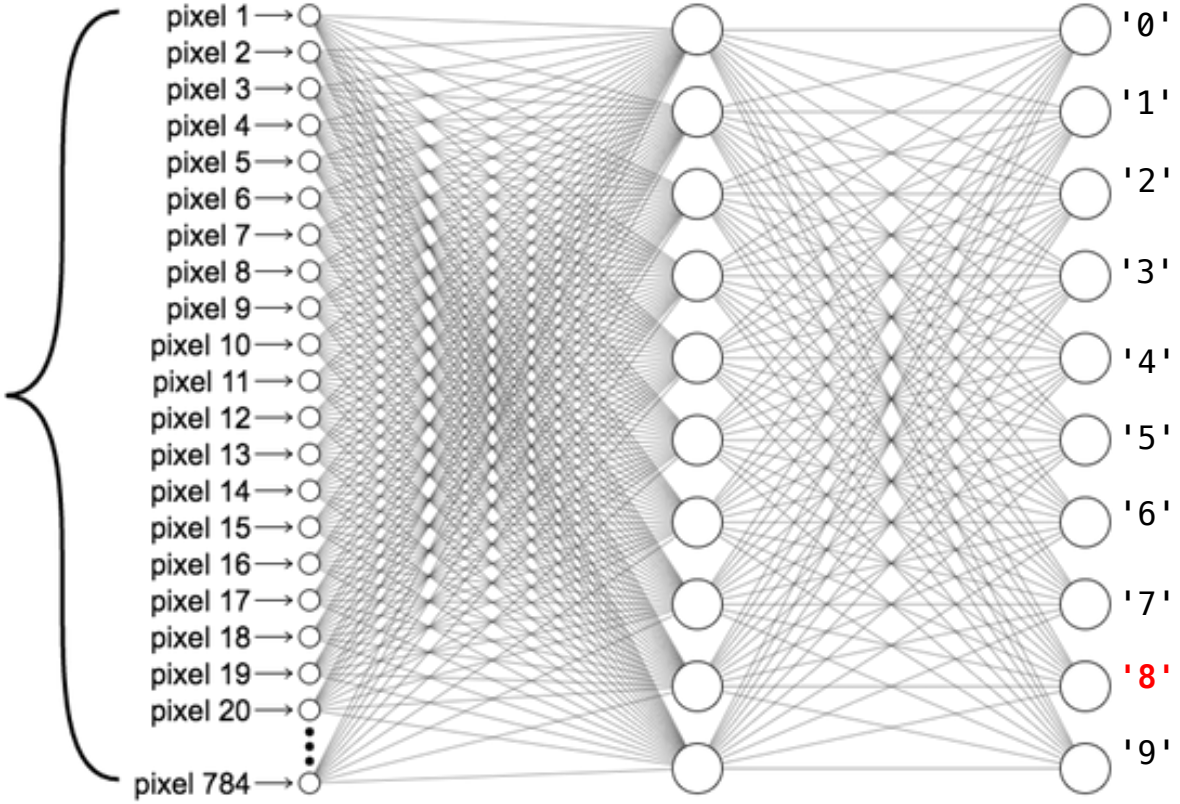


https://sites.google.com/view/vnn2022

MANCHESTER 1824
The University of Manchester

'8' -> '5'

MNIST



MANCHESTER 1824
The University of Manchester

References

- Sun, Youcheng, Min Wu, Wenjie Ruan, Xiaowei Huang, Marta Kwiatkowska, and Daniel Kroening. "Concolic testing for deep neural networks." Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering. 2018.

- Sun, Youcheng, Muhammad Usman, Divya Gopinath, and Corina S. Păsăreanu. "VPN: Verification of Poisoning in Neural Networks." In Software Verification and Formal Methods for ML-Enabled Autonomous Systems: 5th International Workshop (FoMLAS), 2022.

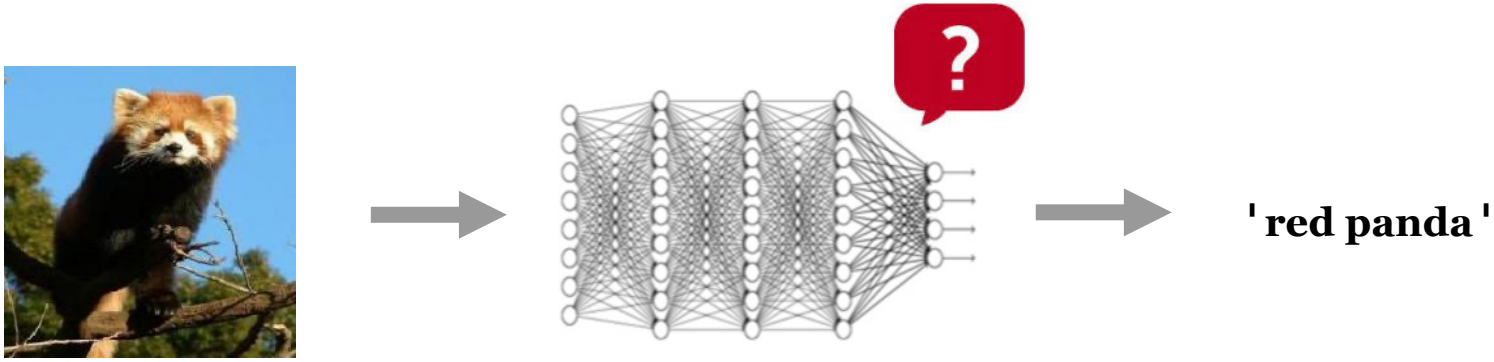
- Sun, Youcheng, Hana Chockler, Xiaowei Huang, and Daniel Kroening. "Explaining image classifiers using statistical fault localization." In European Conference on Computer Vision (ECCV) 2020

MANCHESTER 1824
The University of Manchester

Deep Neural Networks: Explanation

MANCHESTER 1824
The University of Manchester

Why



MANCHESTER 1824
The University of Manchester

Software fault localisation

```
int main() {
  int input1, input2, input3; // C1
  int least = input1;
  int most = input1;

  if (most < input2)
    most = input2; // C2

  if (most < input3)
    most = input3; // C3

  if (least > input2)
    most = input2; // C4 (bug)

  if (least > input3)
    least = input3; // C5

  assert(least <= most);
}
```

MANCHESTER 1824
The University of Manchester

Software fault localisation

```
int main() {
  int input1, input2, input3; // C1
  int least = input1;
  int most = input1;

                              // C2

  if (most < input3)
    most = input3; // C3

  if (least > input2)
    most = input2; // C4 (bug)

                              // C5

  assert(least <= most);
}
```

input1	input2	input3	C2	C3	C4	C5	Assert
200	100	300	**0**	**1**	**1**	**0**	**False**

MANCHESTER 1824
The University of Manchester

Software fault localisation

```
int main() {
  int input1, input2, input3; // C1
  int least = input1;
  int most = input1;

                              // C2

                              // C3

  if (least > input2)
    most = input2; // C4 (bug)

  if (least > input3)
    least = input3; // C5

  assert(least <= most);
}
```

input1	input2	input3	C2	C3	C4	C5	Assert
200	100	300	**0**	**1**	**1**	**0**	**False**
300	100	200	**0**	**0**	**1**	**1**	**False**

MANCHESTER 1824
The University of Manchester

Software fault localisation

```
int main() {
  int input1, input2, input3; // C1
  int least = input1;
  int most = input1;
                              // C2
                              // C3
                              // C4 (bug)
  if (least > input3)
    least = input3; // C5
  assert(least <= most);
}
```

input1	input2	input3	C2	C3	C4	C5	Assert
200	100	300	**0**	**1**	**1**	**0**	**False**
300	100	200	**0**	**0**	**1**	**1**	**False**
300	300	200	**0**	**0**	**0**	**1**	**True**

MANCHESTER 1824
The University of Manchester

Software fault localisation

```
int main() {
  int input1, input2, input3; // C1
  int least = input1;
  int most = input1;

  if (most < input2)
    most = input2; // C2

  [redacted]

  [redacted] (bug)

  [redacted]

  assert(least <= most);
}
```

input1	input2	input3	C2	C3	C4	C5	Assert
200	100	300	**0**	**1**	**1**	**0**	**False**
300	100	200	**0**	**0**	**1**	**1**	**False**
300	300	200	**0**	**0**	**0**	**1**	**True**
100	300	200	**1**	**0**	**0**	**0**	**True**

MANCHESTER 1824
The University of Manchester

Software fault localisation

```
int main() {
  int input1, input2, input3; // C1
  int least = input1;
  int most = input1;

  if (most < input2)
    most = input2; // C2

  [redacted]

  [redacted] (bug)

  [redacted]

  assert(least <= most);
}
```

input1	input2	input3	C2	C3	C4	C5	Assert
200	100	300	**0**	**1**	**1**	**0**	**False**
300	100	200	**0**	**0**	**1**	**1**	**False**
300	300	200	**0**	**0**	**0**	**1**	**True**
100	300	200	**1**	**0**	**0**	**0**	**True**
…							

MANCHESTER 1824
The University of Manchester

Spectrum

- $\langle a^s_{ep}, a^s_{ef}, a^s_{np}, a^s_{nf} \rangle$

To count the number of times the statement *s* is executed (e) or not executed (n) on passing (p) and on failing (f) tests.

input1	input2	input3	C2	C3	C4	C5	Assert
200	100	300	**0**	**1**	**1**	**0**	**False**
300	100	200	**0**	**0**	**1**	**1**	**False**
300	300	200	**0**	**0**	**0**	**1**	**True**
100	300	200	**1**	**0**	**0**	**0**	**True**
…							

a^s_{ep} is the number of tests that passed and executed s

$$a^{C2}_{ep} = 1$$
$$a^{C3}_{ep} = 0$$
$$a^{C4}_{ep} = 0$$
$$a^{C5}_{ep} = 1$$

MANCHESTER 1824
The University of Manchester

Spectrum

- $\langle a^s_{ep}, a^s_{ef}, a^s_{np}, a^s_{nf} \rangle$

To count the number of times the statement *s* is executed (e) or not executed (n) on passing (p) and on failing (f) tests.

input1	input2	input3	C2	C3	C4	C5	Assert
200	100	300	**0**	**1**	**1**	**0**	**False**
300	100	200	**0**	**0**	**1**	**1**	**False**
300	300	200	**0**	**0**	**0**	**1**	**True**
100	300	200	**1**	**0**	**0**	**0**	**True**
…							

a^s_{ef} is the number of tests that failed and executed s

$a^{C2}_{ef} = ?$
$a^{C3}_{ef} = ?$
$a^{C4}_{ef} = ?$
$a^{C5}_{ef} = ?$

MANCHESTER 1824
The University of Manchester

Spectrum

- $\langle a^s_{ep}, a^s_{ef}, a^s_{np}, a^s_{nf} \rangle$

To count the number of times the statement *s* is executed (e) or not executed (n) on passing (p) and on failing (f) tests.

input1	input2	input3	C2	C3	C4	C5	Assert
200	100	300	**0**	**1**	**1**	**0**	**False**
300	100	200	**0**	**0**	**1**	**1**	**False**
300	300	200	**0**	**0**	**0**	**1**	**True**
100	300	200	**1**	**0**	**0**	**0**	**True**
…							

a^s_{ef} is the number of tests that failed and executed s

$a^{C2}_{ef} = 0$
$a^{C3}_{ef} = 1$
$a^{C4}_{ef} = 2$
$a^{C5}_{ef} = 1$

MANCHESTER 1824
The University of Manchester

Spectrum

- $\langle a^s_{ep}, a^s_{ef}, a^s_{np}, a^s_{nf} \rangle$

To count the number of times the statement *s* is executed (e) or not executed (n) on passing (p) and on failing (f) tests.

input1	input2	input3	C2	C3	C4	C5	Assert
200	100	300	**0**	**1**	**1**	**0**	**False**
300	100	200	**0**	**0**	**1**	**1**	**False**
300	300	200	**0**	**0**	**0**	**1**	**True**
100	300	200	**1**	**0**	**0**	**0**	**True**
…							

a^s_{np} is the number of tests that passed and not executed s

$$a^{C2}_{np} = 1$$
$$a^{C3}_{np} = 2$$
$$a^{C4}_{np} = 2$$
$$a^{C5}_{np} = 1$$

MANCHESTER 1824
The University of Manchester

Spectrum

- $\langle a^s_{ep}, a^s_{ef}, a^s_{np}, a^s_{nf} \rangle$

To count the number of times the statement *s* is executed (e) or not executed (n) on passing (p) and on failing (f) tests.

input1	input2	input3	C2	C3	C4	C5	Assert
200	100	300	**0**	**1**	**1**	**0**	**False**
300	100	200	**0**	**0**	**1**	**1**	**False**
300	300	200	**0**	**0**	**0**	**1**	**True**
100	300	200	**1**	**0**	**0**	**0**	**True**
…							

a^s_{nf} is the number of tests that failed and not executed s

$$a^{C2}_{nf} = 2$$
$$a^{C3}_{nf} = 1$$
$$a^{C4}_{nf} = 0$$
$$a^{C5}_{nf} = 1$$

MANCHESTER 1824
The University of Manchester

Measures

- Spectra

$<a^{C2}_{ep} = 1, a^{C2}_{ef} = 0, a^{C2}_{np} = 1, a^{C2}_{nf} = 2>$

$<a^{C3}_{ep} = 0, a^{C3}_{ef} = 1, a^{C3}_{np} = 2, a^{C3}_{nf} = 1>$

$<a^{C4}_{ep} = 0, a^{C4}_{ef} = 2, a^{C4}_{np} = 2, a^{C4}_{nf} = 0>$

$<a^{C5}_{ep} = 1, a^{C5}_{ef} = 1, a^{C5}_{np} = 1, a^{C5}_{nf} = 1>$

- Spectra based measures

Ochiai: $\frac{a^s_{ef}}{a^s_{ef} + a^s_{nf} + a^s_{ep} + \frac{10000 a^s_{ef} a^s_{ep}}{a^s_{ef}}}$

Zoltar: $\frac{a^s_{ef}}{\sqrt{(a^s_{ef} + a^s_{nf})(a^s_{ef} + a^s_{ep})}}$

Tarantula: $\frac{\frac{a^s_{ef}}{a^s_{ef}+a^s_{nf}}}{\frac{a^s_{ef}}{a^s_{ef}+a^s_{nf}} + \frac{a^s_{ep}}{a^s_{ep}+a^s_{np}}}$

Wong-II: $a^s_{ef} - a^s_{ep}$

MANCHESTER 1824
The University of Manchester

Ranking

Program statements	Suspicious scores (Wong-II)
C4	2
C3	1
C5	0
C2	-1

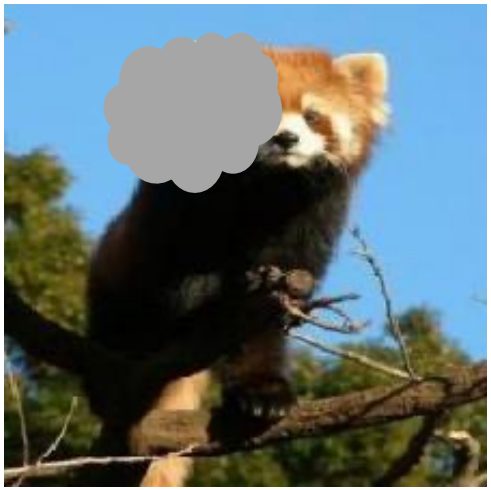
- To debug from higher ranked, more suspicious program statements
- Different measures may return different ranking
  - Ochiai: C4 (1.0), C3 (0.5), C5 (0.001), C2 (0.0)
  - No single best measure
- We only use 4 test cases …

MANCHESTER 1824
The University of Manchester

How to explain an image classifier?

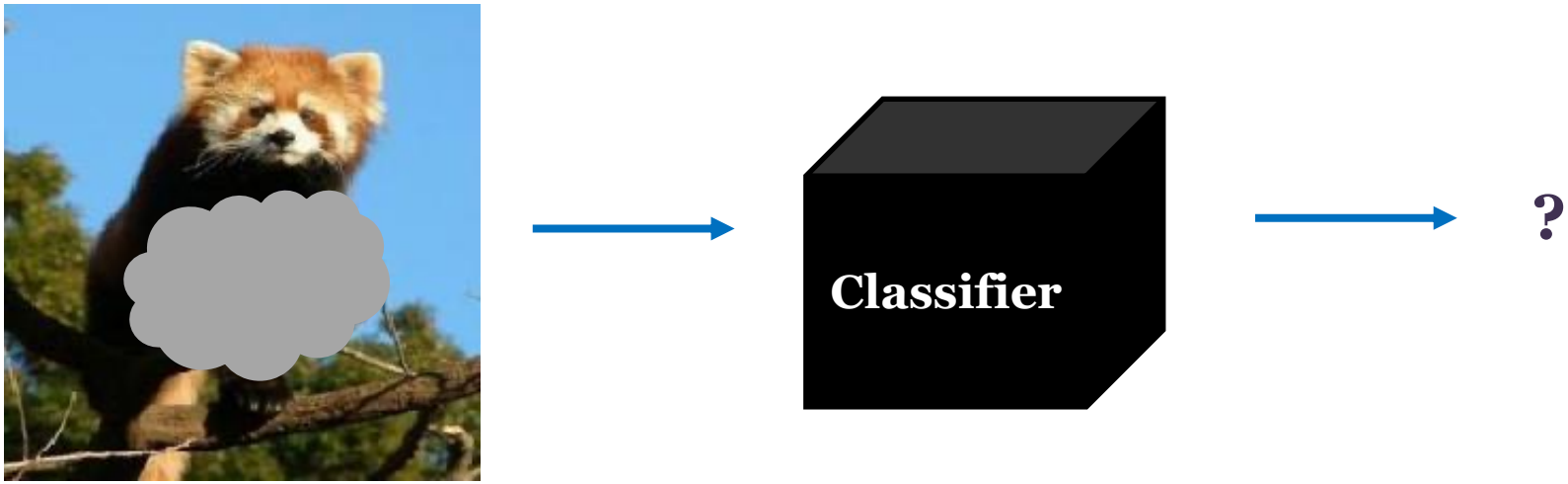
MANCHESTER 1824
The University of Manchester

How to explain an image classifier?



MANCHESTER 1824
The University of Manchester

How to explain an image classifier?



MANCHESTER 1824
The University of Manchester

How to explain an image classifier?

MANCHESTER 1824
The University of Manchester

Statistical measures for explanations

- $\langle a^s_{ep}, a^s_{ef}, a^s_{np}, a^s_{nf} \rangle$

To count the number of times the pixel s is not masked (e) or masked (n) when the classifier's decision does not change (p) and does change (f).

E.g., a^s_{ep} is the number of mutants (i.e., masked inputs) in labeled as 'red panda' in which s is not masked

- Software fault localisation measures can now be applied

MANCHESTER 1824
The University of Manchester

Explaining image classifiers

- Rank list of pixels of the input image
- Synthesize the explanation following the pixel ranking (from high to low)
    - (Definition) An explanation in image classification is a minimal subset of pixels of a given input image that is sufficient for the DNN to classify the image

original image

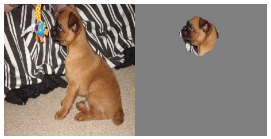
vs

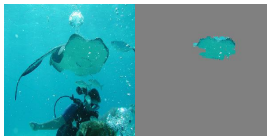
explanation

MANCHESTER 1824
The University of Manchester

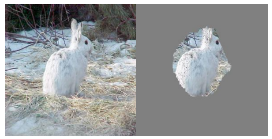
Explaining Google's Xception

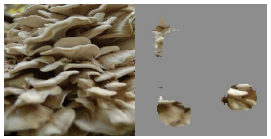

'cowboy hat'


'dog'


'numbfish'

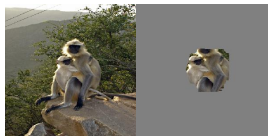

'sheep'


'hare'


'mushroom'


'wool'


'turnstile'

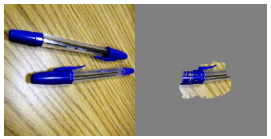

'langur'


'whistle'


'unicycle'


'fire engine'


'traffic light'

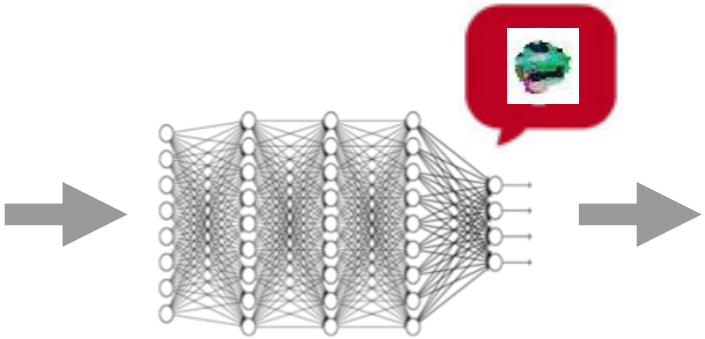

'ballpoint'


'bolo tie'

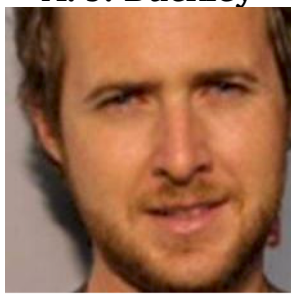

'projector'

MANCHESTER 1824
The University of Manchester

Explanation for identifying backdoor



‘A. J. Buckley’



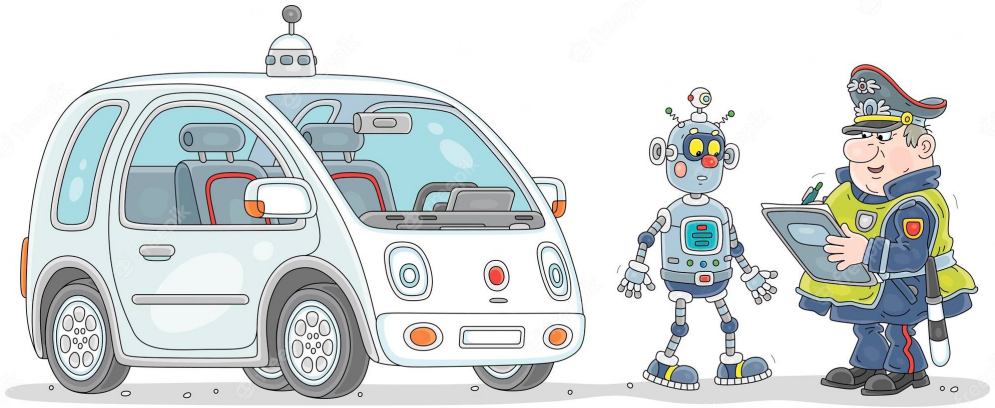
MANCHESTER 1824
The University of Manchester

References

- Sun, Youcheng, Hana Chockler, Xiaowei Huang, and Daniel Kroening. "Explaining image classifiers using statistical fault localization." In European Conference on Computer Vision (ECCV) 2020

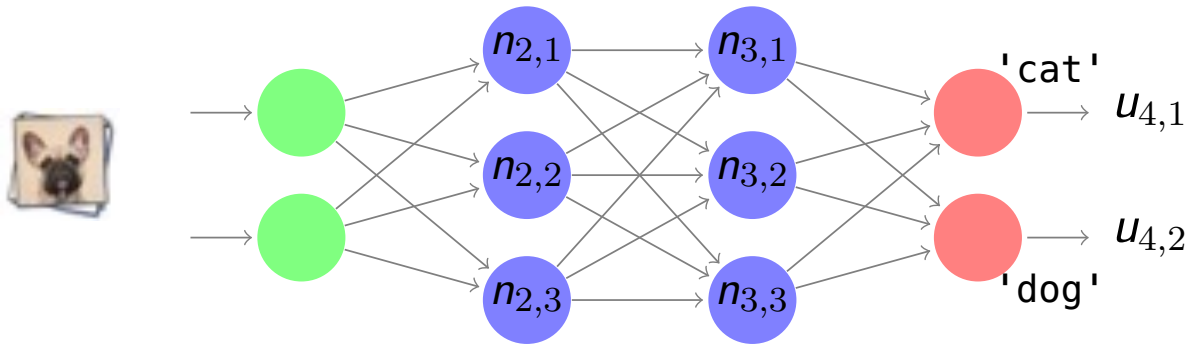
MANCHESTER 1824
The University of Manchester

Deep Neural Networks: Testing



MANCHESTER 1824
The University of Manchester

Testing DNNs



- How much testing?
  - What's the stop condition?

MANCHESTER 1824
The University of Manchester

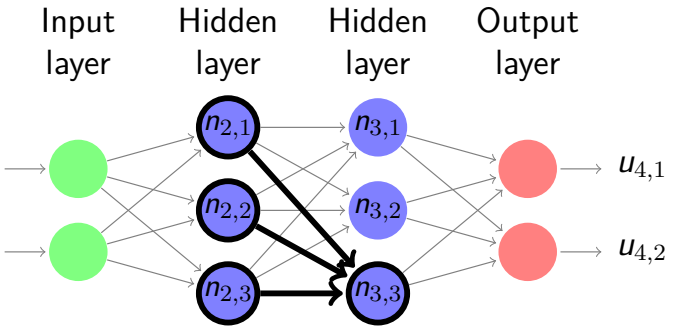
Coverage criteria

- Neuron coverage
- Neuron boundary coverage
- MC/DC for DNNs
- …

MANCHESTER 1824
The University of Manchester

Neuron coverage (NC)

For any hidden neuron $n_{k,i}$, there exists a test case $t \in \mathcal{T}$ such that the neuron $n_{k,i}$ is activated: $u_{k,i} > 0$.



$$label = \text{argmax}_{1 \le l \le s_K} u_{K,l}$$

Test coverage conditions:

$$\{\exists x. u[x]_{k,i} > 0 \mid 2 \le k \le K-1, 1 \le i \le s_k\}$$

1) neuron activation value

$$u_{k,i} = b_{k,i} + \sum_{1 \le h \le s_{k-1}} w_{k-1,h,i} \cdot v_{k-1,h}$$

weighted sum plus a bias;

w,b are parameters learned

2) rectified linear unit (ReLU):

$$v_{k,i} = \max\{u_{k,i}, 0\}$$

MANCHESTER 1824
The University of Manchester

Neuron coverage

For any hidden neuron $n_{k,i}$, there exists a test case $t \in \mathcal{T}$ such that the neuron $n_{k,i}$ is activated: $u_{k,i} > 0$.

Test coverage conditions:

$$\{\exists x.u[x]_{k,i} > 0 \mid 2 \leq k \leq K-1, 1 \leq i \leq s_k\}$$

- $\approx$ statement (line) coverage

```
...

// 1) neuron activation value
double u_{k,i} = b_{k,i};
for (unsigned h = 1; h ≤ s_{k−1}; h += 1)
{
  u_{k,i} += w_{k−1,h,i} * v_{k−1,h};
}

double v_{k,i} = 0;

// 2) ReLU
if (u_{k,i} > 0)
{
    v_{k,i} = u_{k,i};    ⇐ this line is covered
}

...
```

MANCHESTER 1824
The University of Manchester

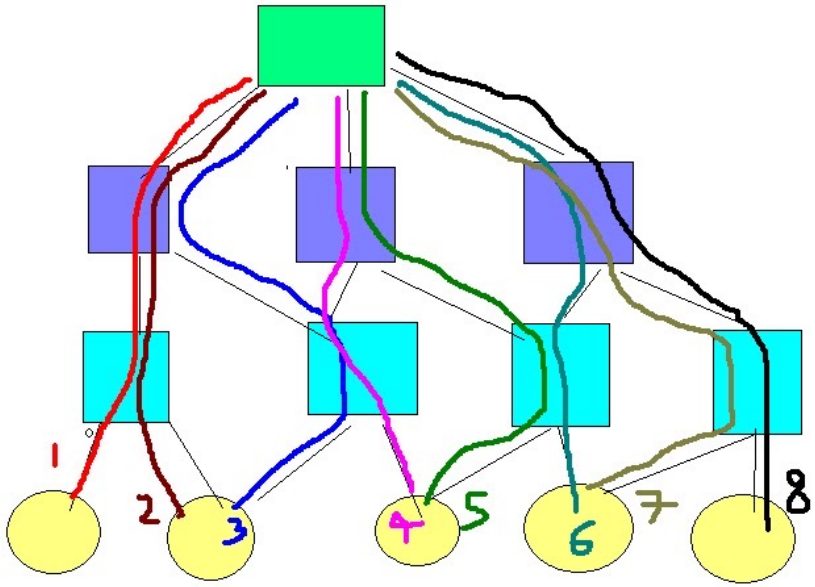
- Neuron boundary coverage

```
...

// 1) neuron activation value
double u_{k,i} = b_{k,i};
for (unsigned h = 1; h ≤ s_{k-1}; h += 1)
{
  u_{k,i} += w_{k-1,h,i} * v_{k-1,h};
}

double v_{k,i} = 0;

// 2) ReLU
if (u_{k,i} > 0)
{
  v_{k,i} = u_{k,i};
}

...
```

boundary values of $u_{k,i}$?

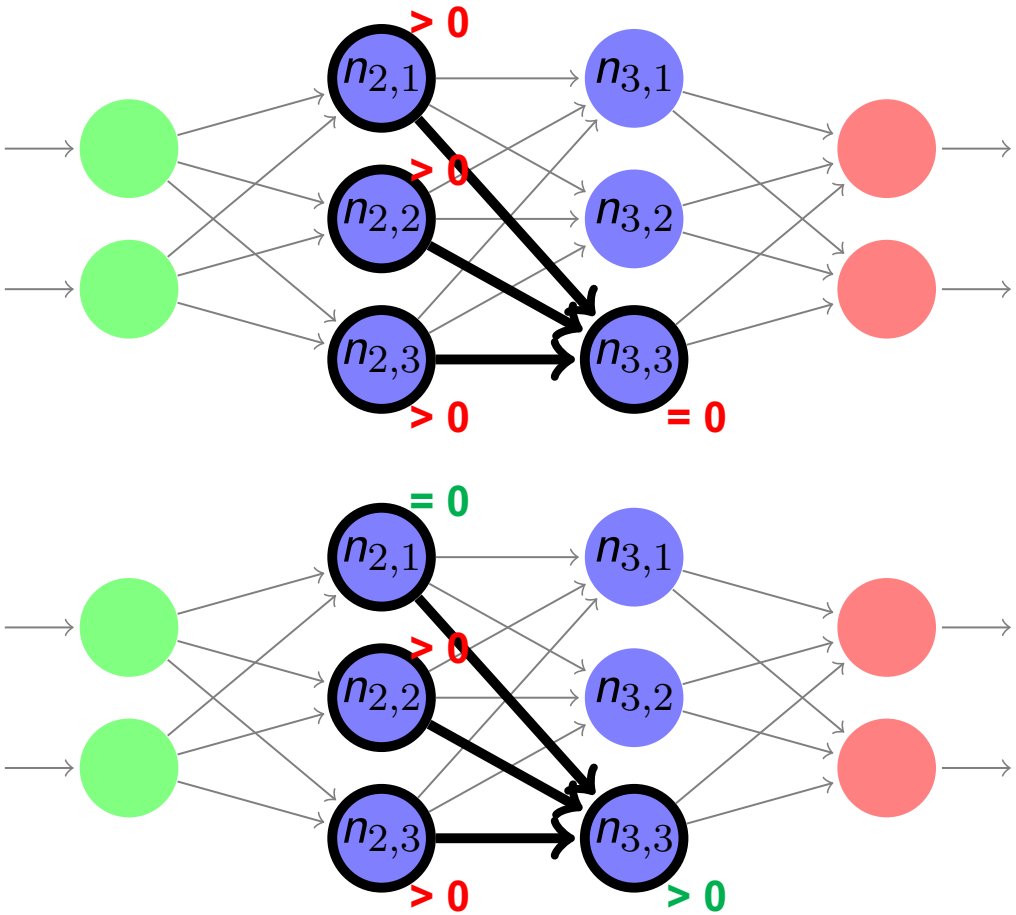
⇐ this line is covered

- All program execution paths?



MANCHESTER 1824
The University of Manchester

MC/DC for DNNs



Decision neuron: $n_{3,3}$

Condition neuron(s): $n_{2,1}$ $n_{2,2}$ $n_{2,3}$

A family of criteria

- Sign-Sign Cover (SSC)
- Value-Sign Cover (VSC)
- Sign-Value Cover (SVC)
- Value-Value Cover (VVC)

Neurons → features

MANCHESTER 1824
The University of Manchester

Measuring coverage

```
Total number of tests in test set: 10000

COVERAGE REPORT:
 10%|▮▯                                                                  | 993/10000 [00:10<01:36, 93.31it/s]
Current coverages (~1000 test images): [KMNC %, TKNC %, NBC %, SNAC %, NC %] = [24.71, 78.55, 0.07, 0.13, 56.52]
 20%|▮▮▯                                                                | 1992/10000 [00:20<01:26, 92.24it/s]
Current coverages (~2000 test images): [KMNC %, TKNC %, NBC %, SNAC %, NC %] = [34.34, 80.0, 0.2, 0.26, 57.97]
 30%|▮▮▮▯                                                               | 2996/10000 [00:29<01:03, 109.47it/s]
Current coverages (~3000 test images): [KMNC %, TKNC %, NBC %, SNAC %, NC %] = [39.91, 80.43, 0.2, 0.26, 58.7]
 40%|▮▮▮▮▯                                                              | 3990/10000 [00:39<00:53, 112.20it/s]
Current coverages (~4000 test images): [KMNC %, TKNC %, NBC %, SNAC %, NC %] = [43.5, 80.87, 0.26, 0.4, 58.7]
 50%|▮▮▮▮▮▯                                                             | 4992/10000 [00:50<00:49, 100.93it/s]
Current coverages (~5000 test images): [KMNC %, TKNC %, NBC %, SNAC %, NC %] = [46.33, 80.94, 0.36, 0.59, 58.7]
 60%|▮▮▮▮▮▮                                                             | 5993/10000 [01:00<00:36, 111.15it/s]
Current coverages (~6000 test images): [KMNC %, TKNC %, NBC %, SNAC %, NC %] = [48.57, 81.16, 0.46, 0.79, 59.42]
 70%|▮▮▮▮▮▮▮▯                                                           | 6991/10000 [01:11<00:33, 91.02it/s]
Current coverages (~7000 test images): [KMNC %, TKNC %, NBC %, SNAC %, NC %] = [50.46, 81.3, 0.49, 0.86, 59.42]
 80%|▮▮▮▮▮▮▮▮                                                           | 7995/10000 [01:22<00:22, 90.26it/s]
Current coverages (~8000 test images): [KMNC %, TKNC %, NBC %, SNAC %, NC %] = [51.9, 81.45, 0.56, 0.92, 59.42]
 90%|▮▮▮▮▮▮▮▮▮▯                                                         | 8998/10000 [01:34<00:13, 74.10it/s]
Current coverages (~9000 test images): [KMNC %, TKNC %, NBC %, SNAC %, NC %] = [53.21, 81.52, 0.59, 0.99, 59.42]
100%|▮▮▮▮▮▮▮▮▮▮                                                         | 10000/10000 [01:46<00:00, 93.88it/s]

FINAL COVERAGES:
k-Multisection Neuron Coverage (k: 1000) = 54.33%
Top-k Neuron Coverage (k: 10) = 81.59%
Neuron Boundary Coverage = 0.66%
Strong Neuron Activation Coverage = 1.05%
Neuron Coverage (threshold: 0.75) = 59.42%
```

https://github.com/DNNCov/DNNCov

MANCHESTER 1824
The University of Manchester

Tests generation

```
...

// 1) neuron activation value
u_{k,i} = b_{k,i}
for (unsigned h = 0; h ≤ s_{k-1}; h += 1)
{
   u_{k,i} += w_{k-1,h,i} · v_{k-1,h}
}


v_{k,i} = 0

// 2) ReLU
if ( u_{k,i} > 0 )
{
    v_{k,i} = u_{k,i}
}

...
```

What if not satisfied?

MANCHESTER 1824
The University of Manchester

References

- Sun, Youcheng, Min Wu, Wenjie Ruan, Xiaowei Huang, Marta Kwiatkowska, and Daniel Kroening. "Concolic testing for deep neural networks." Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering. 2018.