Study on DNN Robustness Issues (reports)

Guanqin Zhang
supervisor: Dr.Yulei Sui
[1]UTS School of Computer Science

July 29, 2022

Index

Introduction
DNN creates a successful story

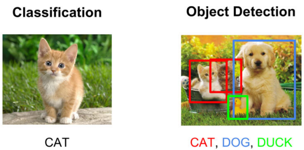


Image Classification

Machine Translation

Auto Driving

Medical Diagnosis

Finance forecasting

[1] Eykholt, Kevin, et al. "Robust physical-world attacks on deep learning visual classification." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

[2] Sharif, Mahmood, et al. "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition." Proceedings of the 2016 acm sigsac conference on computer and communications security. 2016.

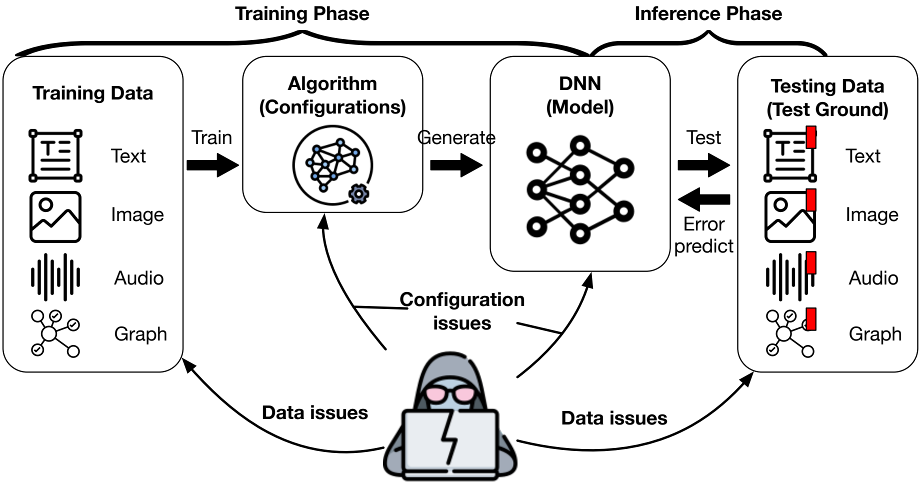
Main principle: DNN should reliably operate in accordance with human intended purpose:

- Should not pose unreasonable risks
- Adpot safety measures the magnitude of potential risks.
- On going risks should be managed appropriately

Main principle: DNN should reliably operate in accordance with human intended purpose:

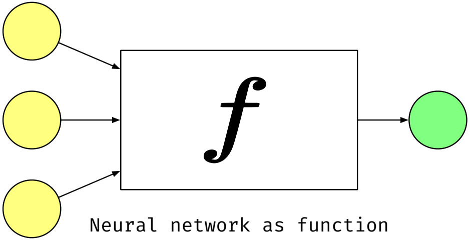
- Should not pose unreasonable risks (Address the robustness issues).
- Adpot safety measures to the magnitude of potential risks (Robustness Measurement).
- On going risks should be managed appropriately (Quantitative robustness evaluation)

Framework of identifying issues



Formalization of the DNNs

A DNN model represents as a function $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$, which accepts an input $x \in \mathbb{X} \subseteq \mathbb{R}^n$, and returns an output $y \in \mathbb{Y} \subseteq \mathbb{R}^m$, where $\mathbb{X}$ and $\mathbb{Y}$ are the inputs and outputs in the real number domain with n and m dimensions, respectfully.



Neural network as function

Formalization of the DNNs

To guarantee the robustness of the function $f : \mathbb{X} \to \mathbb{Y}$ is to ensure

$$x \in \mathbb{X} \ \Rightarrow y = f(x) \in \mathbb{Y}, \tag{1}$$

which involves checking whether input-output relations of the function hold:

$$\mathbb{X} = \{\hat{x} : \|\hat{x} - x\|_{\mathrm{p}} \leq \sigma\}, \tag{2}$$

where $\hat{x}$ is the samples in the neighbourhood of a given input x from a converged DNN model, the metric to measure the disturbance can be any p norm and,

$$\mathbb{Y} = \{y : y_{i^*} > y_j, \forall j \neq i^*\}, \tag{3}$$

where the desired label is i^* from human purpose.

- **Perturbed inputs**: During the training period, perturbed inputs are commonly imposed or introduced to mislead the learning process. In response to the perturbations, a robust DNN can be formalized as:

$$\forall x \in \mathbb{X}, \hat{x} \in \mathbb{X}, \|x - \hat{x}\|_{\mathrm{p}} < \sigma \Rightarrow f(\hat{x}) = y \in \mathbb{Y}, \tag{4}$$

where $\hat{x}$ denotes the perturbed inputs under p normalization with σ distance (the degree of perturbations) to the original input x.

Here, a robust DNN enables large decision boundaries to tolerate input perturbations.

- **Perturbed outputs**: If the training dataset contains corrupted or fuzzed labels ($\hat{y}$) under δ distance to the human-desired labels (y) and deviated by τ times to the inference result:

$$\forall (x, \hat{y}) \in (\mathbb{X}, \mathbb{Y}), \hat{y} = \tau \cdot f(x), \|f(x) - \hat{y}\|_{\mathrm{p}} < \delta \Rightarrow f(x) = y \in \mathbb{Y}. \tag{5}$$

Here, a robust model ensures that the model f still outputs the correct y.

- **Configuration perturbations**:

$$\forall x \in \mathbb{X}, \theta \in \Theta, \hat{\theta} \in \Theta, \|\theta - \hat{\theta}\|_{\mathrm{p}} < \eta \Rightarrow f_{\theta}(x) = f_{\hat{\theta}}(x) = y \in \mathbb{Y}, \tag{6}$$

where $\hat{\theta}$ denotes the configuration under p normalization with η distance (configuration differences) to the configuration θ, and η denotes the boundary.

Here, the non-determinism from configurations (e.g., model initialization and hyperparameters) is another reason for the variance of DNNs. A trained DNN is said to be robust if it is less ambiguous.

Issue Factors

Table 2: Robustness affecting factors and the corresponding settings

Surface	Objective	Factor	Setting
Data (F_D)	Input x	F_1 Adversarial attack	Perturbation distance (σ) Perturbation ratio (r_p)
	Output y	F_2 Label flipping attack F_3 Label noise injection	Flipping ratio (r_f) Noise ratio (r_n)
Configuration (F_C)	Model parameter θ_p	F_4 Weight perturbation F_5 Bias perturbation	Perturbation distance (η_w) Perturbation distance (η_b)
	Model structure θ_s	F_6 Conv layer modification[1]	Number of layers (η_{cl}) filter size (η_{fs})
		F_7 FC layer modification[2]	Number of layers (η_{fl}) Number of neurons (η_{fn})
	...	...	...

[1] Conv layer denotes the one-dimensional (1D) and two-dimensional (2D) layers in DNNs.
[2] FC layer denotes the fully connected layer in DNNs.

Study Objective

In Table 2, we consider the perturbation surface (i.e., data (F_D) and configuration (F_C)) to modify specific objectives (o), i.e., input, output, model parameter and structure, through a total number of N factors $\mathbf{F} = \{\mathbf{F_1} \ldots \mathbf{F_N}\}$.

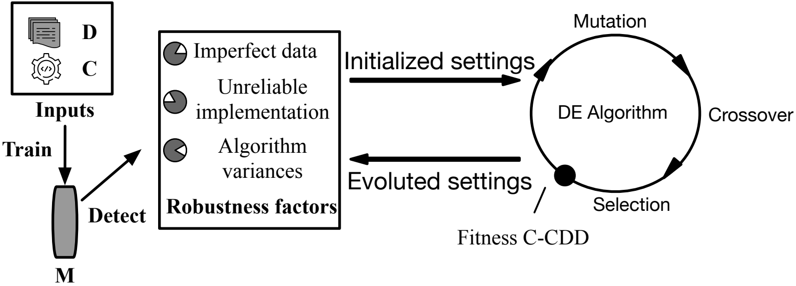
The collection of all subset without the empty set $fac = \{P(F) \setminus \{\varnothing\}\}$, where $P(F)$ is the power set of all single factors F, contains $\mathbf{2^n - 1}$ possible combinations.

Assuming $\mathcal{P}_{S \in fac}(o)$ is the function to manipulate the objective o, where S contains single or combined factors, $\mathcal{P}_{\{F_1, F_5\}}((x, \theta_p))$ represents the manipulation strategies to generate perturbed inputs and bias simultaneously with respect to the perturbation distance (σ), ratio (r_p), and bias perturbation distance (η_b).

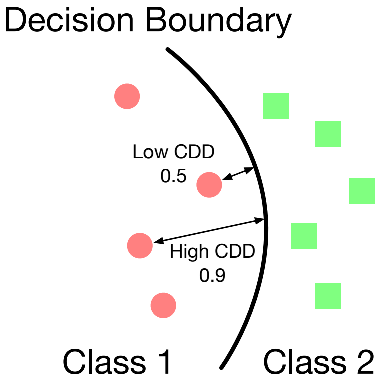
DNN is **non-linear**, so we can't infer the combinational effects based on single effects to find the most affecting combinations.

Factor	Settings	Searching Space
F_1	Perturbation distance (σ)	$[0, 10]$
F_2	Flipping ratio (r_f)	$[0, 80\%]$
F_3	Noise ratio (r_n)	$[0, 50\%]$
...	...	...

Differential Evolution (standard)



Decision Boundary



C-CDD

Following that, we define *Cumulative Confidence Decision Distance* (C-CDD) to measure the confidence of all predicted samples. Meanwhile, C-CDD reflects the relative distance of the inputs from the decision boundary to the human desired classes. In this project, the C-CDD can be presented as:

$$\mathbb{E}_{(x,y)\in\mathcal{X}\times\mathcal{Y},\theta\in\Theta} CDD(x, f_\theta) = \mathbb{E}_{(x,y)\in\mathcal{X}\times\mathcal{Y},\theta\in\Theta} \| f_\theta^i(x) - f_\theta^j(x) \|_{\mathtt{p}}, \tag{7}$$

where x, y, θ are the corresponding perturbed samples if perturbation exists otherwise the original ones, $\mathcal{X}, \mathcal{Y}, \Theta$ denotes the inputs, outputs and configuration set, i denotes the human desired class and j is the non-human desired class with the maximum prediction probability.