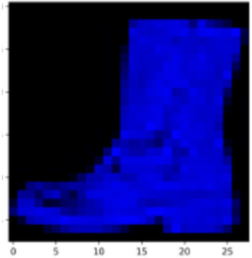
Towards Formal XAI: Formally Approximate Minimal Explanations of Neural Networks

Aritra Majumder

Chennai Mathematical Institute

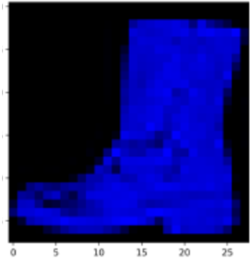
July 3, 2025

Motivation



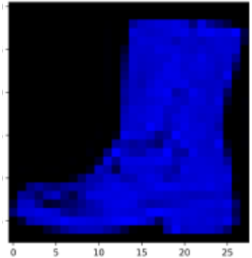
Motivation

- This picture clearly depicts a night time. If someone asks why, we will say, the sky is dark and there are moon and stars in the sky.



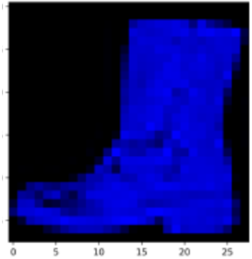
Motivation

- This picture clearly depicts a night time. If someone asks why, we will say, the sky is dark and there are moon and stars in the sky.
- So, these statements serve as an explanation for the classification instance.



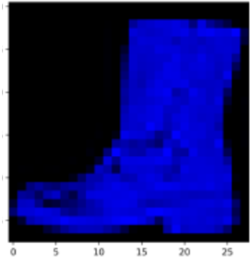
Motivation

- This picture clearly depicts a night time. If someone asks why, we will say, the sky is dark and there are moon and stars in the sky.
- So, these statements serve as an explanation for the classification instance.
- If we train a DNN to classify this instance, (if highly accurate) the DNN will classify as night, but if we ask why, we won't find an answer.



Motivation

- This picture clearly depicts a night time. If someone asks why, we will say, the sky is dark and there are moon and stars in the sky.
- So, these statements serve as an explanation for the classification instance.
- If we train a DNN to classify this instance, (if highly accurate) the DNN will classify as night, but if we ask why, we won't find an answer.
- DNNs can't provide a verbal explanation for the classification instance.



Motivation

- DNNs are now being used in numerous domains.

Motivation

- DNNs are now being used in numerous domains.
- Unfortunately, DNNs are ”black-boxes, can’t be interpreted.

Motivation

- DNNs are now being used in numerous domains.
- Unfortunately, DNNs are ”black-boxes, can’t be interpreted.
- This is a substantial concern in safety-critical systems and also prone to adversarial attacks.

Motivation

- DNNs are now being used in numerous domains.
- Unfortunately, DNNs are ”black-boxes, can’t be interpreted.
- This is a substantial concern in safety-critical systems and also prone to adversarial attacks.
- Can we say that the moon being in the sky is enough to explain the classification?

Motivation

- DNNs are now being used in numerous domains.
- Unfortunately, DNNs are ”black-boxes, can’t be interpreted.
- This is a substantial concern in safety-critical systems and also prone to adversarial attacks.
- Can we say that the moon being in the sky is enough to explain the classification?
- ”**Explainable AI**” (XAI): construct models for explaining and interpreting the decisions of DNNs.

Motivation

- DNNs are now being used in numerous domains.
- Unfortunately, DNNs are ”black-boxes, can’t be interpreted.
- This is a substantial concern in safety-critical systems and also prone to adversarial attacks.
- Can we say that the moon being in the sky is enough to explain the classification?
- ”**Explainable AI**” (XAI): construct models for explaining and interpreting the decisions of DNNs.
- Pixel values are inputs to the DNN. So, can a DNN say that the image is classified as night as a subset of pixels is white?

Motivation

- DNNs are now being used in numerous domains.
- Unfortunately, DNNs are ”black-boxes, can’t be interpreted.
- This is a substantial concern in safety-critical systems and also prone to adversarial attacks.
- Can we say that the moon being in the sky is enough to explain the classification?
- ”**Explainable AI**” (XAI): construct models for explaining and interpreting the decisions of DNNs.
- Pixel values are inputs to the DNN. So, can a DNN say that the image is classified as night as a subset of pixels is white?
- This paper depicts a similar approach, i.e., we can provide a subset of features as an explanation to justify the instance.

Motivation

- DNNs are now being used in numerous domains.
- Unfortunately, DNNs are ”black-boxes, can’t be interpreted.
- This is a substantial concern in safety-critical systems and also prone to adversarial attacks.
- Can we say that the moon being in the sky is enough to explain the classification?
- ”**Explainable AI**” (XAI): construct models for explaining and interpreting the decisions of DNNs.
- Pixel values are inputs to the DNN. So, can a DNN say that the image is classified as night as a subset of pixels is white?
- This paper depicts a similar approach, i.e., we can provide a subset of features as an explanation to justify the instance.
- How to formalize the term ”Justify”?

Brief Overview

1. Formalizing Explainability.

Brief Overview

1. Formalizing Explainability.
2. An Algorithm to compute provably correct approximate minimum explanations.

Brief Overview

1. Formalizing Explainability.
2. An Algorithm to compute provably correct approximate minimum explanations.
3. Methods to optimize the algorithm.

Brief Overview

1. Formalizing Explainability.
2. An Algorithm to compute provably correct approximate minimum explanations.
3. Methods to optimize the algorithm.
4. How to make Explanations more compact using bundles.

Brief Overview

1. Formalizing Explainability.
2. An Algorithm to compute provably correct approximate minimum explanations.
3. Methods to optimize the algorithm.
4. How to make Explanations more compact using bundles.
5. Experimental results and comparisons.

Brief Overview

1. Formalizing Explainability.
2. An Algorithm to compute provably correct approximate minimum explanations.
3. Methods to optimize the algorithm.
4. How to make Explanations more compact using bundles.
5. Experimental results and comparisons.
6. Future works.

DNN Verification

Definition

A DNN Verification query is a tuple (P, N, Q), N is a DNN that maps an input vector x to an output vector $y = N(x)$, P is a predicate on x (precondition) and Q is a predicate on y (postcondition).

DNN Verification

Definition

A DNN Verification query is a tuple (P, N, Q), N is a DNN that maps an input vector x to an output vector $y = N(x)$, P is a predicate on x (precondition) and Q is a predicate on y (postcondition).

- A DNN verifier checks whether there exists an input x_0 such that $P(x_0) \wedge Q(N(x_0))$ holds:
  - **SAT** case: Determines if there exists an input satisfying the condition (satisfiability).
  - **UNSAT** case: Verifies that no such input exists (validity).
- Just note that the DNN verification problem is known to be NP-Complete, we will use this information later.

Classification

Definition

A **classification problem** is a tuple (F, D, K, N) where

- $F = \{1, \ldots, m\}$ denotes the features;
- $D = \{D_1 \times D_2 \times \ldots \times D_m\}$ denotes the domains of each of the features;
- $K = \{c_1, c_2, \ldots, c_n\}$ is a set of classes, i.e., the possible labels;
- $\mathbb{F} = D_1 \times D_2 \times \ldots \times D_m$ is the entire input space
- $N : \mathbb{F} \to K$ is a (non-constant) classification function (in our case is a neural network).

Classification

Definition

A **classification problem** is a tuple (F, D, K, N) where

- $F = \{1, \ldots, m\}$ denotes the features;
- $D = \{D_1 \times D_2 \times \ldots \times D_m\}$ denotes the domains of each of the features;
- $K = \{c_1, c_2, \ldots, c_n\}$ is a set of classes, i.e., the possible labels;
- $\mathbb{F} = D_1 \times D_2 \times \ldots \times D_m$ is the entire input space
- $N : \mathbb{F} \rightarrow K$ is a (non-constant) classification function (in our case is a neural network).

Definition

A **classification instance** is a pair (v, c) where $v \in \mathbb{F}, c \in K$ and $c = N(v)$. In other words, v is mapped by the neural network N to class c.

Explanation: Informal Overview

- For a given classification instance (v, c), why was v mapped to c?

Explanation: Informal Overview

- For a given classification instance (v, c), why was v mapped to c?
- Is there a set of features which determines this instance? i.e., if we fix the values to those features, will such inputs always belong to class c?

Explanation: Informal Overview

- For a given classification instance (v, c), why was v mapped to c?
- Is there a set of features which determines this instance? i.e., if we fix the values to those features, will such inputs always belong to class c?
- If we set set the sky dark and draw a moon, it will always be a night time.

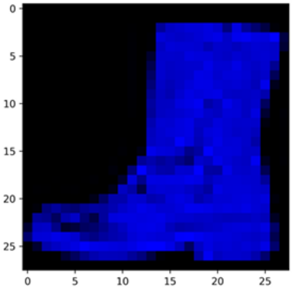
Explanation: Informal Overview

- For a given classification instance (v, c), why was v mapped to c?
- Is there a set of features which determines this instance? i.e., if we fix the values to those features, will such inputs always belong to class c?
- If we set set the sky dark and draw a moon, it will always be a night time.
- Those pixels serve as an explanation because it will always be night, whether there is a river or a tree.

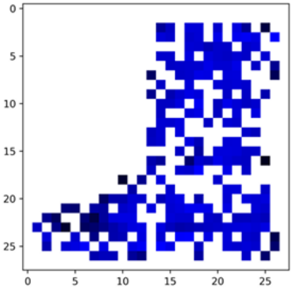
Explanation: Informal Overview

- For a given classification instance (v, c), why was v mapped to c?
- Is there a set of features which determines this instance? i.e., if we fix the values to those features, will such inputs always belong to class c?
- If we set set the sky dark and draw a moon, it will always be a night time.
- Those pixels serve as an explanation because it will always be night, whether there is a river or a tree.
- In other words, even if the values to the features that are not in the explanation are changed arbitrarily, the classification remains the same.

Example

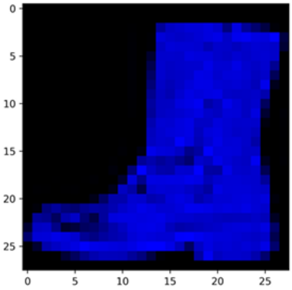


(a) Original Image

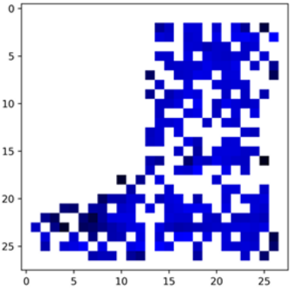


(b) Explanation

Example



(a) Original Image



(b) Explanation

- F: All the pixels, i.e., *784 pixels*.
- D_i: Assume 2 possible colors, i.e., $\{Black, Blue\}$.
- Explanation: A subset that is enough for the class to be "Shoe".
- Even if we change the remaining pixels, still a "Shoe".

Formal Definition

Definition

Given an input $v = (v_1, \ldots, v_m)$, with the classification $N(v) = c$, for some $c \in K$, an explanation (sometimes, referred to as abductive explanation or AXP) is a subset of the features $E \subseteq F$, such that

$$\forall x \in \mathbb{F}. \left[\bigwedge_{i \in E} (x_i = v_i) \implies (N(x) = c) \right]$$

Formal Definition

Definition

Given an input $v = (v_1, \ldots, v_m)$, with the classification $N(v) = c$, for some $c \in K$, an explanation (sometimes, referred to as abductive explanation or AXP) is a subset of the features $E \subseteq F$, such that

$$\forall x \in \mathbb{F}. \left[\bigwedge_{i \in E} (x_i = v_i) \implies (N(x) = c) \right]$$

- Basically, it says that $E \subseteq F$ is an explanation (for a given instance) if fixing the values to the features in E will imply that the output is fixed to the class c, whatever values the remaining features may take.

Formal Definition

Definition

Given an input $v = (v_1, \ldots, v_m)$, with the classification $N(v) = c$, for some $c \in K$, an explanation (sometimes, referred to as abductive explanation or AXP) is a subset of the features $E \subseteq F$, such that

$$\forall x \in \mathbb{F}. \left[\bigwedge_{i \in E} (x_i = v_i) \implies (N(x) = c)\right]$$

- Basically, it says that $E \subseteq F$ is an explanation (for a given instance) if fixing the values to the features in E will imply that the output is fixed to the class c, whatever values the remaining features may take.
- Note that, by definition, the set of all the features serves as a trivial explanation.

Formal Definition

Definition

Given an input $v = (v_1, \ldots, v_m)$, with the classification $N(v) = c$, for some $c \in K$, an explanation (sometimes, referred to as abductive explanation or AXP) is a subset of the features $E \subseteq F$, such that

$$\forall x \in \mathbb{F}. \left[\bigwedge_{i \in E} (x_i = v_i) \implies (N(x) = c)\right]$$

- Basically, it says that $E \subseteq F$ is an explanation (for a given instance) if fixing the values to the features in E will imply that the output is fixed to the class c, whatever values the remaining features may take.
- Note that, by definition, the set of all the features serves as a trivial explanation.
- Also note that, there might be multiple explanations for a classification instance.

Formal Definition

Definition

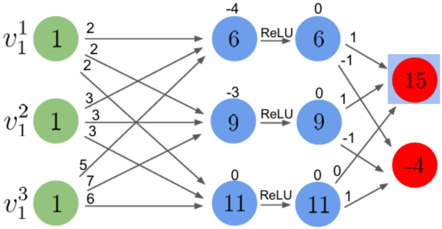
Given an input $v = (v_1, \ldots, v_m)$, with the classification $N(v) = c$, for some $c \in K$, an explanation (sometimes, referred to as abductive explanation or AXP) is a subset of the features $E \subseteq F$, such that

$$\forall x \in \mathbb{F}. \, [\bigwedge_{i \in E} (x_i = v_i) \implies (N(x) = c)]$$

- Basically, it says that $E \subseteq F$ is an explanation (for a given instance) if fixing the values to the features in E will imply that the output is fixed to the class c, whatever values the remaining features may take.
- Note that, by definition, the set of all the features serves as a trivial explanation.
- Also note that, there might be multiple explanations for a classification instance.
- We will see an example in the next slide.

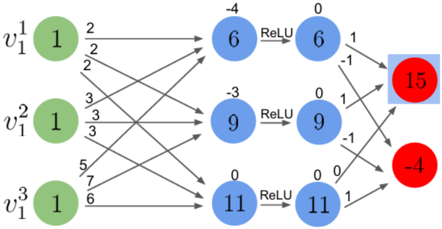
Example

Consider the following DNN with 3 input nodes and 2 output nodes:



Example

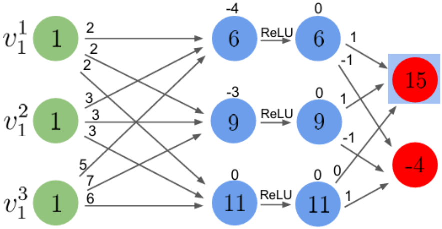
Consider the following DNN with 3 input nodes and 2 output nodes:



- Here the feature space is the set of nodes $\{v_1^1, v_1^2, v_1^3\}$ and each of the node can take values from the set $\{0, 1\}$, so the value space will be $\{0, 1\}^3$.
- Top output node classifies c_1, bottom output node classifies c_2.

Example

Consider the following DNN with 3 input nodes and 2 output nodes:



- Here the feature space is the set of nodes $\{v_1^1, v_1^2, v_1^3\}$ and each of the node can take values from the set $\{0, 1\}$, so the value space will be $\{0, 1\}^3$.
- Top output node classifies c_1, bottom output node classifies c_2.
- For input $v_1 = [1, 1, 1]^T$, the output class is predicted to be c_1 because the value 15 of the output node corresponding to c_1 is higher compared to the other -4. So, we have a classification instance (v_1, c_1) where $c_1 = N(v_1)$

Example

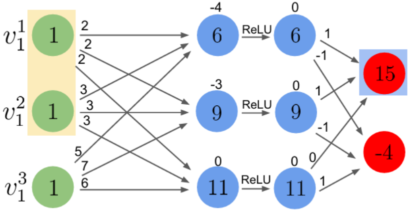
- The set $\{v_1^1, v_1^2\} \subseteq \{v_1^1, v_1^2, v_1^3\}$ is an explanation for the instance (v_1, c_1)

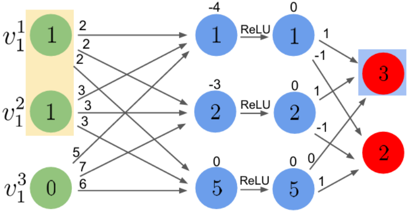
Example

- The set $\{v_1^1, v_1^2\} \subseteq \{v_1^1, v_1^2, v_1^3\}$ is an explanation for the instance (v_1, c_1)
- To verify this, let's assign all possible values to v_1^3 and see if the class changes. In this simple example, v_1^3 can only be 0 or 1.

Example

- The set $\{v_1^1, v_1^2\} \subseteq \{v_1^1, v_1^2, v_1^3\}$ is an explanation for the instance (v_1, c_1)
- To verify this, let's assign all possible values to v_1^3 and see if the class changes. In this simple example, v_1^3 can only be 0 or 1.





Verifying Explanation as an UNSAT Query

Definition

Recall that, given an input $v = (v_1, \ldots, v_m)$, with the classification $N(v) = c$, for some $c \in K$, an explanation (sometimes, referred to as abductive explanation) is a subset of the features $E \subseteq F$, such that

$$\forall x \in \mathbb{F}.\ [\bigwedge_{i \in E} (x_i = v_i) \implies (N(x) = c)]$$

Verifying Explanation as an UNSAT Query

Definition

Recall that, given an input $v = (v_1, \ldots, v_m)$, with the classification $N(v) = c$, for some $c \in K$, an explanation (sometimes, referred to as abductive explanation) is a subset of the features $E \subseteq F$, such that

$$\forall x \in \mathbb{F}.\ [\bigwedge_{i \in E} (x_i = v_i) \implies (N(x) = c)]$$

- This can be encoded as a verification query $\langle P, N, Q \rangle = \langle E = v, N, Q_{\neg c} \rangle$, if this query is UNSAT, E is an explanation.

Verifying Explanation as an UNSAT Query

Definition

Recall that, given an input $v = (v_1, \ldots, v_m)$, with the classification $N(v) = c$, for some $c \in K$, an explanation (sometimes, referred to as abductive explanation) is a subset of the features $E \subseteq F$, such that

$$\forall x \in \mathbb{F}. \left[\bigwedge_{i \in E} (x_i = v_i) \implies (N(x) = c)\right]$$

- This can be encoded as a verification query $\langle P, N, Q \rangle = \langle E = v, N, Q_{\neg c} \rangle$, if this query is UNSAT, E is an explanation.
- The example from the previous slide can be encoded as: $UNSAT\langle x_1 = 1 \wedge x_2 = 1 \wedge N(x) \neq c_1 \rangle$, where x_is a binary variable.

Minimal and Minimum Explanations

The smaller the explanation, the better the understandability.

Minimal and Minimum Explanations

The smaller the explanation, the better the understandability.

Definition

A subset $E \subseteq F$ is a minimal explanation of an instance (v, c) if it is an explanation that ceases to be valid when any single feature is removed from it:

$$(\forall x \in F)\left[\bigwedge_{i \in E}(x_i = v_i) \rightarrow (N(x) = c)\right] \wedge$$

$$(\forall j \in E)\left[\exists y \in F\left[\bigwedge_{i \in E \setminus \{j\}}(y_i = v_i) \wedge (N(y) \neq c)\right]\right]$$

Minimal and Minimum Explanations

The smaller the explanation, the better the understandability.

Definition

A subset $E \subseteq F$ is a minimal explanation of an instance (v, c) if it is an explanation that ceases to be valid when any single feature is removed from it:

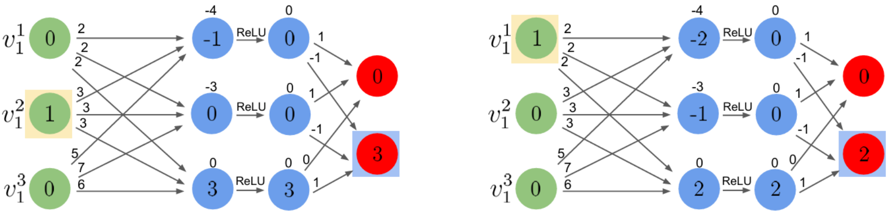
$$(\forall x \in F)\left[\bigwedge_{i \in E}(x_i = v_i) \to (N(x) = c)\right] \wedge$$

$$(\forall j \in E)\left[\exists y \in F\left[\bigwedge_{i \in E \setminus \{j\}}(y_i = v_i) \wedge (N(y) \neq c)\right]\right]$$

Definition

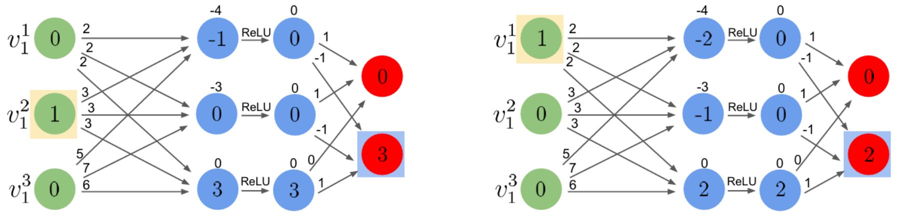
A minimum explanation is *minimal explanation* of the smallest size.

Examples

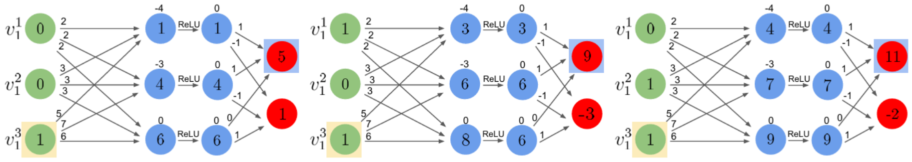


$\{v_1^1, v_1^2\}$ is a minimal explanation for input $V_1 = [1, 1, 1]^T$.

Examples

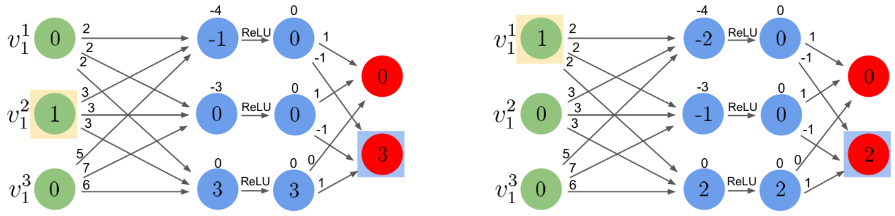


$\{v_1^1, v_1^2\}$ is a minimal explanation for input $V_1 = [1, 1, 1]^T$.

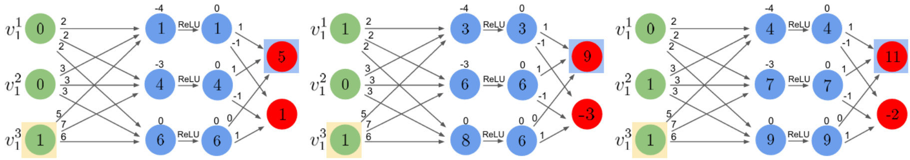


$\{v_1^3\}$ is a minimum explanation for input $V_1 = [1, 1, 1]^T$.

Examples



$\{v_1^1, v_1^2\}$ is a minimal explanation for input $V_1 = [1,1,1]^T$.



$\{v_1^3\}$ is a minimum explanation for input $V_1 = [1,1,1]^T$.

What will be a trivial algorithm to find a minimum explanation?

Minimal Explanations?

Minimal Explanations?

Algorithm 2 T_{UB}: Upper Bounding Thread

Use a heuristic model to sort F's features by ascending relevance
for each $f \in F$ **do**
    Explanation $\leftarrow$ F$\setminus$Free
    if Verify((Explanation$\setminus$\{f\})=v,N,$Q_{\neg c}$) is `UNSAT` **then**
        Free $\leftarrow$ Free $\cup$ \{f\}
        UB $\leftarrow$ UB $-$ 1
    end if
end for

Minimal Explanations?

Algorithm 2 T_{UB}: Upper Bounding Thread

```
Use a heuristic model to sort F's features by ascending relevance
for each f ∈ F do
    Explanation ← F∖Free
    if Verify((Explanation∖{f})=v,N,Q_¬c) is UNSAT then
        Free ← Free ∪ {f}
        UB ← UB − 1
    end if
end for
```

1. Gives an over-approximation of the Minimum Explanation.
2. Linear number of UNSAT queries needed in the worst case.

Minimal Explanations?

Algorithm 2 T_{UB}: Upper Bounding Thread

```
Use a heuristic model to sort F's features by ascending relevance
for each f ∈ F do
    Explanation ← F∖Free
    if Verify((Explanation∖{f})=v,N,Q_¬c) is UNSAT then
        Free ← Free ∪ {f}
        UB ← UB − 1
    end if
end for
```

1. Gives an over-approximation of the Minimum Explanation.
2. Linear number of UNSAT queries needed in the worst case.
3. UNSAT query can be prohibitively expensive as the Free set grows.

Minimal Explanations?

Algorithm 2 T_{UB}: Upper Bounding Thread

```
Use a heuristic model to sort F's features by ascending relevance
for each f ∈ F do
    Explanation ← F∖Free
    if Verify((Explanation∖{f})=v,N,Q_¬c) is UNSAT then
        Free ← Free ∪ {f}
        UB ← UB − 1
    end if
end for
```

1. Gives an over-approximation of the Minimum Explanation.
2. Linear number of UNSAT queries needed in the worst case.
3. UNSAT query can be prohibitively expensive as the Free set grows.
4. How far are we from a minimum explanation?

What's coming next?

1. Both of the naive algorithms were not efficient, linear number of queries still bad for zillion features.

What's coming next?

1. Both of the naive algorithms were not efficient, linear number of queries still bad for zillion features.

What's coming next?

1. Both of the naive algorithms were not efficient, linear number of queries still bad for zillion features.
2. Moreover, we don't even know how far we are from the minimum explanation in case of the latter algorithm.

What's coming next?

1. Both of the naive algorithms were not efficient, linear number of queries still bad for zillion features.
2. Moreover, we don't even know how far we are from the minimum explanation in case of the latter algorithm.
3. Can we get a lower bound (under-approximation) for the minimum explanation size to get an idea how far we are?

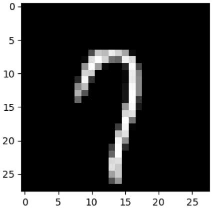
What's coming next?

1. Both of the naive algorithms were not efficient, linear number of queries still bad for zillion features.
2. Moreover, we don't even know how far we are from the minimum explanation in case of the latter algorithm.
3. Can we get a lower bound (under-approximation) for the minimum explanation size to get an idea how far we are?
4. Also, are there some features which will always be in every explanation?

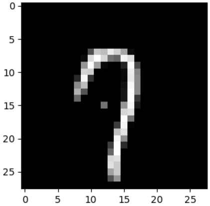
What's coming next?

1. Both of the naive algorithms were not efficient, linear number of queries still bad for zillion features.
2. Moreover, we don't even know how far we are from the minimum explanation in case of the latter algorithm.
3. Can we get a lower bound (under-approximation) for the minimum explanation size to get an idea how far we are?
4. Also, are there some features which will always be in every explanation?
5. Both the answers are Yes, thanks to the dual concept of Contrastive Example which will help us calculate the lower bound of the minimum explanation as well as speed up the algorithms we have already seen.

One Pixel Attack

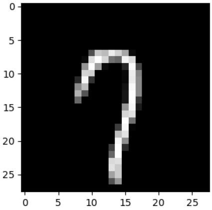


(**a**) Original input

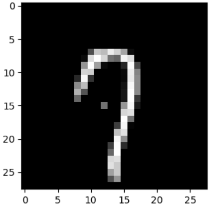


(**b**) One-pixel attacked input

One Pixel Attack



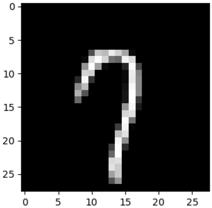
(**a**) Original input



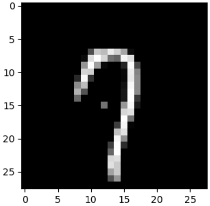
(**b**) One-pixel attacked input

- Heard of one pixel attack?

One Pixel Attack



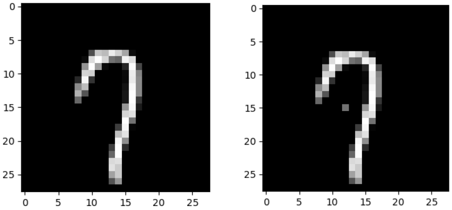
(**a**) Original input



(**b**) One-pixel attacked input

- Heard of one pixel attack?
- It's an attack such that flipping a single pixel will cause a neural network to misclassify.

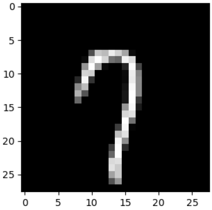
One Pixel Attack



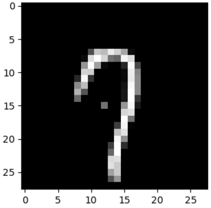
(**a**) Original input (**b**) One-pixel attacked input

- Heard of one pixel attack?
- It's an attack such that flipping a single pixel will cause a neural network to misclassify.
- In the above picture, the original image was classified as 1.

One Pixel Attack



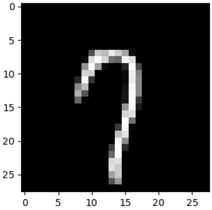
(**a**) Original input



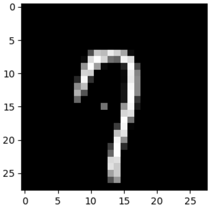
(**b**) One-pixel attacked input

- Heard of one pixel attack?
- It's an attack such that flipping a single pixel will cause a neural network to misclassify.
- In the above picture, the original image was classified as 1.
- However, after flipping a single pixel, the image was classified as 9.

One Pixel Attack



(**a**) Original input



(**b**) One-pixel attacked input

- Heard of one pixel attack?
- It's an attack such that flipping a single pixel will cause a neural network to misclassify.
- In the above picture, the original image was classified as 1.
- However, after flipping a single pixel, the image was classified as 9.
- Now, we will formally introduce Contrastive Examples.

Contrastive Examples

Definition

A subset of features $C \subseteq F$ is a contrastive example (CXP), if altering one or more of the features in C causes a misclassification of given classification instance (v, c). Formally:

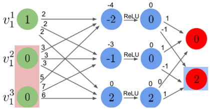
$$\exists x \in \mathbb{F}.[\wedge_{i \in F \setminus C}(x_i = v_i) \wedge (N(x) \neq c)]$$

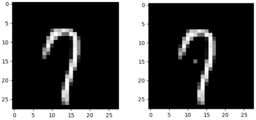
Contrastive Examples

Definition

A subset of features $C \subseteq F$ is a contrastive example (CXP), if altering one or more of the features in C causes a misclassification of given classification instance (v, c). Formally:

$$\exists x \in \mathbb{F}.[\wedge_{i \in F \setminus C}(x_i = v_i) \wedge (N(x) \neq c)]$$





(a) Original input

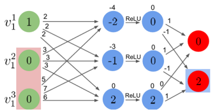
(b) One-pixel attacked input

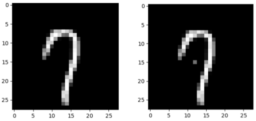
Contrastive Examples

Definition

A subset of features $C \subseteq F$ is a contrastive example (CXP), if altering one or more of the features in C causes a misclassification of given classification instance (v, c). Formally:

$$\exists x \in \mathbb{F}.[\wedge_{i \in F \setminus C}(x_i = v_i) \wedge (N(x) \neq c)]$$





(a) Original input (b) One-pixel attacked input

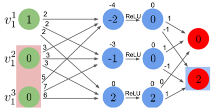
- Verify as query $\langle P, N, Q \rangle = \langle F \setminus C = v, N, Q_{\neg c} \rangle$, query is SAT $\implies$ C is a CXP.

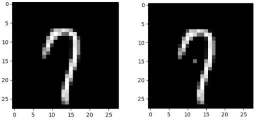
Contrastive Examples

Definition

A subset of features $C \subseteq F$ is a contrastive example (CXP), if altering one or more of the features in C causes a misclassification of given classification instance (v, c). Formally:

$$\exists x \in \mathbb{F}.[\wedge_{i \in F \setminus C}(x_i = v_i) \wedge (N(x) \neq c)]$$





(a) Original input

(b) One-pixel attacked input

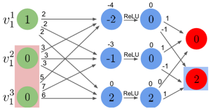
- Verify as query $\langle P, N, Q \rangle = \langle F \setminus C = v, N, Q_{\neg c} \rangle$, query is SAT $\implies$ C is a CXP.
- Note that, every superset of a CXP is also a CXP.

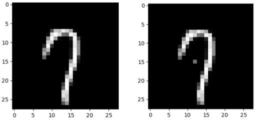
Contrastive Examples

Definition

A subset of features $C \subseteq F$ is a contrastive example (CXP), if altering one or more of the features in C causes a misclassification of given classification instance (v, c). Formally:

$$\exists x \in \mathbb{F}.[\wedge_{i \in F \setminus C}(x_i = v_i) \wedge (N(x) \neq c)]$$





(a) Original input

(b) One-pixel attacked input

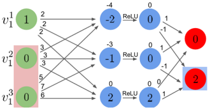
- Verify as query $\langle P, N, Q\rangle = \langle F \setminus C = v, N, Q_{\neg c}\rangle$, query is SAT $\implies$ C is a CXP.
- Note that, every superset of a CXP is also a CXP.
- Smaller the CXP, more useful it is. (in the context of the paper)

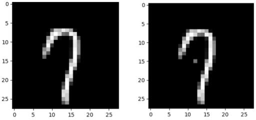
Contrastive Examples

Definition

A subset of features $C \subseteq F$ is a contrastive example (CXP), if altering one or more of the features in C causes a misclassification of given classification instance (v, c). Formally:

$$\exists x \in \mathbb{F}.[\wedge_{i \in F \setminus C}(x_i = v_i) \wedge (N(x) \neq c)]$$





(a) Original input (b) One-pixel attacked input

- Verify as query $\langle P, N, Q \rangle = \langle F \setminus C = v, N, Q_{\neg c} \rangle$, query is SAT $\implies$ C is a CXP.
- Note that, every superset of a CXP is also a CXP.
- Smaller the CXP, more useful it is. (in the context of the paper)
- The complement of an AXP is not a CXP.

Important Lemmata

Lemma

Every contrastive singleton is contained in all the explanations.

Important Lemmata

Lemma

Every contrastive singleton is contained in all the explanations.

Proof.

- Let N be a classification function, (v, c) a classification instance.

Important Lemmata

Lemma

Every contrastive singleton is contained in all the explanations.

Proof.

- Let N be a classification function, (v, c) a classification instance.
- Without loss of generality, let $E = \{1, ..., i\} \subseteq F$ be an explanation.

Important Lemmata

Lemma

Every contrastive singleton is contained in all the explanations.

Proof.

- Let N be a classification function, (v, c) a classification instance.
- Without loss of generality, let $E = \{1, ..., i\} \subseteq F$ be an explanation.
- Let $\{j\} \subseteq F$ a contrastive singleton.

Important Lemmata

Lemma

Every contrastive singleton is contained in all the explanations.

Proof.

- Let N be a classification function, (v, c) a classification instance.
- Without loss of generality, let $E = \{1, ..., i\} \subseteq F$ be an explanation.
- Let $\{j\} \subseteq F$ a contrastive singleton.
- Assume towards contradiction that j is not contained in the explanation E, i.e., $j > i$.

$\square$

Important Lemmata

Lemma

Every contrastive singleton is contained in all the explanations.

Proof.

- Let N be a classification function, (v, c) a classification instance.
- Without loss of generality, let $E = \{1, ..., i\} \subseteq F$ be an explanation.
- Let $\{j\} \subseteq F$ a contrastive singleton.
- Assume towards contradiction that j is not contained in the explanation E, i.e., $j > i$.
- With respect to v, we will find an input x which has same values for the features in explanation but a different value for the feature j and a different output class.

$\square$

Important Lemmata

Lemma

Every contrastive singleton is contained in all the explanations.

Proof.

- Let N be a classification function, (v, c) a classification instance.
- Without loss of generality, let $E = \{1, ..., i\} \subseteq F$ be an explanation.
- Let $\{j\} \subseteq F$ a contrastive singleton.
- Assume towards contradiction that j is not contained in the explanation E, i.e., $j > i$.
- With respect to v, we will find an input x which has same values for the features in explanation but a different value for the feature j and a different output class.
- Then, we will prove that such E can't be an explanation.

□

Important Lemmata

Proof.

- Let's try to find such x. We first fix the values for the all the features in $F - \{j\}$ according to v and then go through all possible values x_j that j can take.

Important Lemmata

Proof.

- Let's try to find such x. We first fix the values for the all the features in $F - \{j\}$ according to v and then go through all possible values x_j that j can take.
- As $\{j\}$ is a contrastive singleton, we will definitely find an input for which the output class is different from v's.

Important Lemmata

Proof.

- Let's try to find such x. We first fix the values for the all the features in $F - \{j\}$ according to v and then go through all possible values x_j that j can take.
- As $\{j\}$ is a contrastive singleton, we will definitely find an input for which the output class is different from v's.
- But x and v agrees on all the values for the features in the explanation and they should have the same output class, which is a contradiction.

Important Lemmata

Proof.

- Let's try to find such x. We first fix the values for the all the features in $F - \{j\}$ according to v and then go through all possible values x_j that j can take.
- As $\{j\}$ is a contrastive singleton, we will definitely find an input for which the output class is different from v's.
- But x and v agrees on all the values for the features in the explanation and they should have the same output class, which is a contradiction.

$$\exists(x_1..x_j..x_n) \in \mathbb{F} \cdot \left[\bigwedge_{i \in E} (x_i = v_i \wedge j \neq i) \wedge (N(x) \neq c) \right]$$

Important Lemmata

Proof.

- Let's try to find such x. We first fix the values for the all the features in $F - \{j\}$ according to v and then go through all possible values x_j that j can take.
- As $\{j\}$ is a contrastive singleton, we will definitely find an input for which the output class is different from v's.
- But x and v agrees on all the values for the features in the explanation and they should have the same output class, which is a contradiction.

$$\exists(x_1..x_j..x_n) \in \mathbb{F} \cdot \left[\bigwedge_{i \in E} (x_i = v_i \wedge j \neq i) \wedge (N(x) \neq c) \right]$$

- So, E can't be an explanation.

$\square$

Important Lemmata

Lemma

All explanations contain at least one element of every contrastive pair.

Important Lemmata

Lemma

All explanations contain at least one element of every contrastive pair.

Proof.

- This proof follows the same argument.

□

Important Lemmata

Lemma

All explanations contain at least one element of every contrastive pair.

Proof.

- This proof follows the same argument.
- Assume there exists a contrastive pair (C) and an explanation (E) for (v, c) that are disjoint.

$\square$

Important Lemmata

Lemma

All explanations contain at least one element of every contrastive pair.

Proof.

- This proof follows the same argument.
- Assume there exists a contrastive pair (C) and an explanation (E) for (v, c) that are disjoint.
- We fix the values of the features not in C and go through all possible value assignments to the features in C, we must get an input instance with the output class $\neq c$, let's call it x.
- But, x and v agrees on all the values for the features in E, and so, should have same output class c.

$\square$

Important Lemmata

Lemma

All explanations contain at least one element of every contrastive pair.

Proof.

- This proof follows the same argument.
- Assume there exists a contrastive pair (C) and an explanation (E) for (v, c) that are disjoint.
- We fix the values of the features not in C and go through all possible value assignments to the features in C, we must get an input instance with the output class $\neq c$, let's call it x.
- But, x and v agrees on all the values for the features in E, and so, should have same output class c.
- This is a contradiction and therefore E can't be an explanation.

□

Important Lemmata

- The above two lemmata can be generalized:

Important Lemmata

- The above two lemmata can be generalized:

Lemma

All explanations contain at least one element of all the contrastive examples of arbitrary sizes.

Important Lemmata

- The above two lemmata can be generalized:

Lemma

All explanations contain at least one element of all the contrastive examples of arbitrary sizes.

Definition

Given a collection $\mathcal{S}$ of sets from a universe U, a set $h \subseteq U$ is called a *hitting set* for $\mathcal{S}$ if

$$\forall s \in \mathcal{S}, \quad h \cap s \neq \emptyset.$$

A hitting set h is said to be:

- **Minimal** if no proper subset of h is also a hitting set.
- **Minimum** if it has the smallest possible cardinality among all hitting sets.

Important Lemmata

Lemma

A Minimal Hitting Set of all contrastive examples constitutes a minimal explanation.
The Minimum Hitting Set (MHS) of all contrastive examples is exactly the minimum explanation.

Important Lemmata

Lemma

A Minimal Hitting Set of all contrastive examples constitutes a minimal explanation.
The Minimum Hitting Set (MHS) of all contrastive examples is exactly the minimum explanation.

- We can list down all the contrastive examples.

Important Lemmata

Lemma

A Minimal Hitting Set of all contrastive examples constitutes a minimal explanation.
The Minimum Hitting Set (MHS) of all contrastive examples is exactly the minimum explanation.

- We can list down all the contrastive examples.
- Then we can pick an element from each of the contrastive example in order (skip if already hit).

Important Lemmata

Lemma

A Minimal Hitting Set of all contrastive examples constitutes a minimal explanation.
The Minimum Hitting Set (MHS) of all contrastive examples is exactly the minimum explanation.

- We can list down all the contrastive examples.
- Then we can pick an element from each of the contrastive example in order (skip if already hit).
- Converges to Minimal Hitting Set, i.e., Minimal Explanation.

Important Lemmata

Lemma

A Minimal Hitting Set of all contrastive examples constitutes a minimal explanation.
The Minimum Hitting Set (MHS) of all contrastive examples is exactly the minimum explanation.

- We can list down all the contrastive examples.
- Then we can pick an element from each of the contrastive example in order (skip if already hit).
- Converges to Minimal Hitting Set, i.e., Minimal Explanation.
- Unfortunately, MHS problem is known to be NP-Complete.

Important Lemmata

Lemma

A Minimal Hitting Set of all contrastive examples constitutes a minimal explanation.
The Minimum Hitting Set (MHS) of all contrastive examples is exactly the minimum explanation.

- We can list down all the contrastive examples.
- Then we can pick an element from each of the contrastive example in order (skip if already hit).
- Converges to Minimal Hitting Set, i.e., Minimal Explanation.
- Unfortunately, MHS problem is known to be NP-Complete.
- Further, enumerating all contrastive examples may in itself take exponential time.

Important Lemmata

Lemma

A Minimal Hitting Set of all contrastive examples constitutes a minimal explanation.
The Minimum Hitting Set (MHS) of all contrastive examples is exactly the minimum explanation.

- We can list down all the contrastive examples.
- Then we can pick an element from each of the contrastive example in order (skip if already hit).
- Converges to Minimal Hitting Set, i.e., Minimal Explanation.
- Unfortunately, MHS problem is known to be NP-Complete.
- Further, enumerating all contrastive examples may in itself take exponential time.
- Finally, DNN verification is itself an NP-Complete problem which may slow down the AXP and CXP queries.

Any Hope?

1. Though enumerating all the CXPs is expensive, we can actually clip the process of enumerating at any point to get an under-approximation of the minimum explanation.

Any Hope?

1. Though enumerating all the CXPs is expensive, we can actually clip the process of enumerating at any point to get an under-approximation of the minimum explanation.
2. Apart from that, as we are calculating the CXPs, we can use this information to further speed up the algorithm for calculating a minimal explanation.

Any Hope?

1. Though enumerating all the CXPs is expensive, we can actually clip the process of enumerating at any point to get an under-approximation of the minimum explanation.
2. Apart from that, as we are calculating the CXPs, we can use this information to further speed up the algorithm for calculating a minimal explanation.
3. Let's say, we already know that $\{f_1\}$ is a contrastive singleton. So, f_1 must be included in all the explanations.

Any Hope?

1. Though enumerating all the CXPs is expensive, we can actually clip the process of enumerating at any point to get an under-approximation of the minimum explanation.
2. Apart from that, as we are calculating the CXPs, we can use this information to further speed up the algorithm for calculating a minimal explanation.
3. Let's say, we already know that $\{f_1\}$ is a contrastive singleton. So, f_1 must be included in all the explanations.
4. This will enable us to add f_1 in the minimal explanation without making any extra verification queries.

Any Hope?

1. Though enumerating all the CXPs is expensive, we can actually clip the process of enumerating at any point to get an under-approximation of the minimum explanation.
2. Apart from that, as we are calculating the CXPs, we can use this information to further speed up the algorithm for calculating a minimal explanation.
3. Let's say, we already know that $\{f_1\}$ is a contrastive singleton. So, f_1 must be included in all the explanations.
4. This will enable us to add f_1 in the minimal explanation without making any extra verification queries.
5. Using this methods, the authors proposed a novel approach to provably approximate the Minimal Explanation we will get.

Any Hope?

1. Though enumerating all the CXPs is expensive, we can actually clip the process of enumerating at any point to get an under-approximation of the minimum explanation.
2. Apart from that, as we are calculating the CXPs, we can use this information to further speed up the algorithm for calculating a minimal explanation.
3. Let's say, we already know that $\{f_1\}$ is a contrastive singleton. So, f_1 must be included in all the explanations.
4. This will enable us to add f_1 in the minimal explanation without making any extra verification queries.
5. Using this methods, the authors proposed a novel approach to provably approximate the Minimal Explanation we will get.
6. We will first explain the naive version and then we will try to optimize the approach to speed up the algorithm.

Provable Approximations for Minimal Explanations

Algorithm 1 Minimal Explanation Search

Input N (Neural network), F (features), v (input values), c (class prediction)

1: Singletons, Pairs, Free $\leftarrow \varnothing$, UB $\leftarrow |F|$, LB $\leftarrow 0$ $\triangleright$ Global variables
2: Launch thread T_{UB}
3: Launch thread T_{LB}
4: **return** F$\setminus$Free, $\frac{\text{UB}}{\text{LB}}$

Provable Approximations for Minimal Explanations

Algorithm 1 Minimal Explanation Search

Input N (Neural network), F (features), v (input values), c (class prediction)

1: Singletons, Pairs, Free $\leftarrow \varnothing$, UB $\leftarrow |F|$, LB $\leftarrow 0$ $\triangleright$ Global variables
2: Launch thread T_{UB}
3: Launch thread T_{LB}
4: **return** F$\setminus$Free, $\frac{\text{UB}}{\text{LB}}$

- The approach runs two parallel threads: an upper bounding thread T_{UB} and a lower bounding thread T_{LB}.

Provable Approximations for Minimal Explanations

Algorithm 1 Minimal Explanation Search

Input N (Neural network), F (features), v (input values), c (class prediction)

1: Singletons, Pairs, Free $\leftarrow \varnothing$, UB $\leftarrow |F|$, LB $\leftarrow 0$ $\triangleright$ Global variables
2: Launch thread T_{UB}
3: Launch thread T_{LB}
4: **return** F$\setminus$Free, $\frac{\text{UB}}{\text{LB}}$

- The approach runs two parallel threads: an upper bounding thread T_{UB} and a lower bounding thread T_{LB}.
- T_{UB} computes a minimal explanation by reducing the feature space, providing an upper bound (over-approximation) on the minimum explanation size.

Provable Approximations for Minimal Explanations

Algorithm 1 Minimal Explanation Search

Input N (Neural network), F (features), v (input values), c (class prediction)

1: Singletons, Pairs, Free $\leftarrow \varnothing$, UB $\leftarrow |F|$, LB $\leftarrow 0$ ▷ Global variables
2: Launch thread T_{UB}
3: Launch thread T_{LB}
4: **return** F$\setminus$Free, $\frac{\text{UB}}{\text{LB}}$

- The approach runs two parallel threads: an upper bounding thread T_{UB} and a lower bounding thread T_{LB}.
- T_{UB} computes a minimal explanation by reducing the feature space, providing an upper bound (over-approximation) on the minimum explanation size.
- T_{LB} constructs contrastive sets to compute a lower bound (under-approximation) on the same, and the UB and LB together enable estimating the approximation ratio.

The Upper Bounding Thread

Algorithm 2 T_{UB}: Upper Bounding Thread

```
1: Use a heuristic model to sort F's features by ascending relevance
2: for each f ∈ F do
3:     Explanation ← F∖Free
4:     if Verify((Explanation∖{f})=v,N,Q_¬c) is UNSAT then
5:         Free ← Free ∪ {f}
6:         UB ← UB − 1
7:     end if
8: end for
```

The Lower Bounding Thread

Algorithm 3 T_{LB}: Lower Bounding Thread

for each $f \in F$ **do** ▷ Find all singletons
  if Verify((F∖{f}=v,N,$Q_{\neg c}$) is `SAT` **then**
    Singletons ← Singletons ∪ $\{f\}$
    LB ← LB +1
  end if
end for

AllPairs ← Distinct pairs of F∖Singletons
for each (a,b) ∈ AllPairs **do** ▷ Find all pairs
  if Verify((F∖{a,b}=v,N,$Q_{\neg c}$) is `SAT` **then**
    Pairs ←Pairs ∪ $\{(a, b)\}$
  end if
end for
LB ← LB + MVC(Pairs)

The Lower Bounding Thread

Algorithm 3 T_{LB}: Lower Bounding Thread

for each $f \in F$ **do** ▷ Find all singletons
    if Verify((F∖{f}=v,N,$Q_{\neg c}$) is SAT **then**
        Singletons ← Singletons ∪ $\{f\}$
        LB ← LB +1
    end if
end for

AllPairs ← Distinct pairs of F∖Singletons
for each (a,b) ∈ AllPairs **do** ▷ Find all pairs
    if Verify((F∖{a,b}=v,N,$Q_{\neg c}$) is SAT **then**
        Pairs ←Pairs ∪ $\{(a, b)\}$
    end if
end for
LB ← LB + MVC(Pairs)

- Note that, finding the MHS of all contrastive pairs is the 2-MHS problem, which is the Minimum Vertex Cover problem.

The Lower Bounding Thread

We can constrain LB by the following inequality,

$$
\begin{aligned}
\mathrm{LB} &= |\mathrm{Singletons}| + |\mathrm{MVC}(\mathrm{Pairs})| \\
&\leq \sum_{\mathrm{k=1}}^{\mathrm{k=maxk}} |\mathrm{K\text{-}MHS(k\text{-}Cxps)}| \\
&= \mathrm{MHS(Cxps)} = E_M
\end{aligned}
$$

where, E_M denotes the size of the Minimum Explanation.

The Lower Bounding Thread

We can constrain LB by the following inequality,

$$\begin{aligned} \text{LB} &= |\text{Singletons}| + |\text{MVC}(\text{Pairs})| \\ &\leq \sum_{\text{k}=1}^{\text{k}=\text{maxk}} |\text{K-MHS}(\text{k-Cxps})| \\ &= \text{MHS}(\text{Cxps}) = E_M \end{aligned}$$

where, E_M denotes the size of the Minimum Explanation.

- Unfortunately, the MVC problem is NP-Complete.
- However, the problem has a linear 2-approximating greedy algorithm, which can be used for finding a lower bound in cases of large feature spaces.

Towards the Optimization

1. The Upper Bounding Thread forms the backbone of our approach, but suffers from limited scalability.

1. The Upper Bounding Thread forms the backbone of our approach, but suffers from limited scalability.
2. As T_{UB} progresses and more features are freed, the growing search space significantly slows down the verifier.

Towards the Optimization

1. The Upper Bounding Thread forms the backbone of our approach, but suffers from limited scalability.
2. As T_{UB} progresses and more features are freed, the growing search space significantly slows down the verifier.
3. To address this, the authors have proposed three methods.

Towards the Optimization

1. The Upper Bounding Thread forms the backbone of our approach, but suffers from limited scalability.
2. As T_{UB} progresses and more features are freed, the growing search space significantly slows down the verifier.
3. To address this, the authors have proposed three methods.
4. The first method removes those that definitely belong to any explanation and hence need to be checked for inclusion in Free.

Towards the Optimization

1. The Upper Bounding Thread forms the backbone of our approach, but suffers from limited scalability.
2. As T_{UB} progresses and more features are freed, the growing search space significantly slows down the verifier.
3. To address this, the authors have proposed three methods.
4. The first method removes those that definitely belong to any explanation and hence need to be checked for inclusion in Free.
5. The second method uses binary search to discover the minimal explanation in fewer calls.

Towards the Optimization

1. The Upper Bounding Thread forms the backbone of our approach, but suffers from limited scalability.
2. As T_{UB} progresses and more features are freed, the growing search space significantly slows down the verifier.
3. To address this, the authors have proposed three methods.
4. The first method removes those that definitely belong to any explanation and hence need to be checked for inclusion in Free.
5. The second method uses binary search to discover the minimal explanation in fewer calls.
6. The third method is a heuristic to remove those features from the search space that are very likely to be in the explanation.

Method 1: Information from T_{LB}

Algorithm 4 T_{UB} using information from T_{LB}

```
Use a heuristic model to sort F by ascending relevance
RemainingFeatures ← F∖Singletons
for each f ∈ RemainingFeatures do
    Explanation ← F∖Free
    if Verify((Explanation∖{f})=v,N,Q_¬c) is UNSAT then
        Free ← Free ∪ {f}
        UB ← UB − 1
        Delete all features in a pair with f from RemainingFeatures
    end if
end for
```

Method 1: Information from T_{LB}

Algorithm 4 T_{UB} using information from T_{LB}

Use a heuristic model to sort F by ascending relevance
RemainingFeatures ← F∖Singletons
for each f ∈ RemainingFeatures **do**
  Explanation ← F∖Free
  if Verify((Explanation∖{f})=v,N,$Q_{\neg c}$) is `UNSAT` **then**
    Free ← Free ∪ {f}
    UB ← UB – 1
    Delete all features in a pair with f from RemainingFeatures
  end if
end for

1. Leverage contrastive examples found by T_{LB} to expedite T_{UB}.

Method 1: Information from T_{LB}

Algorithm 4 T_{UB} using information from T_{LB}

```
Use a heuristic model to sort F by ascending relevance
RemainingFeatures ← F∖Singletons
for each f ∈ RemainingFeatures do
    Explanation ← F∖Free
    if Verify((Explanation∖{f})=v,N,Q_¬c) is UNSAT then
        Free ← Free ∪ {f}
        UB ← UB – 1
        Delete all features in a pair with f from RemainingFeatures
    end if
end for
```

1. Leverage contrastive examples found by T_{LB} to expedite T_{UB}.
2. Contrastive Singletons must be a part of any minimal explanation.

Method 1: Information from T_{LB}

Algorithm 4 T_{UB} using information from T_{LB}

Use a heuristic model to sort F by ascending relevance
RemainingFeatures ← F∖Singletons
for each f ∈ RemainingFeatures **do**
  Explanation ← F∖Free
  if Verify((Explanation∖{f})=v,N,$Q_{\neg c}$) is `UNSAT` **then**
    Free ← Free ∪ {f}
    UB ← UB – 1
    Delete all features in a pair with f from RemainingFeatures
  end if
end for

1. Leverage contrastive examples found by T_{LB} to expedite T_{UB}.
2. Contrastive Singletons must be a part of any minimal explanation.
3. Same goes for one of the elements of every Contrastive Pair.

Method 2: Binary Search

Algorithm 6 T_{UB} using binary-search

Use a heuristic model to sort F by ascending relevance
$L = 0$
$R = |F| - 1$
while $L \leq |F| - 1$ **do**
    while $L \leq R$ **do** ▷ The inner loop is a single binary search
        Mid $\leftarrow \frac{L-R}{2}$
        Explanation $\leftarrow$F$\setminus$Free
        if Verify((Explanation$\setminus\{$L,L+1,...,Mid$\})$=v,N,$Q_{\neg c}$) is `UNSAT` **then**
            Free $\leftarrow$ Free $\cup$ $\{$L,L+1,...,Mid$\}$
            UB $\leftarrow$ UB $-|\{L, L+1, ...,\mathrm{Mid}\}|$
            $L \leftarrow$ Mid+1
        else
            $R \leftarrow$ Mid$-$1
        end if
    end while
    $L \leftarrow L + 1$
    $R \leftarrow |F| - 1$
end while

Method 2: Binary Search

Algorithm 6 T_{UB} using binary-search

```
Use a heuristic model to sort F by ascending relevance
L = 0
R = |F| − 1
while L ≤ |F| − 1 do
    while L ≤ R do                          ▷ The inner loop is a single binary search
        Mid ← (L−R)/2
        Explanation ←F∖Free
        if Verify((Explanation∖{L,L+1,...,Mid})=v,N,Q_¬c) is UNSAT then
            Free ← Free ∪ {L,L+1,...,Mid}
            UB ← UB −|{L, L+1, ...,Mid}|
            L ← Mid+1
        else
            R ← Mid−1
        end if
    end while
    L ← L + 1
    R ← |F| − 1
end while
```

- A heuristic model to sort the features based on ”importance”.

Method 2: Binary Search

Algorithm 6 T_{UB} using binary-search

```
Use a heuristic model to sort F by ascending relevance
L = 0
R = |F| − 1
while L ≤ |F| − 1 do
    while L ≤ R do                    ▷ The inner loop is a single binary search
        Mid ← (L−R)/2
        Explanation ←F∖Free
        if Verify((Explanation∖{L,L+1,...,Mid})=v,N,Q_¬c) is UNSAT then
            Free ← Free ∪ {L,L+1,...,Mid}
            UB ← UB −|{L, L+1, ...,Mid}|
            L ← Mid+1
        else
            R ← Mid−1
        end if
    end while
    L ← L + 1
    R ← |F| − 1
end while
```

- A heuristic model to sort the features based on ”importance”.
- Ideally, features in Minimum Explanation get highest weights.

Method 2: Binary Search

Algorithm 6 T_{UB} using binary-search

```
Use a heuristic model to sort F by ascending relevance
L = 0
R = |F| − 1
while L ≤ |F| − 1 do
    while L ≤ R do                    ▷ The inner loop is a single binary search
        Mid ← (L−R)/2
        Explanation ←F∖Free
        if Verify((Explanation∖{L,L+1,...,Mid})=v,N,Q_¬c) is UNSAT then
            Free ← Free ∪ {L,L+1,...,Mid}
            UB ← UB −|{L, L+1, ...,Mid}|
            L ← Mid+1
        else
            R ← Mid−1
        end if
    end while
    L ← L + 1
    R ← |F| − 1
end while
```

- A heuristic model to sort the features based on "importance".
- Ideally, features in Minimum Explanation get highest weights.
- Utilize the structure, use binary search to find the pivot.

Method 3: Local Singleton Search

1. Verification queries for explanation gets more expensive as Free grows.

Method 3: Local Singleton Search

1. Verification queries for explanation gets more expensive as Free grows.
2. Authors proposed a hypothesis that *the values that are witnesses to the violation of a feature not being in Free will be clustered close to each other*.

Method 3: Local Singleton Search

1. Verification queries for explanation gets more expensive as Free grows.
2. Authors proposed a hypothesis that *the values that are witnesses to the violation of a feature not being in Free will be clustered close to each other*.
3. Leverage this hypothesis to reduce the features to be scanned, i.e., RemainingFeatures.

Method 3: Local Singleton Search

1. Verification queries for explanation gets more expensive as Free grows.
2. Authors proposed a hypothesis that *the values that are witnesses to the violation of a feature not being in Free will be clustered close to each other*.
3. Leverage this hypothesis to reduce the features to be scanned, i.e., RemainingFeatures.
4. If a counterexample is found, fix the variables inside the explanation according to the counterexample, try to find another feature that also has violations close to the counterexample.

Method 3: Local Singleton Search

1. Verification queries for explanation gets more expensive as Free grows.
2. Authors proposed a hypothesis that *the values that are witnesses to the violation of a feature not being in Free will be clustered close to each other*.
3. Leverage this hypothesis to reduce the features to be scanned, i.e., RemainingFeatures.
4. If a counterexample is found, fix the variables inside the explanation according to the counterexample, try to find another feature that also has violations close to the counterexample.
5. If there is such a feature then it's likely a contrastive singleton based on the hypothesis and can be exempted from checking.

Method 3: Local Singleton Search

1. Verification queries for explanation gets more expensive as Free grows.
2. Authors proposed a hypothesis that *the values that are witnesses to the violation of a feature not being in Free will be clustered close to each other*.
3. Leverage this hypothesis to reduce the features to be scanned, i.e., RemainingFeatures.
4. If a counterexample is found, fix the variables inside the explanation according to the counterexample, try to find another feature that also has violations close to the counterexample.
5. If there is such a feature then it's likely a contrastive singleton based on the hypothesis and can be exempted from checking.
6. Trade-off between the size of the explanation and the time complexity.

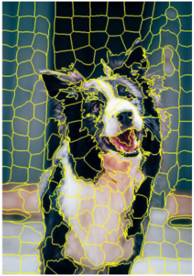
Algorithm for method 3: Local-Singleton Search

Algorithm 5 T_{UB} using local-singleton search

Use a heuristic model to sort F by ascending relevance
RemainingFeatures $\leftarrow$ F$\setminus$Singletons
for each f $\in$ RemainingFeatures **do**
    Explanation $\leftarrow$ F$\setminus$Free
    if Verify((Explanation$\setminus$\{f\})=v,N,$Q_{\neg c}$) is `UNSAT` **then**
        Free $\leftarrow$ Free $\cup$ \{f\}
        UB $\leftarrow$ UB $-$ 1
    else
        Extract counter example C
        LocalSingletons $\leftarrow \varnothing$
        for each f' $\in$ RemainingFeatures **do**
            if Verify(Explanation$\setminus\{f'\}$ = C,N,$Q_{\neg c}$) is `SAT` **then**
                LocalSingletons $\leftarrow$ LocalSingletons $\cup$ $\{f'\}$
            end if
        end for
        RemainingFeatures $\leftarrow$ RemainingFeatures $\setminus$ LocalSingletons
    end if
end for

Bundles

- Mainly to tackle the challenge ”low-level” explanations for certain users.
- Intuitively, bundles are a partitioning of the features into disjoint sets.
- Explanations in terms of bundles are often easier to comprehend.
- Also curtails the search space traversed by the verifier hence speeds up the process further.



Definition

A **bundle** is a subset $b \subseteq F$. The set of all bundles $B = \{b_1, \ldots, b_n\}$ forms a partitioning of F, i.e., $F = \coprod b_i$.

Bundle Explanations

Definition

A ***bundle explanation*** E_B for a classification instance (v, c) is a subset of bundles, $E_B \subseteq B$, such that

$$\forall x \in \mathbb{F}.[\wedge_{i \in \cup E_B}(x_i = v_i) \implies (N(x) = c)]$$

Bundle Explanations

Definition

A ***bundle explanation*** E_B for a classification instance (v, c) is a subset of bundles, $E_B \subseteq B$, such that

$$\forall x \in \mathbb{F}.[\wedge_{i \in \cup E_B}(x_i = v_i) \implies (N(x) = c)]$$

Theorem

The union of features in a bundle explanation is an explanation.

Bundle Explanations

Definition

A ***bundle explanation*** E_B for a classification instance (v, c) is a subset of bundles, $E_B \subseteq B$, such that

$$\forall x \in \mathbb{F}.[\wedge_{i \in \cup E_B}(x_i = v_i) \implies (N(x) = c)]$$

Theorem

The union of features in a bundle explanation is an explanation.

- Can develop similar notions like minimum bundle explanations, contrastive bundle examples, etc.
- Algorithmic methods can be extended to the case of bundles to perhaps get more comprehensible explanations.

Experimental Setup

- They implemented a Python framework using the Marabou DNN verifier, supporting proofs, abstraction, and optimization.

Experimental Setup

- They implemented a Python framework using the Marabou DNN verifier, supporting proofs, abstraction, and optimization.
- Feature relevance was computed using LIME; the MVC used the 2-approximate greedy algorithm.

Experimental Setup

- They implemented a Python framework using the Marabou DNN verifier, supporting proofs, abstraction, and optimization.
- Feature relevance was computed using LIME; the MVC used the 2-approximate greedy algorithm.
- Experiments ran on x86-64 GNU/Linux with one Intel Xeon Gold 6130 core and 1-hour timeout.

Experimental Setup

- They implemented a Python framework using the Marabou DNN verifier, supporting proofs, abstraction, and optimization.
- Feature relevance was computed using LIME; the MVC used the 2-approximate greedy algorithm.
- Experiments ran on x86-64 GNU/Linux with one Intel Xeon Gold 6130 core and 1-hour timeout.
- Benchmarks included DNNs trained on MNIST (96.6% accuracy) and Fashion-MNIST (87.6% accuracy) with a 784-30-10-10 softmax architecture.

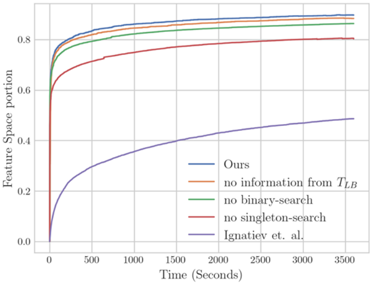
Experimental Setup

- They implemented a Python framework using the Marabou DNN verifier, supporting proofs, abstraction, and optimization.
- Feature relevance was computed using LIME; the MVC used the 2-approximate greedy algorithm.
- Experiments ran on x86-64 GNU/Linux with one Intel Xeon Gold 6130 core and 1-hour timeout.
- Benchmarks included DNNs trained on MNIST (96.6% accuracy) and Fashion-MNIST (87.6% accuracy) with a 784-30-10-10 softmax architecture.
- They selected classification instances where the network had low confidence — i.e., the top two output scores were close.

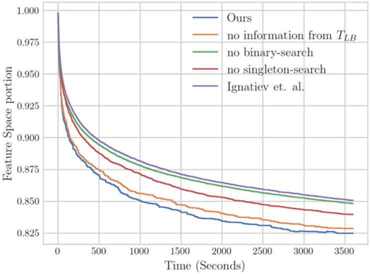
Experimental Setup

- They implemented a Python framework using the Marabou DNN verifier, supporting proofs, abstraction, and optimization.
- Feature relevance was computed using LIME; the MVC used the 2-approximate greedy algorithm.
- Experiments ran on x86-64 GNU/Linux with one Intel Xeon Gold 6130 core and 1-hour timeout.
- Benchmarks included DNNs trained on MNIST (96.6% accuracy) and Fashion-MNIST (87.6% accuracy) with a 784-30-10-10 softmax architecture.
- They selected classification instances where the network had low confidence — i.e., the top two output scores were close.
- Such low-confidence inputs yield more meaningful and larger explanations, aiding thorough experimentation.

Experimental results



(a) Average portion of features verified to participate in the explanation.



(b) Average explanation size.

Fig. 7: Our full and ablation-based results, compared to the state of the art for finding minimal explanations on the MNIST dataset.

Experimental results

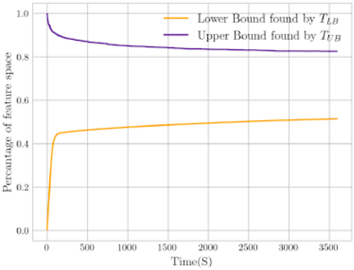


Fig. 8: Average approximation of *minimum* explanation over time.

- The average approximation ratio was 1.6 for MNIST and 1.19 for Fashion-MNIST.

Experimental results

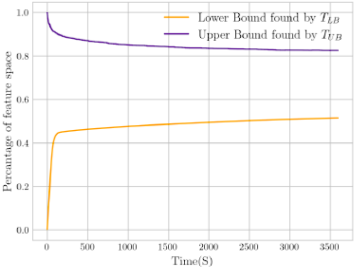
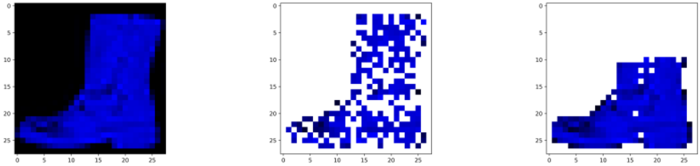


Fig. 8: Average approximation of *minimum* explanation over time.

- The average approximation ratio was 1.6 for MNIST and 1.19 for Fashion-MNIST.



(a) Original Image (b) Explanation (c) Bundle explanation

Future Work

1. Currently, the algorithm finds an explanation for each input points, but can we do the same for a class of inputs, i.e., given a class of inputs what features will always be in each explanation for each input?

Future Work

1. Currently, the algorithm finds an explanation for each input points, but can we do the same for a class of inputs, i.e., given a class of inputs what features will always be in each explanation for each input?
2. Similar problem can be that given a set of features and their assignments, which class of inputs will definitely be explained?

Future Work

1. Currently, the algorithm finds an explanation for each input points, but can we do the same for a class of inputs, i.e., given a class of inputs what features will always be in each explanation for each input?
2. Similar problem can be that given a set of features and their assignments, which class of inputs will definitely be explained?
3. Can we get some approximate algorithms that can find the explanation up to size k of the minimum explanation in linear (or polynomial) time?

Future Work

1. Currently, the algorithm finds an explanation for each input points, but can we do the same for a class of inputs, i.e., given a class of inputs what features will always be in each explanation for each input?
2. Similar problem can be that given a set of features and their assignments, which class of inputs will definitely be explained?
3. Can we get some approximate algorithms that can find the explanation up to size k of the minimum explanation in linear (or polynomial) time?
4. The authors used LIME to sort the input features by relevance. Yu et. al. also proposed an algorithm to approximate FFA (Formal Feature Attribution) in an anytime fashion. We can use that algorithm instead of LIME.

Thank You