

Finding Input Data Domains of Image Classification Models with Hard-Label Black-Box Access



Ji Yi Zhang



Han Fang



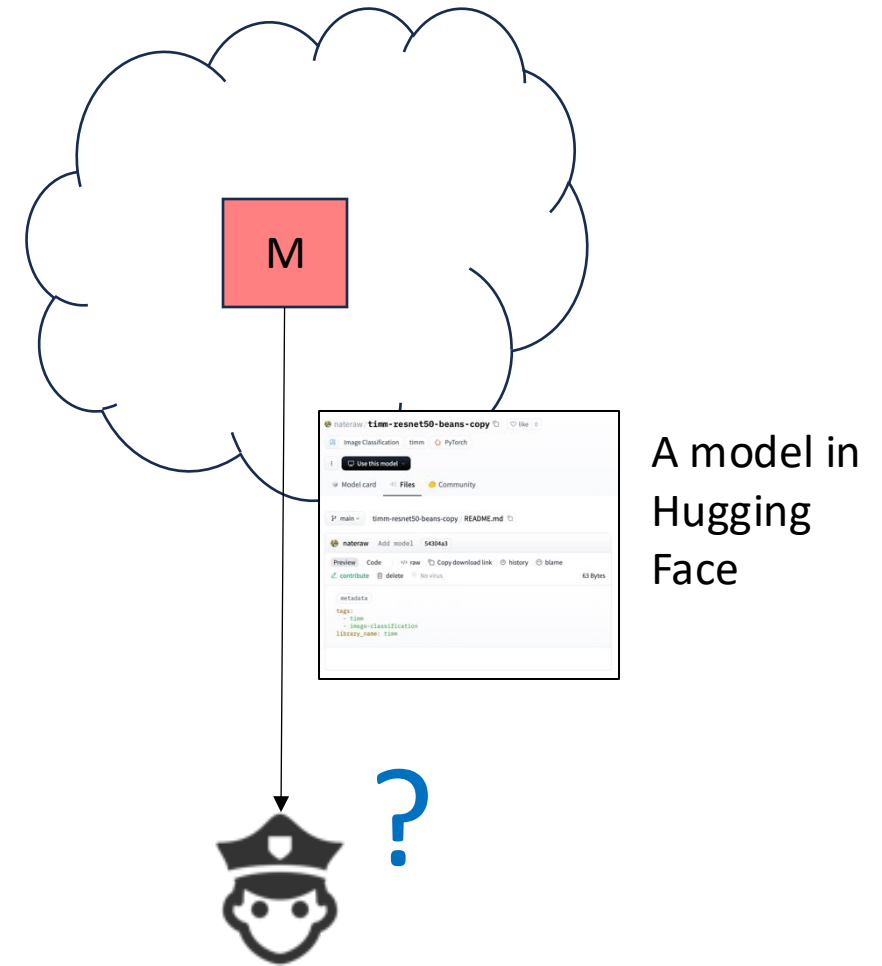
Ee-Chien Chang

School of Computing
National University of Singapore

Background

Forensic of ML models

- There are many datasets and platforms to assist building of ML models. A developer might build models with malicious intents.
- We take the role of an investigator. The investigator has a targeted model ***M***, which could be in seized devices, remote services over Internet, or a poorly documented model in a public repository such as Hugging Face.
- The investigator wants to know more about ***M***.



Useful Techniques for investigation

Model Inversion.

Given a model M and prediction y , find an instance x , s.t.

$$M(x) = y$$

Class Inversion.

Given a classification model M and a class c , find representative instances x_1, \dots, x_n , s.t.

$$M(x_i) = c \quad \text{for each } i$$

Membership inference.

Given a model M and an instance x , determine whether x is in the training set that trains M .

Cloning.

Given blackbox access to a model M , clone another M' .

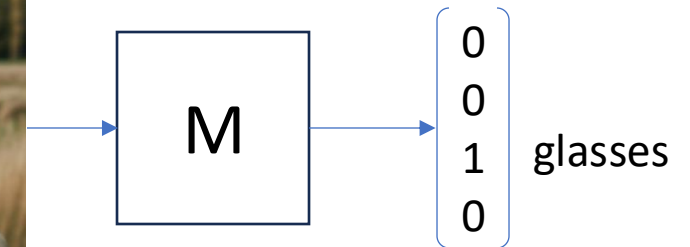
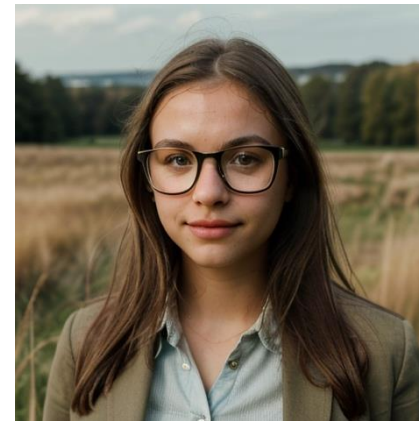
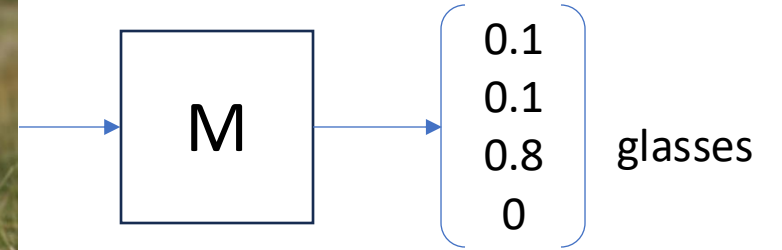
- These techniques require knowledge of the domain to start from. E.g.
 - *knowing that the input are face images;*
 - *samples of the training data.*
- **Our Goal:** We want to fill in this gap, i.e. from blackbox accesses of a model M , find its domain.

Challenges: The problem is “ill-posed”

- An ML model always gives a prediction, even if the input is meaningless.

E.g. a model that is trained on human faces to classify whether the subject is wearing glasses, could give the same output on images of cats. We would expect the investigation outcome is a set of human faces, instead of cat faces.

- How to represent a “domain”?



Proposed approaches

Main idea

Problem: Given blackbox accesses to a model M , determine a D that approximates the training data distribution.

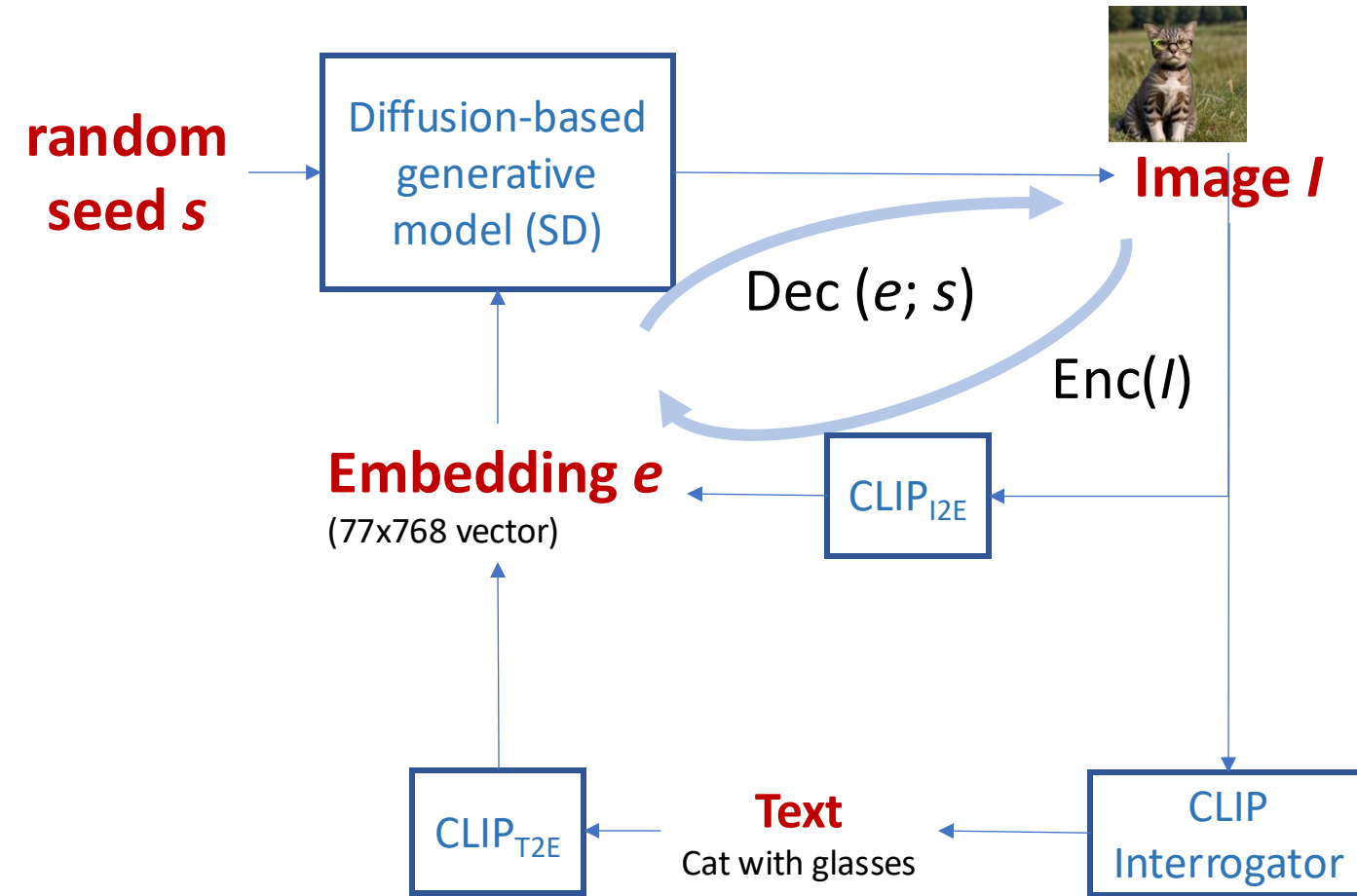
Approach:

- Leverage on pre-trained models (in particular, Image generative model + Visual language model) and/or images corpus.
- Constrain the solution space by the followings:

We seek a distribution D which is

- General: The spread of the distribution D is wide.
- Functionally Relevant: The model M classifies correctly on samples from D .
- Semantically correct: Samples from D meet common sense.

Background on SD (Stable Diffusion) + CLIP (Contrastive Language-Image Pre-training)



- An embedding e with random seeds induces an image distribution.
- A set of embeddings $E = \{e_1, e_2, \dots, e_k\}$ induces a mixture distribution D of D_1, D_2, \dots, D_k .

Method 1 (Generative)

Problem formulation:

Given the classification model M and a class. Determine a set of embeddings $E = \{e_1, e_2, \dots, e_k\}$ that maximize the weighted sum $\lambda (A) + (B)$.

- **Generality of E.**

The spread of the distribution is wide.

$$-\mathbb{E}_s [\cos(\text{Enc}(\text{Dec}(e; s)), e)] \quad \text{-- (A)}$$

- **Functional Relevance of E.**

Probability that a sample is being classified correctly.

$$\Pr_s [\arg \max \mathcal{M}(\text{Dec}(e; s)) = i] \quad \text{-- (B)}$$

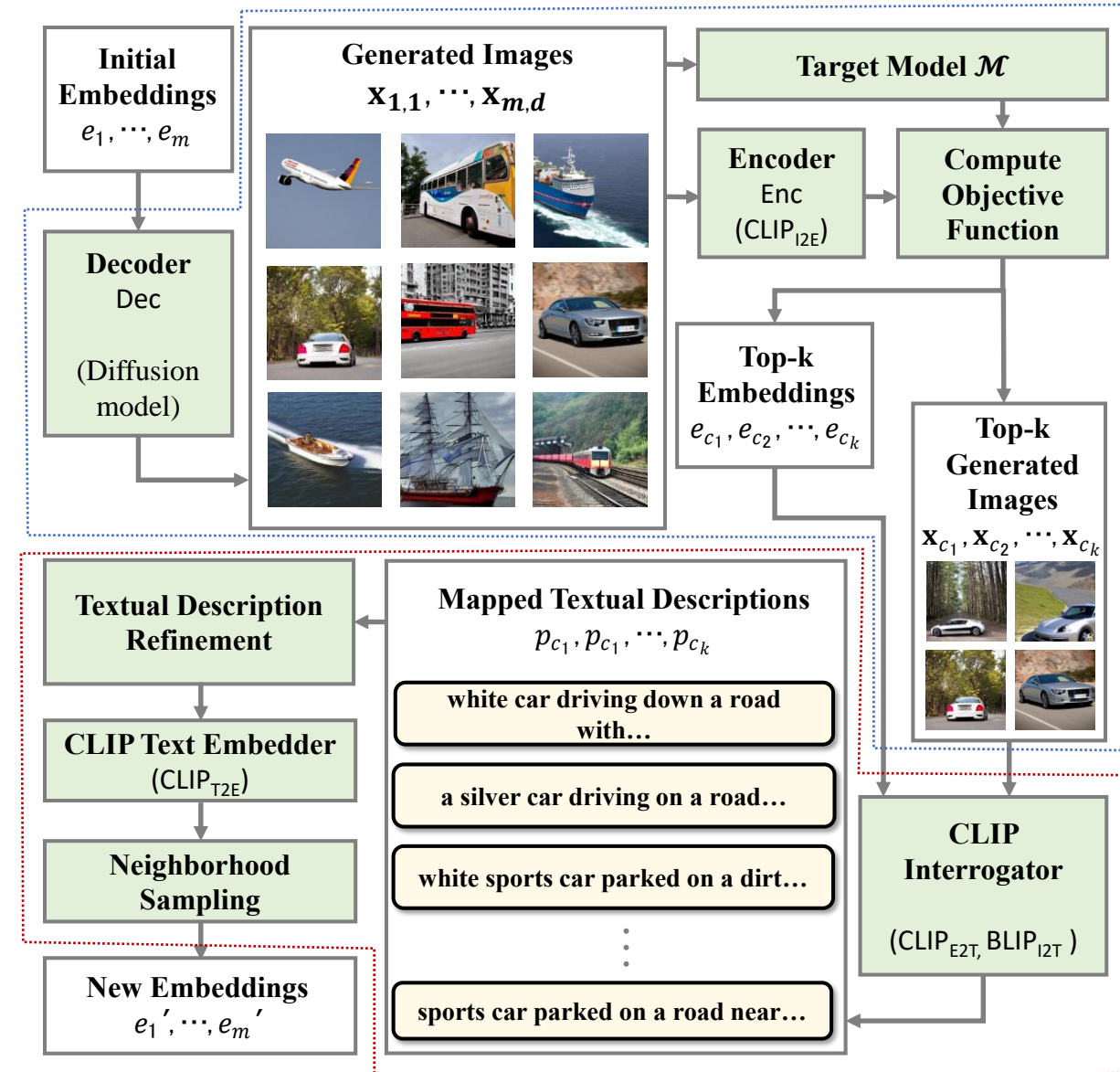
- **Semantically relevance.**

Enforced by employing CLIP to generate the embedding, and with small fixed k .

k

Search Algorithm

- We design an iterative search for the objective function.
- Each iteration starts with a larger set of embeddings.
 - Selects the top k based on the objective function;
 - Expands the top k by manipulation in the text-space;
 - Repeats.



Method 2: Using a Hierarchical Corpus (ImageNet)

Problem formulation:

- Given a model M and a class c , find a subset $D = \{x_1, x_2, \dots, x_k\}$ of the Corpus that maximizes the weighted sum $\alpha (B) + (1 - \alpha) (C)$.

Generality of D.

Enforced by having a large fixed k .

Functional Relevance of D.

Probability that a randomly chosen and transformed image from D is correctly classified.

Semantically relevance of D:

Variance of $CL(x, y)$ on two randomly chosen images from D , where $CL(x, y)$ is the length of the shortest path from the x 's class to y 's class in the hierarchy tree.

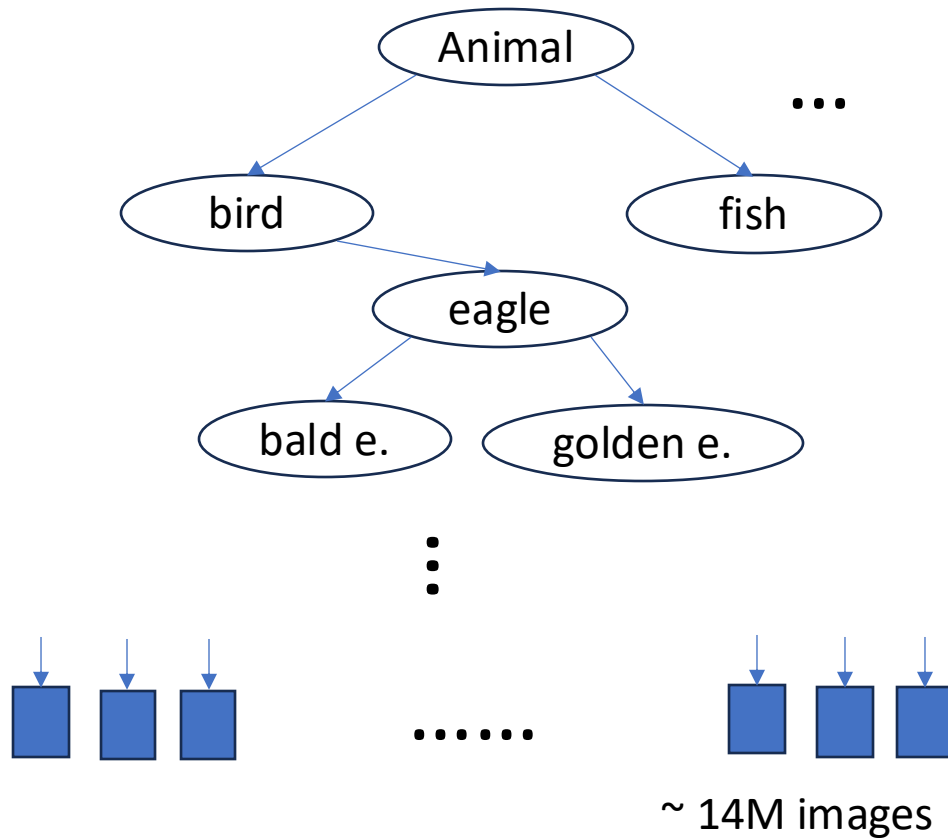
k

*Random Image
transformation such as
rotation*

$$\Pr_{\mathbf{x} \sim D} [\arg \max (M(\mathcal{E}(\mathbf{x}; s))) = c] \quad \text{-- (B)}$$

$$\text{Var}_{\mathbf{x}, \mathbf{y} \sim D} [CL(\text{class}(\mathbf{x}), \text{class}(\mathbf{y}))] \quad \text{-- (C)}$$

Hierarchical Image Corpus (ImageNet)



Images grouped according to the WordNet hierarchy.

Method 1 (Generative) vs Method 2 (Corpus)

- Output type:
 - Method 1 gives a generative model. Output is a set of embeddings whereby arbitrary instances can be sampled from the distribution.
 - Method 2 gives a set of instances/images sampled from the distribution.
- Computing resources:
 - Method 1 (150 mins on A100)
 - Method 2 (~39 seconds on A100)
- Effectiveness:
 - Method 1 outperforms method 2 (see next slide).

We employ Method 2 as a pre-processing to find a starting point for Method 1. Improve the running time to 50 mins while maintaining similar accuracy.

Evaluation

Baseline:
Clone using
the training set

Clone using
other datasets

Method 2

Method 1

Re-train with
Method 1's
output

- Methodology: via a downstream application.
 - Apply proposed method to find the domain of a given classifier **M**.
 - Clone **M** to get **M'** using the found domain.
 - Classification performance of the cloned **M'** indicates accuracy of the domain found.
- In this experiment, **M** classifies the 10 classes in CIFAR 10.

CIFAR 10 class names	Orig. Acc. (a)	Model Cloning							Train with gen data (i)
		CIFAR 10 (b)	Cal tech 101 (c)	Ox. 102 (d)	Image Net (e)	Pre proc Func (f)	Pre proc Full (g)	Final output (h)	
airplane	95.3%	84.5%	73.7%	72.7%	35.8%	78.9%	76.6%	87.3%	97.5%
auto mobile	97.8%	90.7%	0.1%	0.0%	44.6%	94.1%	92.9%	87.9%	97.8%
bird	92.8%	68.1%	17.2%	24.7%	15.2%	39.1%	43.3%	78.1%	93.1%
cat	88.6%	74.7%	77.1%	68.1%	41.3%	71.2%	74.4%	84.9%	92.8%
deer	96.6%	80.8%	72.4%	49.7%	46.5%	61.2%	70.3%	82.5%	92.8%
dog	91.9%	69.2%	3.4%	4.0%	54.0%	61.4%	77.7%	90.9%	96.1%
frog	97.4%	86.2%	72.7%	65.4%	20.1%	64.4%	78.5%	82.5%	95.3%
horse	96.5%	78.5%	0.3%	0.0%	31.1%	53.9%	57.8%	88.2%	96.4%
ship	96.0%	90.7%	83.8%	59.6%	43.2%	67.4%	68.9%	90.4%	97.2%
truck	96.4%	89.8%	54.3%	0.0%	23.2%	38.7%	79.8%	85.3%	97.0%
Average	94.9%	81.3%	45.5%	34.4%	35.5%	63.0%	72.0%	85.8%	95.6%

Evaluation

Baseline:
Clone using
the training set

Clone using
other datasets

Method 2

Method 1




Re-train with
Method 1's
output




- Methodology: via a downstream application.
 - Apply proposed method to find the domain of a given classifier **M**.
 - Clone **M** to get **M'** using the found domain.
 - Classification performance of the cloned **M'** indicates accuracy of the domain found.
- In this experiment, **M** classifies the 10 classes in CIFAR 10.


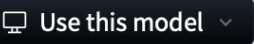

CIFAR 10 class names	Orig. Acc. (a)	Model Cloning							Train with gen data (i)
		CIFAR 10 (b)	Cal tech 101 (c)	Ox. 102 (d)	Image Net (e)	Pre proc Func (f)	Pre proc Full (g)	Final output (h)	
airplane	95.3%	84.5%	73.7%	72.7%	35.8%	78.9%	76.6%	87.3%	97.5%
auto mobile	97.8%	90.7%	0.1%	0.0%	44.6%	94.1%	92.9%	87.9%	97.8%
bird	92.8%	68.1%	17.2%	24.7%	15.2%	39.1%	43.3%	78.1%	93.1%
cat	88.6%	74.7%	77.1%	68.1%	41.3%	71.2%	74.4%	84.9%	92.8%
deer	96.6%	80.8%	72.4%	49.7%	46.5%	61.2%	70.3%	82.5%	92.8%
dog	91.9%	69.2%	3.4%	4.0%	54.0%	61.4%	77.7%	90.9%	96.1%
frog	97.4%	86.2%	72.7%	65.4%	20.1%	64.4%	78.5%	82.5%	95.3%
horse	96.5%	78.5%	0.3%	0.0%	31.1%	53.9%	57.8%	88.2%	96.4%
ship	96.0%	90.7%	83.8%	59.6%	43.2%	67.4%	68.9%	90.4%	97.2%
truck	96.4%	89.8%	54.3%	0.0%	23.2%	38.7%	79.8%	85.3%	97.0%
Average	94.9%	81.3%	45.5%	34.4%	35.5%	63.0%	72.0%	85.8%	95.6%




Example




A model from **Huggingface** without detailed documentation.
(The name “beans” revealed some info. Such info was not fed into our algorithm)


 nateraw / **timm-resnet50-beans-copy**   like 0

 Image Classification  timm  PyTorch

  Use this model 


 Model card  **Files**  Community


 main  timm-resnet50-beans-copy / README.md 


 nateraw Add model 54304a3


Preview




Code

 raw

 Copy download link

 history

 blame

 contribute  delete  No virus 63 Bytes

metadata

```
tags:
- timm
- image-classification
library_name: timm
```

input produced by proposed method.



We later found out that the author uploaded two copies:
(1) timm-resnet50-bean-copy
(2) timm-resnet50-bean.
The later includes an input example.



Summary & Conclusion

- Forensic and attribution of poorly documented models is needed.
- Many useful investigation tools require and assume that there is a domain to start from. Our goal is to fill the gap by finding a relevant domain.
- Finding the domain is an ill-posed problem. We constrain it by introducing three criteria: *Generality, Functional Relevance & Semantic Relevance*.
- We propose two methods. Empirical studies demonstrated effectiveness of the proposed methods.