# Test Adequacy & Input Generation for DNNs

## CS454 Automated Software Testing

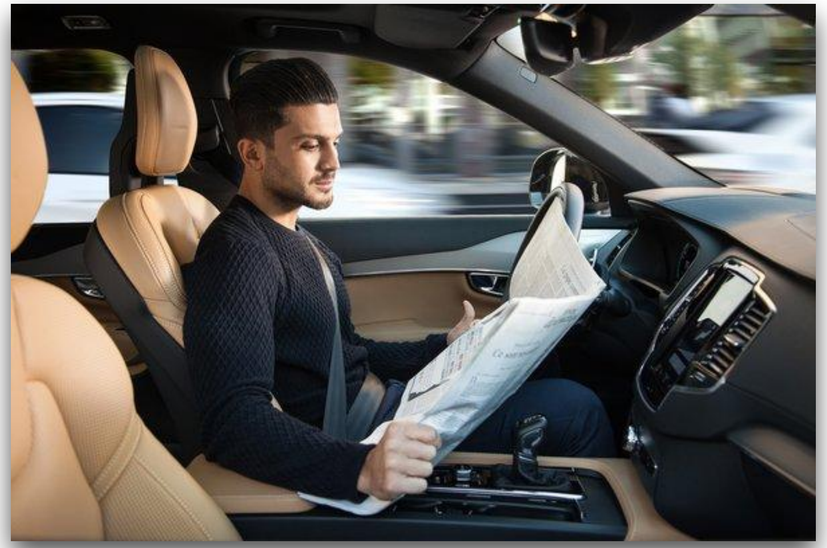
Shin Yoo

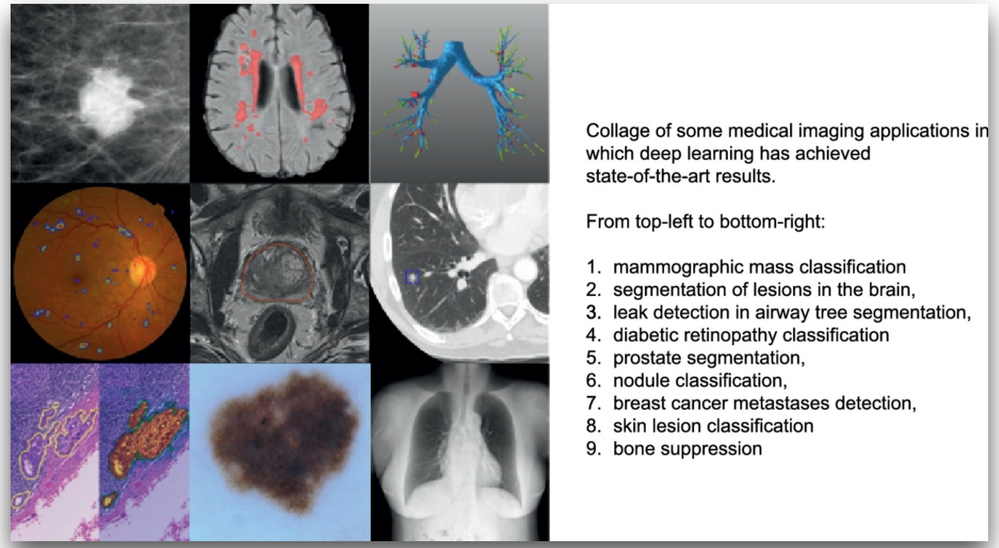
# Machine Learning

- Problems are solved by helping machines discover their own algorithms, without needing to be explicitly told what to do by any human developed algorithms (Wikipedia)
- Waves of recent advances
  - Image Classification (2015~ ish): ImageNet Competition being won by CNNs
  - RL (2016): AlphaGo that combined RL with CNNs and Monte-Carlo Tree Search
  - Transformers (2017): sequence to sequence architecture (e.g., Machine Translation)
  - Large Language Models (2022~): Large Transformers trained with large amounts of data

# ML-based systems are being adopted in safety critical domains.

## Mostly due to their surprising performance…





Collage of some medical imaging applications in which deep learning has achieved state-of-the-art results.

From top-left to bottom-right:

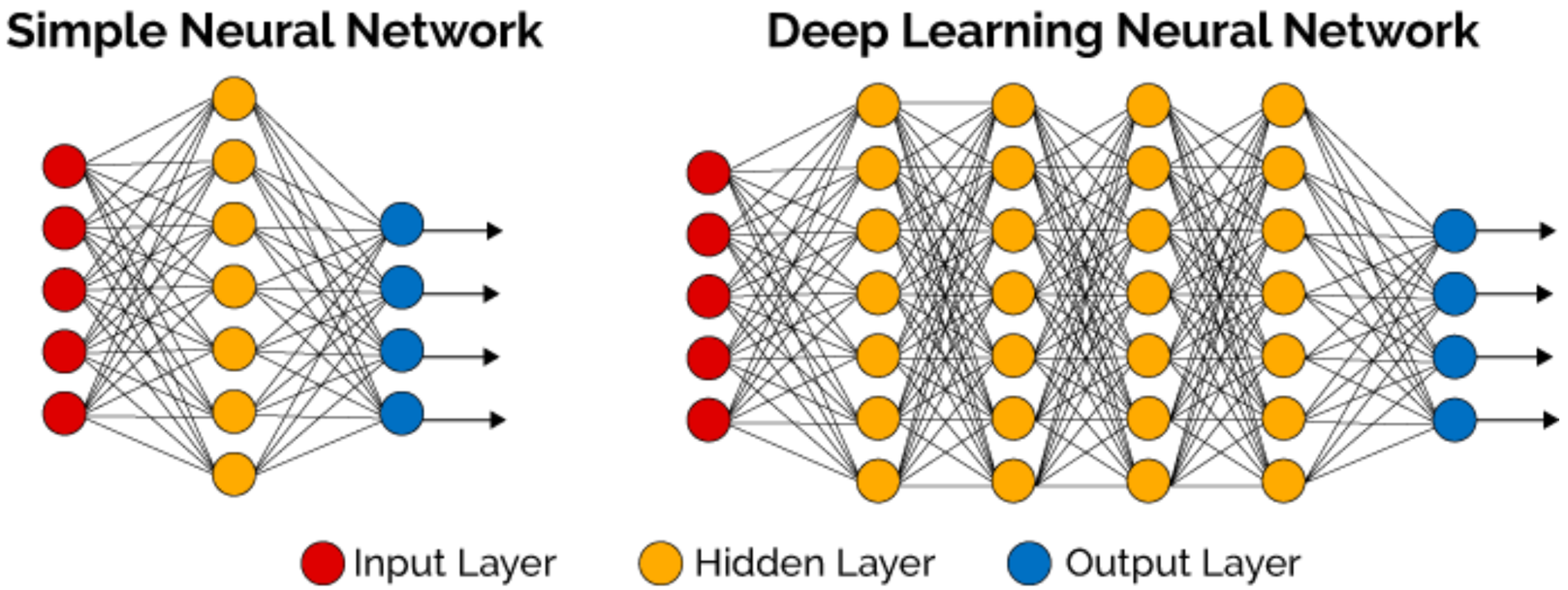
1. mammographic mass classification
2. segmentation of lesions in the brain,
3. leak detection in airway tree segmentation,
4. diabetic retinopathy classification
5. prostate segmentation,
6. nodule classification,
7. breast cancer metastases detection,
8. skin lesion classification
9. bone suppression

Umm, shouldn’t we test these?

# Challenges in Testing DNNs

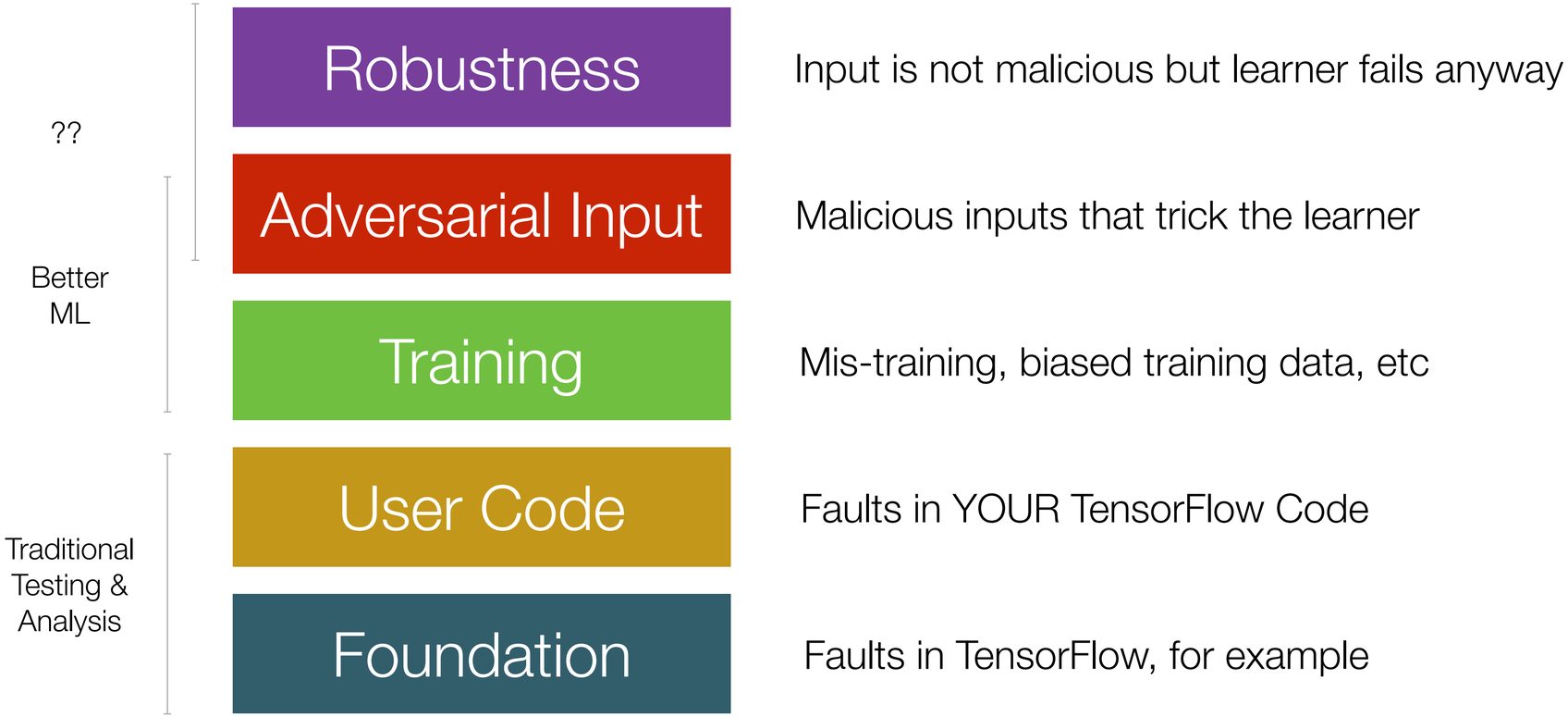
- They are very different computational models from traditional procedural programs that are written by humans.
- Further, ML models are “trained”, not written:
  - Should we expect “learning” to be perfect?
  - What is a bug? If a model’s decision on an input is “undesirable”, is it the input’s fault, or the model’s?
  - These models are often used to replace humans, but testing them requires human judgements (the only source of test oracles).

# Deep Neural Networks




Hardware parallelism (GPUs), advances in back-propagation methods, and other innovations made DNNs surprisingly effective.

# What are the faults?



# What is the test oracle?

- For many practical ML/Deep Learning systems, inputs are raw, real-world perceptions (such as photography/video, voice, etc)
- Currently human judgement (a.k.a. data labelling) is often the only effective test oracle, but this is extremely expensive

# Metamorphic Testing

- In testing, there is a widely known technique that focuses on metamorphic relationships between inputs and outputs
- Given an IO pair for program P, y=P(x), if metamorphic relation f and g hold for input and output, it has to follow that g(y) = P(f(x))
  - For example: if P is the sine function, f(x) = π - x, g(y) = y. That is, if $y_0$ = P($x_0$), y1 = $y_0$ = P(π - $x_0$) = P($x_1$)
- MT cannot replace a full oracle, but if a program violates its own metamorphic properties, **something** is wrong

# Metamorphic Testing for DL

- MT has been applied to test DL robustness
  - If we apply negligible (i.e., bearable by humans) perturbation to the input, the output of the DL system should be the same
- Again, if a DL system violates this, **something** is wrong!
- Interestingly, the metamorphic testing concept is directly linked to the idea of adversarial examples.

# Adversarial Examples

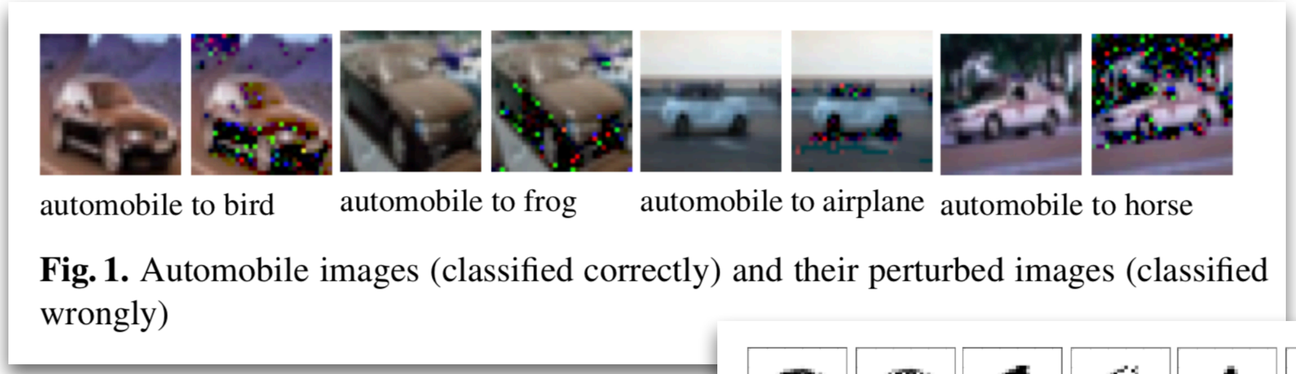
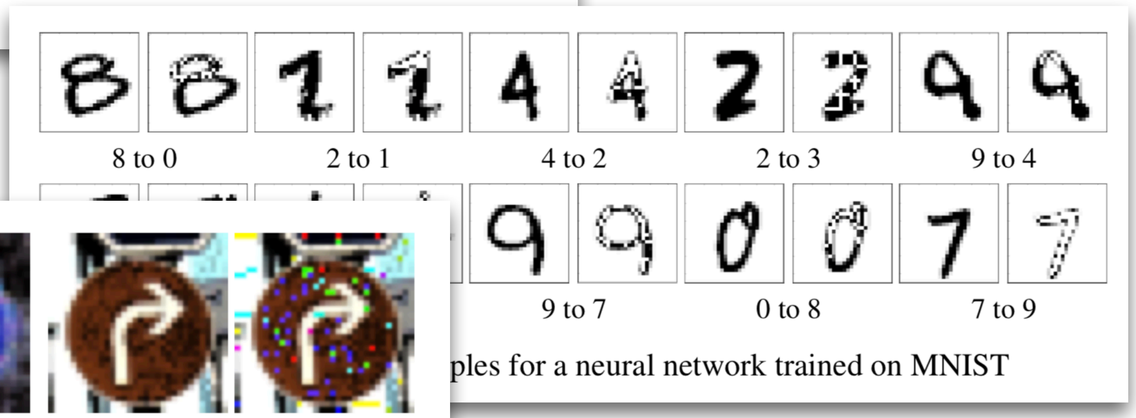
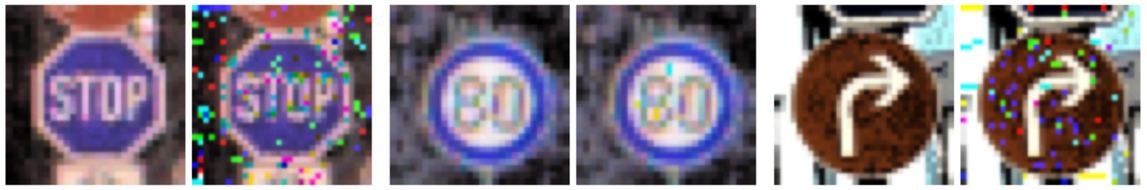



**Fig. 1.** Automobile images (classified correctly) and their perturbed images (classified wrongly)




ples for a neural network trained on MNIST




"stop"
to "30m speed limit"



"80m speed limit"
to "30m speed limit"



"go right"
to "go straight"

**Fig. 11.** Adversarial examples for the network trained on the GTSRB dataset by multi-path search

Safety verification of deep neural networks,
Xiaowei Huang, Marta Kwiatkowska, Sen Wang, Min Wu (https://arxiv.org/abs/1610.06940)

# DNN Robustness

- Adversarial Examples drew a LOT of attention.
- Early DNN testing work from SE was heavily influenced by this: it heavily focused on **robustness** of DNNs, that is, a DNN should be robust against minor perturbations in inputs.
- How would you test the robustness of a given DNN model?

# Input Mutation (or fuzzing) for DNNs

- Suppose a model $M$ outputs $o$ for input $i$, i.e., $M(i) = o$.
- Given an acceptable perturbation, $f$, a robust model should also behave in such a way that: $M(f(i)) = o$.
- Note that this can be interpreted exactly as a metamorphic oracle
- What are acceptable perturbations?
  - Transformations: scaling images, darkening/brightening images, rotating images, etc
  - Noises: Gaussian noise added to image, audio, etc

# Testing Robustness requires Human Judgement

- Many search-based techniques exist for robustness testing of DNNs: find a sequence of input transformations that result in incorrect model output
- To decide whether the result is a problem of robustness, or a problem of something more serious, is not an easy task with no clear line
  - Only humans can decide whether the sequence of transformations creates something that is still “acceptable”
- The found perturbations can be used..
  - To train the model further (adversarial training)
  - To understand the boundaries of robust behaviour

# Test Adequacy for Deep Learning Systems

- Test Adequacy: Is this input adequate for testing? That is, if there is a problem in the target system, will this input reveal it?
  - In general, undecidable in advance (otherwise we would have solved the testing problem)
  - Consequently, in traditional software testing, we rely on certain surrogate metrics, such as coverage (the necessary condition for fault detection)
  - However, what is the necessary condition for fault detection in DNNs???
- So far, we have two groups of thoughts: structural, and non-structural.

# Structural Test Adequacy for DNN

## Essentially, attempt to define structural coverage for neurons

- Various coverage criteria have been suggested:
  - Neuron Coverage: given a set of test input, NC measures % of neutrons that have been activated above a given threshold (e.g., 0.6)
  - k-Multisection Neuron Coverage: mark the range of neuron activation during training, divide the range into k buckets, and count the number of buckets checked during execution of the given input set
  - Strong Neuron Activation Coverage: mark the range of neuron activation during training, and count the number of neutrons that are activated beyond the maximum observed activation value

# Structural Test Adequacy for DNNs

- All designed to improve diversity in test inputs: higher activation in certain neuron —> new DNN behavior —> better chance of detecting anomaly
- This has shown to be effective when evaluating robustness of DNNs: inputs that result in higher structural coverage also tend to reveal abnormal behavior of the DNN under Test
- However, fundamentally limited in the sense that, while we can measure neuron activation, we do not really understand what it means

# Robustness Testing

- Structural coverage has been linked to inputs that can test DNN robustness better.

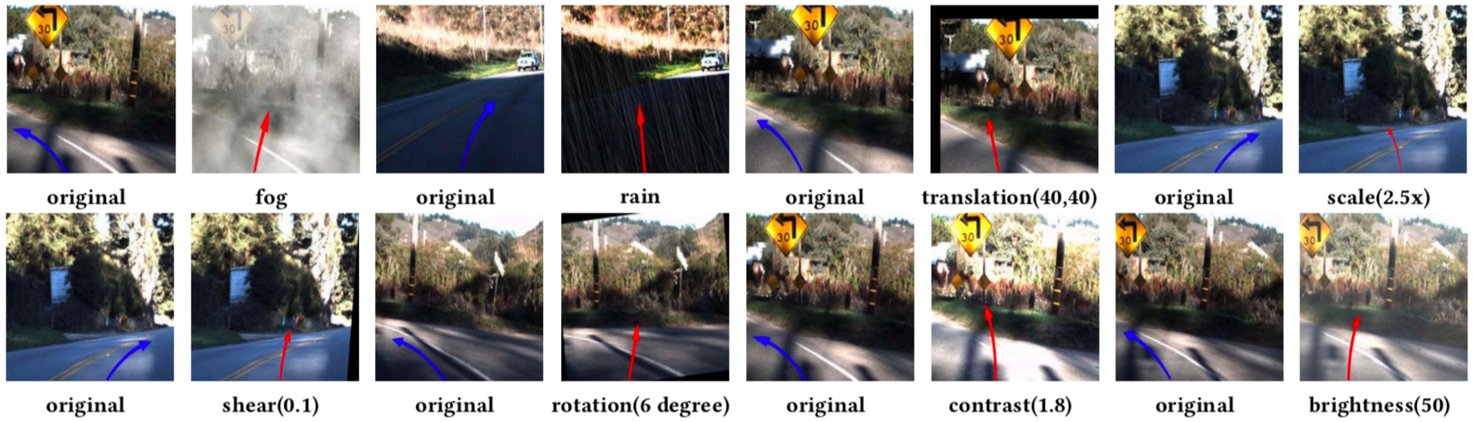


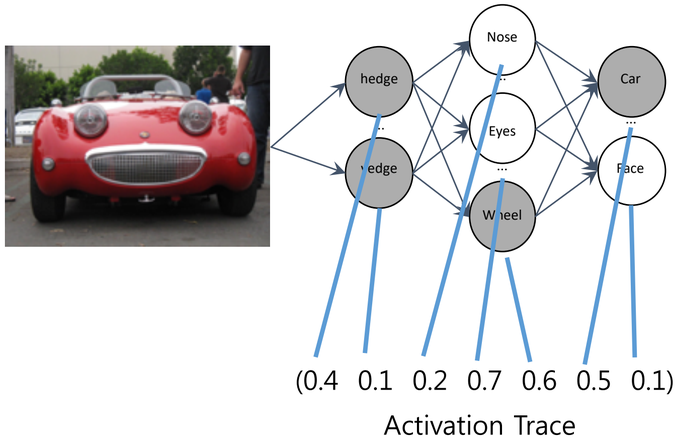

**Figure 7: Sample images showing erroneous behaviors detected by DeepTest using synthetic images. For original images the arrows are marked in blue, while for the synthetic images they are marked in red. More such samples can be viewed at https://deeplearningtest.github.io/deepTest/.**

Y. Tian, K. Pei, S. Jana, and B. Ray. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In Proceedings of the 40th International Conference on Software Engineering, ICSE '18, pages 303–314, New York, NY, USA, 2018. ACM.

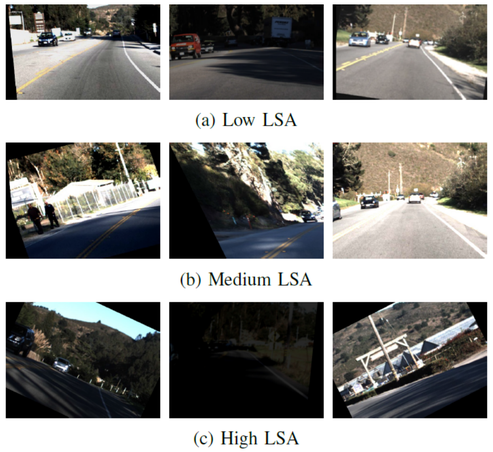
# Non-Structural Test Adequacy for DNNs

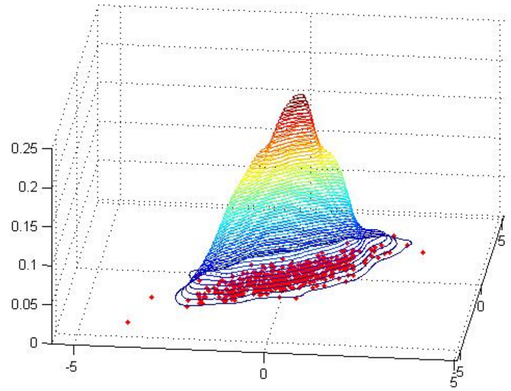
- Ideally, test adequacy should correlate with the likelihood of the input revealing a problem: how do we predict that?
- Surprise Adequacy: we can simply measure how different the new input is from the average training data - if it is very similar, we expect the model to handle the input well; if it is very different, we expect the model to make a mistake
- DeepGini: we can simply look at how confident the input outcome is - if the model is very confident, then likely the decision is okay; if the model is not so confident, then perhaps the decision is more likely to be incorrect

# Surprise Adequacy (Kim et al, ICSE 2019)



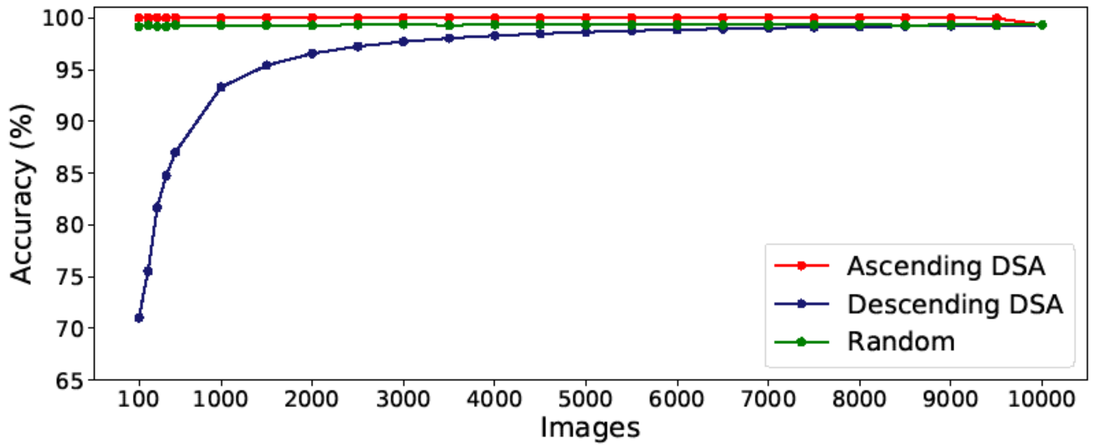

Learnt Knowledge
(from training data)



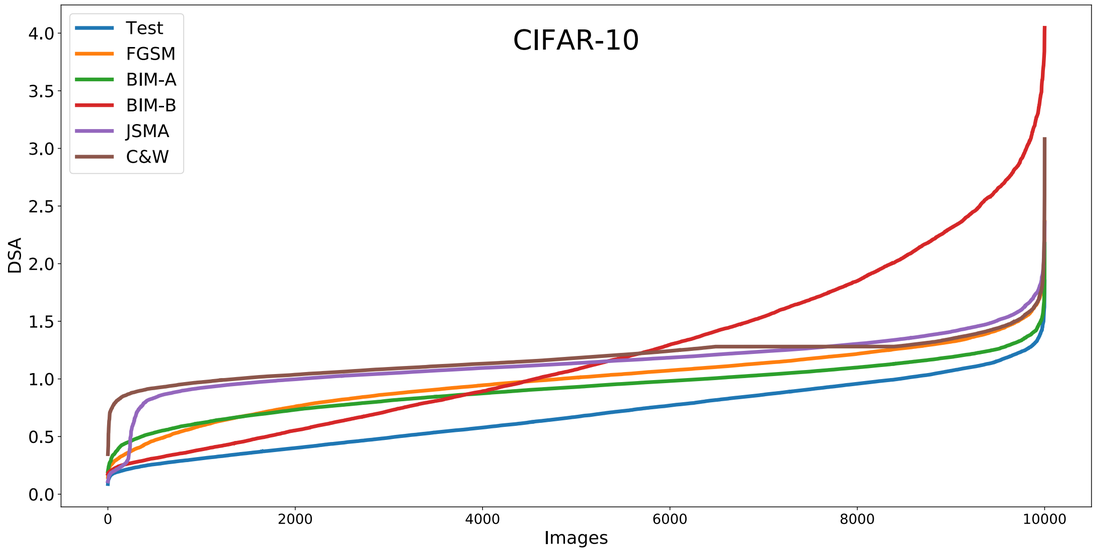




Summarisation
(KDE or point-cloud)

Quantitative Surprise Measure of New Input
Against the Summarisation




**More surprising questions are harder to answer correctly.**

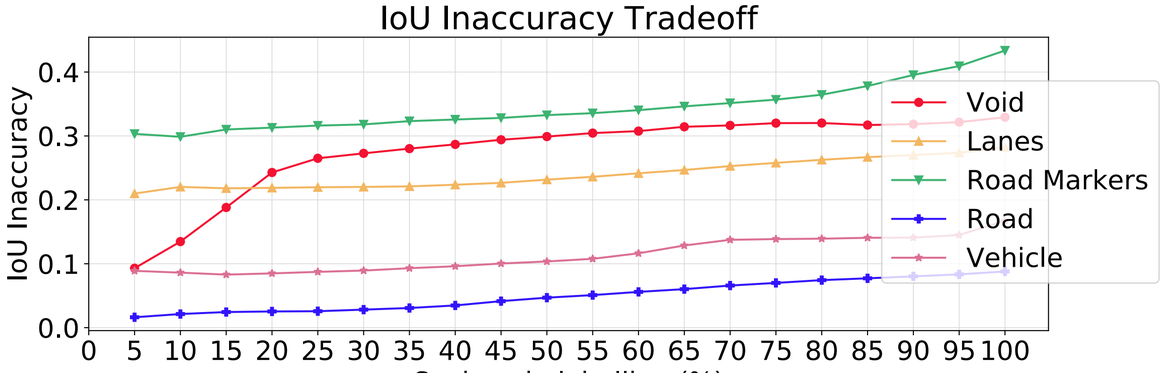


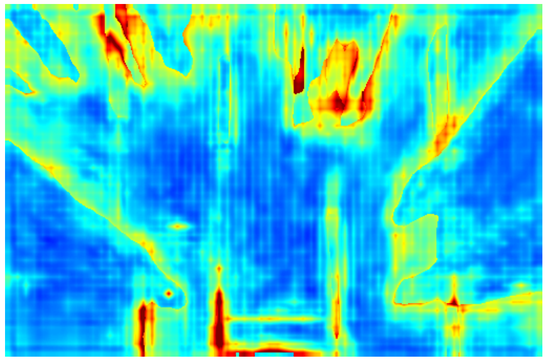

**Trick questions (=adversarial examples) are very surprising.**

# Applying SA

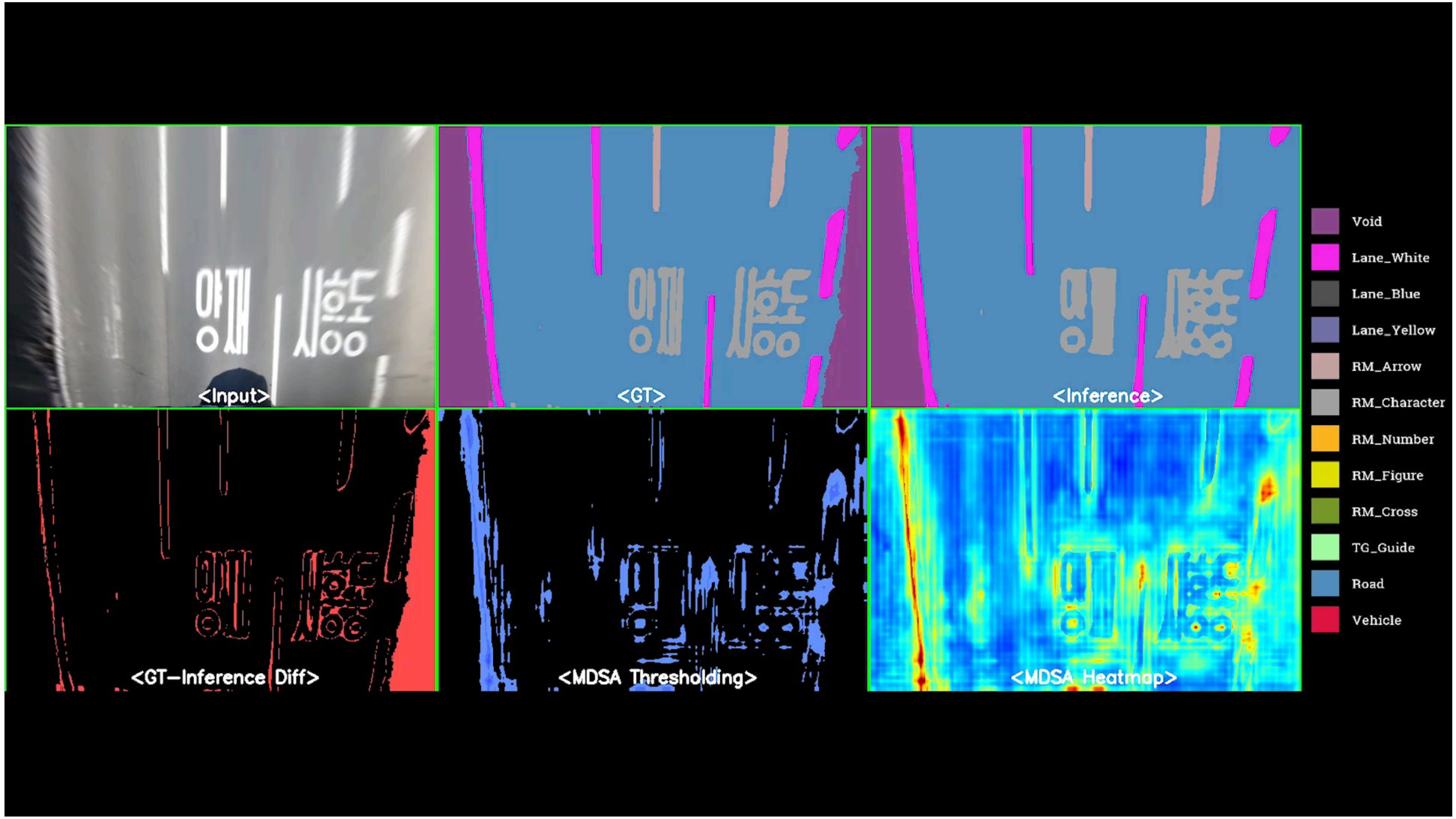
## FSE 2020 Industry Track

- **Reducing DNN Labelling Cost using Surprise Adequacy: An Industrial Case Study for Autonomous Driving** (Jinhan Kim, Jeongil Ju, Robert Feldt, Shin Yoo)
- Collaboration with Hyundai Motors R&D Division
- Low SA input are reasonably reliable, so we can skip labelling them when new input data are collected
- https://arxiv.org/abs/2006.00894



# Video Visualisation



# DeepGini

## Feng et al., ISSTA 2020

- How should we prioritize inputs for DNN classifiers?
- Answer: we should run the ones for which the classifier is **less confident**.
- Quantified confidence is reflected in the final layer of classifiers, i.e., softmax.
- For input $t$ and class $i$, and the probability of $t$ belonging to $i$ denoted as $p_{t,i}$, the uncertainty of $t$ being an instance of $i$ can be expressed as: $p_{t,i}(1 - p_{t,i})$
- Sum this over all $N$ classes:
  - $$\sum_{i \in N} p_{t,i}(1 - p_{t,i}) = \sum_{i \in N} p_{t,i} - \sum_{i \in N} p_{t,i}^2 = 1 - \sum_{i \in N} p_{t,i}^2$$

# DeepGini

## Feng et al., ISSTA 2020

- Consider a binary classification (i.e., two classes).
- Classifier is the most confused when it cannot decide, i.e., $p_{t,0} = p_{t,1} = 0.5$
  - $\xi(t) = 1 - (0.5^2 + 0.5^2) = 0.5$
- Classifier is quite certain that it is one of the two classes, i.e., $p_{t,0} = 0.9, p_{t,1} = 0.1$
  - $\xi(t) = 1 - (0.9^2 + 0.1^2) = 0.18$

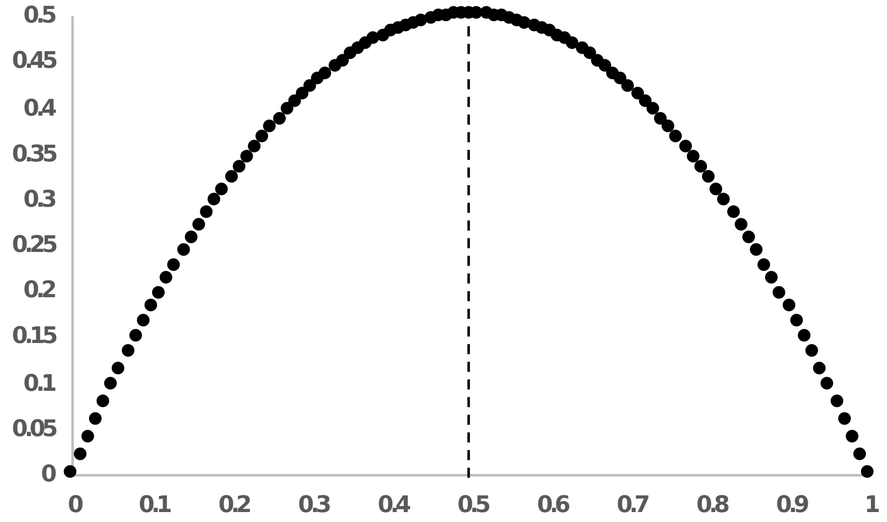



**Figure 3: Distribution of $\xi$ for 2-class problem. X-Axis: the probability that a test input belongs to one of the two classes. Y-Axis: the value of $\xi$.**

# A Big Challenge

- For traditional software, inputs can be either randomly sampled, searched, or synthesised
- For DL systems that interface with the real physical world, inputs have to be collected from the real world
  - You can try random sampling or search, but would it be relevant?

# Sampling Inputs for DNNs

## (Apart from physically collecting them, that is…)

- Model Based Approaches: think of this as a simple parametric generative model - useful for simpler domains.
- Latent Space Approaches: we can exploit latent space embedding - navigate the space while decoding the vectors
- Generative Model Approaches: more sophisticated models such as Stable Diffusion are being explored as the bleeding edge…

# Model Based Input Sampling

## DeepJanus - Riccio & Tonella, FSE 2020

- For evaluating MNIST, an image classifier for hand-written digits, one can design a model of each digits.
- For evaluating steering angle regression model for automous driving, one can design a model of a road (= test input)

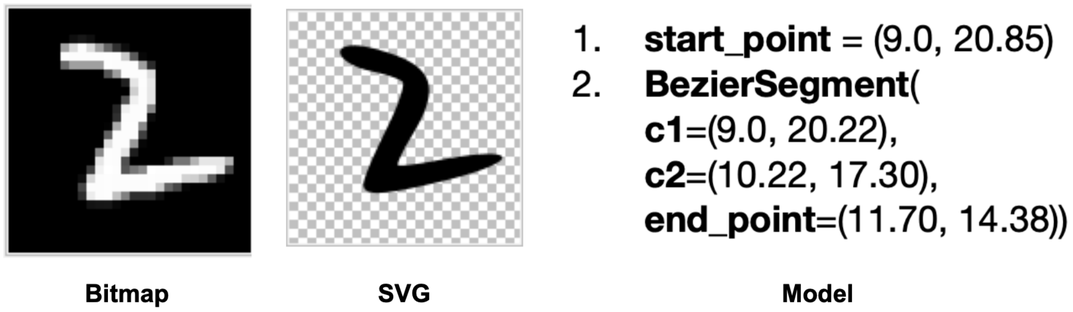



**Figure 2: Bitmap and vector image; model representation of the image returned by Potrace**

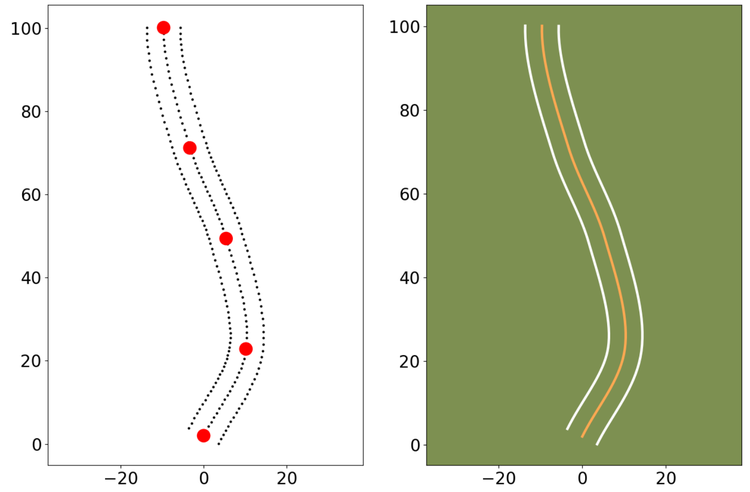



**Figure 3: The model of a road and the corresponding road**

# Latent Space Approaches

## SINVAD - Kang & Yoo, SBST 2020

- Auto-encoders learn the latent space distribution of a domain by encoding and decoding.
- Once we have an autoencoder, we can decode any point in the latent space (i.e., the encoded space)

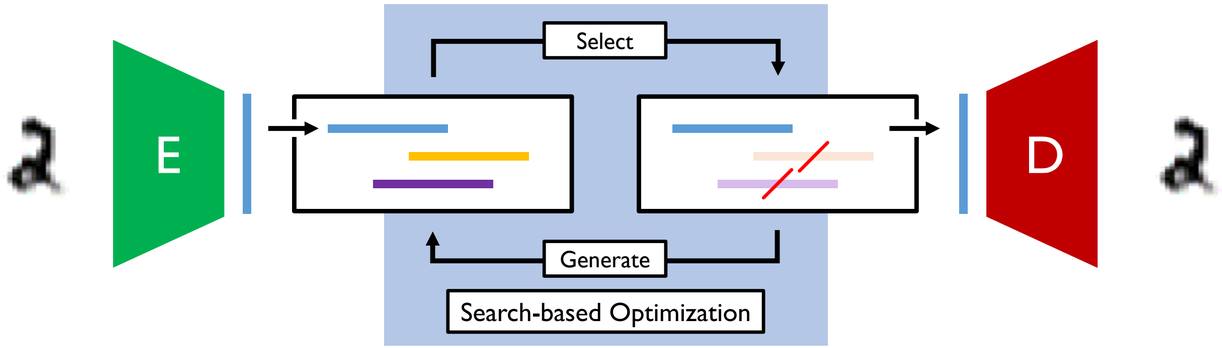



**Figure 1: Diagram of SINVAD. SINVAD uses the VAE encoder (E) to encode images into a *latent space* (bars). Search-based optimization is performed not on raw pixels, but on latent representations. A representation is transformed back to an image using the VAE decoder (D). The use of VAE networks before and after keeps search within the subset of plausible images.**

# Latent Space Approaches

## SINVAD - Kang & Yoo, SBST 2020

- It means that we can interpolate in the latent space, and decode each step, to get “intermediate” images!
- Compare the transition from 4 to 9, using interpolation in 1) latent space, and 2) image space, the original domain.

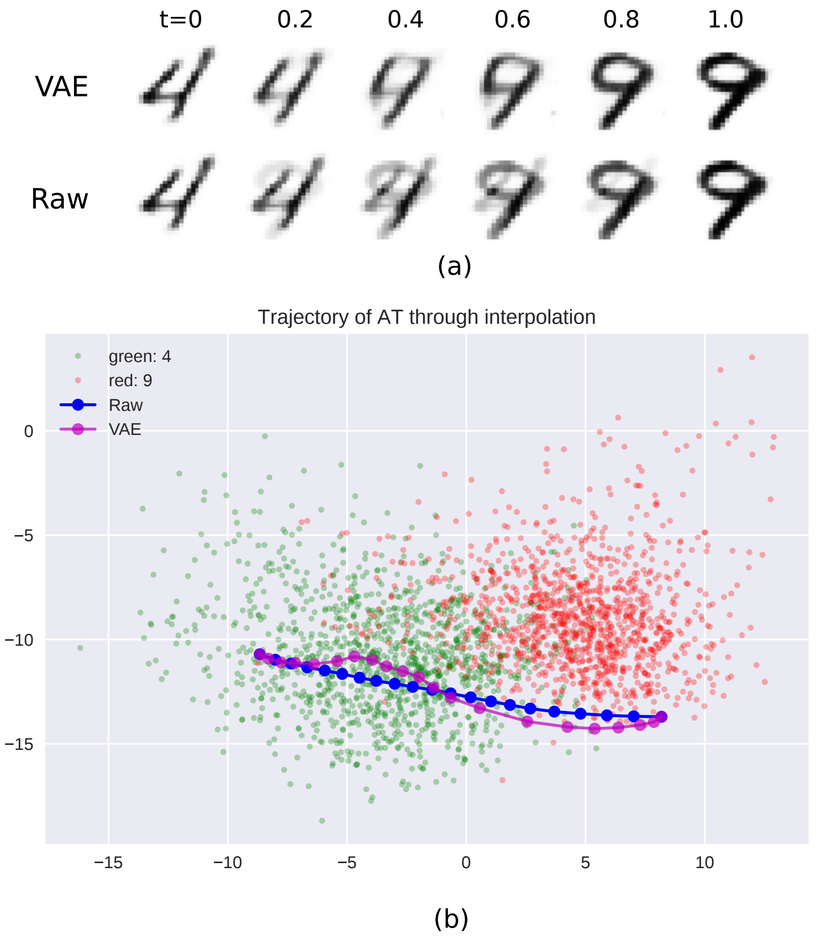



**Figure 5: Trajectories of interpolated images projected into the PCA space.**

# Diffusion Model as an Exploration of Latent Space

- We simply perturb, or interpolate, the latent vector given to the diffusion model
- For example, synthesize a series of images that transition from a pingpong ball to a golf ball…

# Diffusion Model as an Exploration of Latent Space

- Or from a cat to a fruit…!

# Summary

- Major categories of DNN related testing work so far:
  - Robustness Testing (mainly focused on the idea of adversarial examples)
  - Test Adequacy (mainly focused on the idea of finding more interesting inputs)
  - Input Generation (mainly focused on how to automatically synthesize interesting inputs)
- All face the same challenge of having to define what a DNN bug is