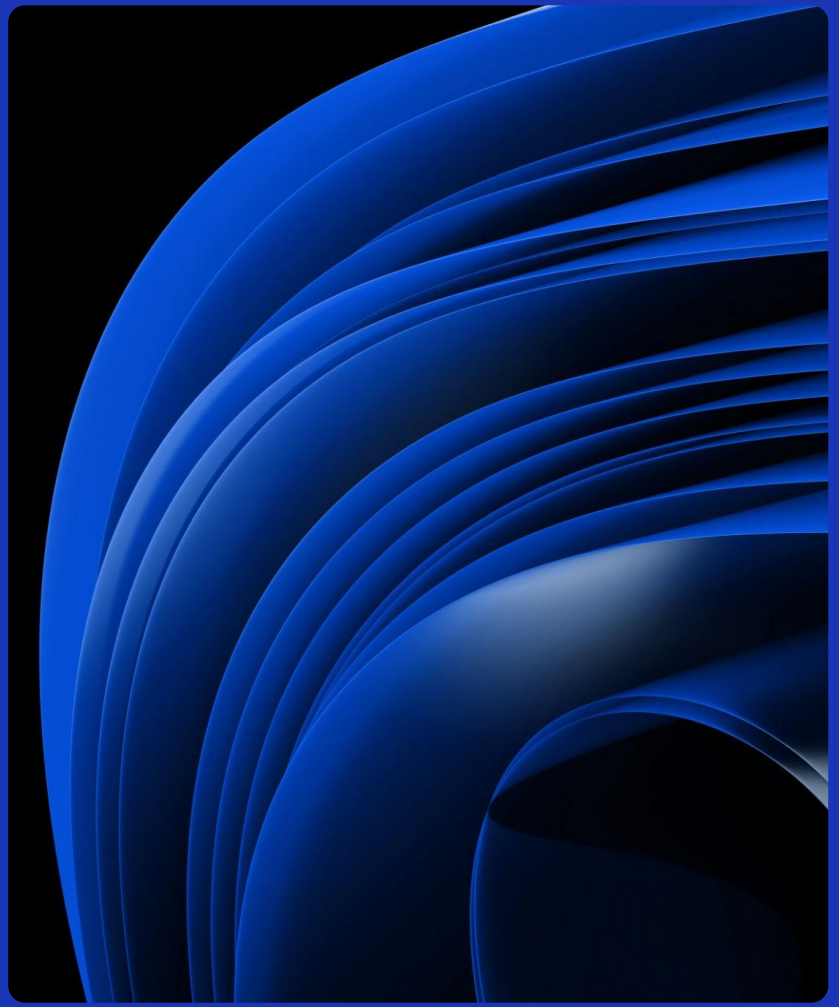
Backdoor MBTI

A Backdoor Learning Multimodal Benchmark Tool Kit for Backdoor Defense Evaluation



What Backdoor Attacks?

- Backdoor attacks involve inserting hidden vulnerabilities (backdoors) into deep learning models.
- When data with a specific trigger (added by an attacker) is processed, the model misclassifies it.
- However, normal data yields expected results, making the attack hard to detect.
- These attacks started in computer vision but have now expanded to natural language processing (NLP) and speech recognition.

Challenges in Backdoor Defense

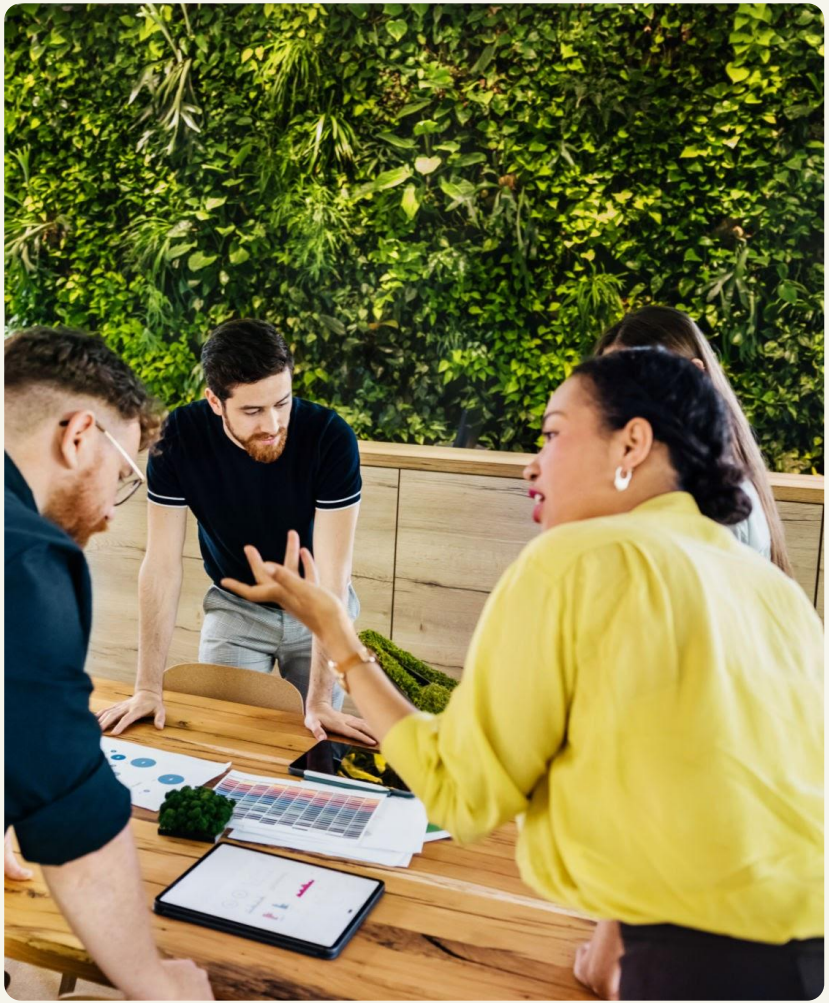
- Current backdoor defenses are usually designed for a single type of data (e.g., images or text).
- However, multimodal learning (which processes different types of data together, like facial recognition, sentiment analysis, action recognition) lacks proper backdoor security benchmarks.
- There is no standardized way to study or counteract backdoor attacks in multimodal models.

BackdoorMBTI

First multimodal backdoor benchmark for evaluating backdoor attacks across three different types of data from eleven datasets. Provides a systematic pipeline for backdoor research, including:

- Data processing
- Data poisoning (inserting backdoors)
- Backdoor training
- Evaluation of attacks and defenses

Toolkit supports evaluating backdoor attacks across different types of data



- **Growing threats from backdoor attacks**, which are rapidly evolving and present real-world risks. Users may encounter poison data, where **attackers infiltrate specific triggers into datasets before training**, potentially impacting all users of these compromised datasets. As neural networks scale up, **training costs also increase, forcing users to rely on third-party training resources that may lack security,** thereby highlighting the real threat posed by backdoor attacks in practical scenarios.

- **Current benchmarks are limited** because they:
  - Focus only on **one modality**.
  - Lack **standardized evaluation methods**.
  - Ignore **real-world issues** like noisy data or label errors.

- BackdoorMBTI **supports three modalities and incorporates eleven representative datasets, seventeen attacks, and seven defense methods.**

- BackdoorMBTI provides **easy-to-access backdoor datasets and establishes a standard benchmark for evaluating backdoor defenses in multimodal settings**.

- BackdoorMBTI **takes into account factors such as low-quality data and erroneous labels, which align with real-world scenarios.**

Table 1: The modality support in current benchmarks.

Benchmarks	Image	Text	Audio
TrojAI [27]	✓		
TrojanZoo [53]	✓		
BackdoorBench [79]	✓	✓	
BackdoorBox [37]	✓		
OpenBackdoor [12]		✓	
Backdoor101 [2]	✓	✓	
Ours	✓	✓	✓

Backdoor attacks

1. **Data Poisoning Attacks** – The attacker **modifies the training data** by inserting hidden triggers, without changing the training process.

2. **Training Control Attacks** – The attacker **controls both the data and the training process**, allowing them to embed more sophisticated backdoors.
    - Example: A hacker trains a voice assistant (like Siri or Alexa) themselves. They secretly program it to ignore emergency words (like “call 911”) if spoken in a certain tone.

3. **Model Modification Attacks** – The attacker **directly alters the AI model itself**, adding hidden vulnerabilities.
    - Example: A hacker adds secret code to a spam filter AI so that emails from a certain sender always avoid the spam folder, even if they are scams.

Backdoor Defense

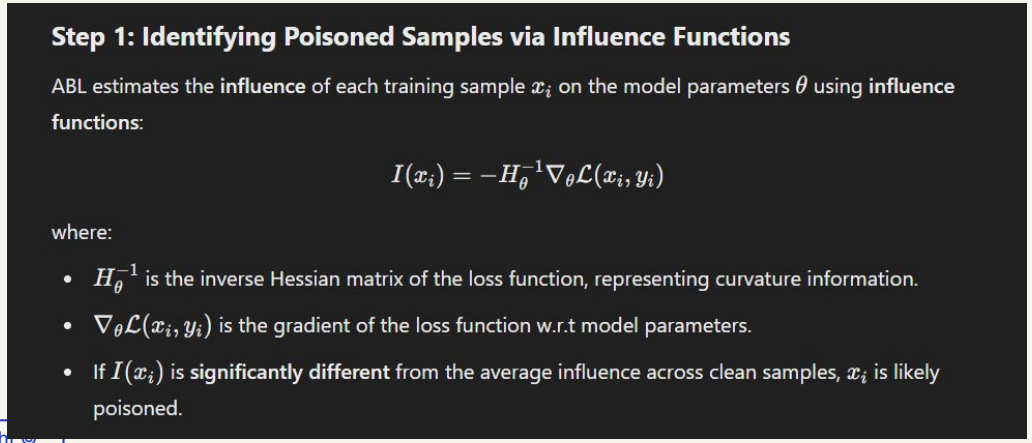
Preprocessing: Remove hidden backdoor triggers from data before they reach the AI model

- **ShrinkPad** : This method changes the size or padding of images to make the backdoor trigger ineffective.
- **AutoEncoder Method** : A type of AI model (autoencoder) is placed before the main AI model to clean the input data.
- **DeepSweep** : Researchers tested 71 different data augmentation techniques (such as blurring, cropping, and rotating images) to prevent backdoor triggers from working.
  - **Spatial Transformations** – Change the position or orientation of objects in the image:
    - **Cropping:** Randomly removes parts of the image, possibly eliminating the backdoor trigger.
    - **Rotation:** Rotates the image by a small angle, making position-dependent triggers ineffective.
    - **Flipping:** Mirrors the image horizontally or vertically, disrupting asymmetrical triggers.
  - **Noise and Blurring** – Modify pixel values to obscure backdoor patterns:
    - **Gaussian Blur:** Smooths the image, making small, specific patterns less visible.
    - **Additive Noise:** Injects random noise, disrupting pixel-based triggers.
  - **Color and Contrast Adjustments** – Alter the appearance of the image:
    - **Brightness/Contrast Changes:** Modifies the intensity of pixels, affecting color-based triggers.
    - **Hue and Saturation Shifts:** Changes the color distribution, making color-based triggers ineffective.

Backdoor Defense

Poison Suppression: Preventing Backdoor Injection During Training
Backdoor attacks modify a model by injecting poisoned data during training. Poison suppression techniques detect and remove these malicious changes before they affect the final model

- **ABL (Adaptive Backdoor Learning)** : ABL follows two steps:
  - **Backdoor Sample Identification:** It detects poisoned samples by analyzing their influence on model gradients.
  - **Unlearning:** It removes backdoor effects by adjusting the loss function.



The **Hessian matrix** is a square matrix of second-order partial derivatives of a function. It describes how the gradient (first derivative) of a function changes, which helps in analyzing **curvature**, optimization, and stability.

Definition

For a function $f : \mathbb{R}^n \to \mathbb{R}$, the Hessian matrix $H_f(x)$ is given by:

$$H_f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

where:

- $\frac{\partial^2 f}{\partial x_i^2}$ are second-order derivatives (curvature along each axis).
- $\frac{\partial^2 f}{\partial x_i \partial x_j}$ are mixed partial derivatives.

Curvature-based optimization refers to methods that use second-order information (like the **Hessian matrix**) to optimize a function. These methods leverage the curvature of the function to determine how to update parameters during the optimization process, often leading to faster and more efficient convergence compared to first-order methods like **gradient descent**.

Why Curvature Matters in Optimization?

When optimizing a function, the **gradient** gives us the direction of steepest ascent (or descent), but it doesn't tell us how the function is curving. The **curvature** tells us how the function behaves near a point:

- If the curvature is **high**, the function changes rapidly in that direction.
- If the curvature is **low**, the function changes slowly in that direction.
- If the curvature is **positive**, we are moving toward a minimum.
- If the curvature is **negative**, we are moving toward a maximum.

Using **curvature** allows us to make better decisions about how far to move and in which direction, which can lead to faster convergence.

Step 2: Unlearning Poisoned Samples

Once poisoned samples are detected, ABL modifies the training process by adjusting the objective function:

$$\mathcal{L}'(\theta) = \mathcal{L}(\theta) + \lambda \sum_{i \in P} \|\nabla_\theta \mathcal{L}(x_i, y_i)\|^2$$

where:

- P is the set of detected poisoned samples.
- The added term **reduces the impact** of poisoned samples by penalizing their gradients.
- λ is a weighting factor to balance unlearning and normal training.

Determining the Value of λ:

1. **Small λ (Low Regularization)**:
    - When λ is small, the model will place more emphasis on the clean data, with little penalty on the poisoned samples.
    - This might not effectively suppress the backdoor samples, especially if they are significantly influencing the training process.
    - **Risk**: The backdoor pattern may persist in the model, leading to a high attack success rate.
2. **Large λ (High Regularization)**:
    - A large λ applies more pressure on the model to suppress the poisoned samples. This results in stronger unlearning of the backdoor effects.
    - However, if λ is too large, it can distort the learning from clean data, making the model underfit, or it can disrupt the optimization process.
    - **Risk**: The model might struggle to learn the correct patterns from clean data if the backdoor suppression is too aggressive.

Backdoor Defense

Poison Suppression: Preventing Backdoor Injection During Training

- **DBD (Debiased Backdoor Detection)**: Poisoned data points tend to cluster together in feature space.They prevent this clustering by:
  - Using **self-supervised learning** (training without labels) to learn general features that are not affected by backdoors.
  - Applying **semi-supervised fine-tuning**, which helps separate real features from backdoor-induced changes.

Step 2: Semi-Supervised Fine-Tuning

Once self-supervised training is complete, DBD fine-tunes the model using clean labels while **reducing the weight of suspicious samples**:

$$\mathcal{L}_{\text{DBD}} = \sum_{(x_i,y_i)\in C} \mathcal{L}(f(x_i), y_i) + \beta \sum_{(x_j,y_j)\in P} \mathcal{L}(f(x_j), y_j)$$

where:

- C is the set of clean samples.
- P is the set of suspected poisoned samples.
- β is a **small weighting factor**, reducing the effect of potentially poisoned samples.

Backdoor Defense

Backdoor Removal and Mitigation:

- **Eliminate or reduce the impact** of backdoor effects in models, often achieved through **fine-tuning** and **pruning**.
- **Fine-Tuning (FT)**: Retrains the model on clean data while keeping most of the original parameters. This helps reduce backdoor effects without losing useful knowledge from the model.
- **Fine-Pruning (FP)**: Involves **removing neurons or weights** that are highly sensitive to backdoor attacks, which helps mitigate backdoor triggers.

The **Trigger Reverse** method in backdoor attack detection generally focuses on identifying the hidden triggers or patterns within the model that cause it to behave inappropriately when presented with a specific input. Below is an explanation of the mathematics behind some of the key techniques used in **Trigger Reverse** methods, focusing on **NC (Nonlinear Channel)** and **FeatureRE (Feature Reverse Engineering)**.

1. Nonlinear Channel (NC) Method:

This method identifies backdoor triggers by **optimizing** a specific input pattern that causes a model to produce a targeted outcome (e.g., misclassification) when the backdoor is triggered.

Mathematical Formulation:

- Let the model's prediction be represented by $f(\mathbf{x}; \theta)$, where:
  - $\mathbf{x}$ is the input,
  - θ represents the model parameters.

- In a backdoored model, for a specific input $\mathbf{x}_{trigger}$, the model will output a different classification (i.e., a malicious classification) compared to what it would output for benign inputs.
- The goal of the **optimization** step in NC is to identify the **trigger input** $\mathbf{x}_{trigger}$ that maximizes the probability of the target output class y_{target}. This can be formulated as:

$$\mathbf{x}_{trigger} = \arg\max_{\mathbf{x}} \mathbb{P}(f(\mathbf{x};\theta) = y_{target})$$

  - This means we search for an input $\mathbf{x}_{trigger}$ such that the model classifies it into the target class y_{target}.
- In practice, this optimization can be performed using gradient-based methods:

$$\nabla_{\mathbf{x}} \mathbb{L}(f(\mathbf{x};\theta), y_{target})$$

where $\mathbb{L}$ is the loss function (usually cross-entropy loss in classification tasks), and $\nabla_{\mathbf{x}}$ is the gradient with respect to the input $\mathbf{x}$. The input $\mathbf{x}$ is updated iteratively to increase the loss for the target class.

2. Feature Reverse Engineering (FeatureRE) Method:

The FeatureRE method attempts to identify backdoor triggers using constraints within the **feature space**, looking for associations between specific features and backdoor behavior. This relies on the idea that backdoors in neural networks are often linked to specific **hyperplanes** in the feature space.

Mathematical Formulation:

- Let the model output $f(\mathbf{x}; \theta)$ be a function of the input $\mathbf{x}$, and $\mathbf{z} = \phi(\mathbf{x})$ be the feature space representation of the input $\mathbf{x}$, where ϕ is the feature extractor (e.g., an intermediate layer in a deep neural network).
- A backdoor trigger often causes a shift in the feature space representation $\mathbf{z}$ of the input, meaning that the model's decision boundary for the malicious class (target class) is **linearly separable** from the benign classes after the trigger is applied.
- We aim to find the trigger by analyzing the **feature space** for the specific **hyperplanes** that separate the backdoor class from the others. The goal is to find a trigger pattern that lies close to this decision boundary in the feature space.

  The feature space constraint can be expressed as:

$$\mathbf{z}_{trigger} = \arg\min_{\mathbf{z}} \|\mathbf{z} - \mathbf{z}_{benign}\|_2$$

Additionally, backdoor triggers can be associated with **perturbations** in the feature space, meaning that small changes in the feature vector $\mathbf{z}$ can cause the model to make a misclassification. This can be formulated as:

$$\mathbf{z}_{backdoor} = \mathbf{z}_{benign} + \delta$$

where δ is the perturbation vector that represents the backdoor trigger in the feature space. The backdoor trigger is identified by finding small perturbations in the feature space that lead to a different classification outcome.

Table 2: An overview of 11 datasets included in BackdoorMBTI.

Task	Dataset	Modality	Classes	Total Instances	Used In Experiment
Object Classification	CIFAR10 [30]	Image	10	60,000	[8, 15, 20, 24, 25, 35, 36, 38, 42, 51, 52, 57, 59, 61, 62, 64, 66, 68–71, 75–77, 80, 81, 85, 86, 88–91]
	TinyImageNet [14, 31]	Image	200	110,000	[2, 8, 15, 22, 25, 34, 35, 42, 44, 47, 57, 59, 62, 64, 76, 89, 90]
Traffic Sign Recognition	GTSRB [65]	Image	43	51,839	[4, 7, 15, 17, 22, 26, 35, 36, 42, 44, 47, 51, 52, 57, 61, 66, 68, 74–77, 84–86, 91]
Facial Recognition	CelebA [46]	Image	8*	202,599	[47, 51, 77]
Sentiment Analysis	SST-2 [63]	Text	2	11,855	[12, 54]
	IMDb [49]	Text	2	50,000	[2, 6, 12, 17, 47, 83]
Topic Classification	DBpedia [1]	Text	14	630,000	[6]
	AG's News [87]	Text	4	31,900	[12, 54]
Speech Command Recognition	SpeechCommands [78]	Audio	35	105,829	[17, 40, 81]
Music Genre Classification	GTZAN [72]	Audio	10	1,000	-
Speaker Identification	VoxCeleb1 [50]	Audio	1251	100,000	-

*CelebA is a face attributes classification dataset, and all the face attributes are labeled in a bin format, in order to do the multi-class classification task on it, we follow the same settings in [77].

Table 3: The implemented attacks in BackdoorMBTI.

Modality	Attack	Visible	Pattern	Add	SS*
Image	BadNets [19]	Visible	Local	Yes	No
	BPP [77]	Invisible	Global	Yes	No
	SSBA [34]	Invisible	Global	No	Yes
	WaNet [51]	Invisible	Global	No	Yes
	LC [71]	Invisible	Global	No	Yes
	SBAT [16]	Invisible	Global	No	Yes
	PNoise [10]	Invisible	Global	Yes	Yes
	DynaTrigger [60]	Visible	Local	Yes	Yes
Text	BadNets [19]	Visible	Local	Yes	No
	AddSent [13]	Visible	Local	Yes	No
	SYNBKD [55]	Invisible	Global	No	Yes
	LWP [32]	Visible	Local	Yes	No
	BITE[82]	Invisible	Local	Yes	Yes
Audio	Blend [9]	-	Local	Yes	No
	DABA [41]	-	Global	Yes	No
	GIS [28]	-	Global	No	No
	UltraSonic [29]	-	Local	Yes	No

*sample specific, whether the trigger is sample specific.

Table 4: The implemented defenses in BackdoorMBTI.

Defense	Input			Stage		Output		
	BM	CD	PD	IT	PT	CM	CD	TP
STRIP [17]	✓	✓			✓		✓	
AC [5]	✓		✓		✓	✓	✓	
FT [40]	✓	✓		✓		✓		
FP [40]	✓	✓			✓	✓		
ABL [35]	✓		✓	✓		✓		
CLP [89]	✓			✓		✓		
NC [74]	✓	✓			✓	✓		✓

a) Input: BM, Backdoor Model; CD, Clean Dataset; PD, Poison Dataset, a dataset including malicious backdoor data in it.
b) Stage: IT, In Training stage; PT, Post Training stage.
c) Output: CM, Clean Model, backdoor model after migration or a secure-training model; CD, Clean Dataset, a sanitized dataset; TP, Trigger Pattern, backdoor trigger pattern reversed by the defense.

- **Stage:** Indicates when the defense is applied in the machine learning pipeline.
  - **IT (In Training):** The defense mechanism is applied during the model's training process. This usually involves modifying the training algorithm or data to prevent backdoors from being learned.
  - **PT (Post Training):** The defense is applied after the model has been trained. This could involve analyzing the model, modifying its parameters, or filtering its outputs to remove the effects of the backdoor.

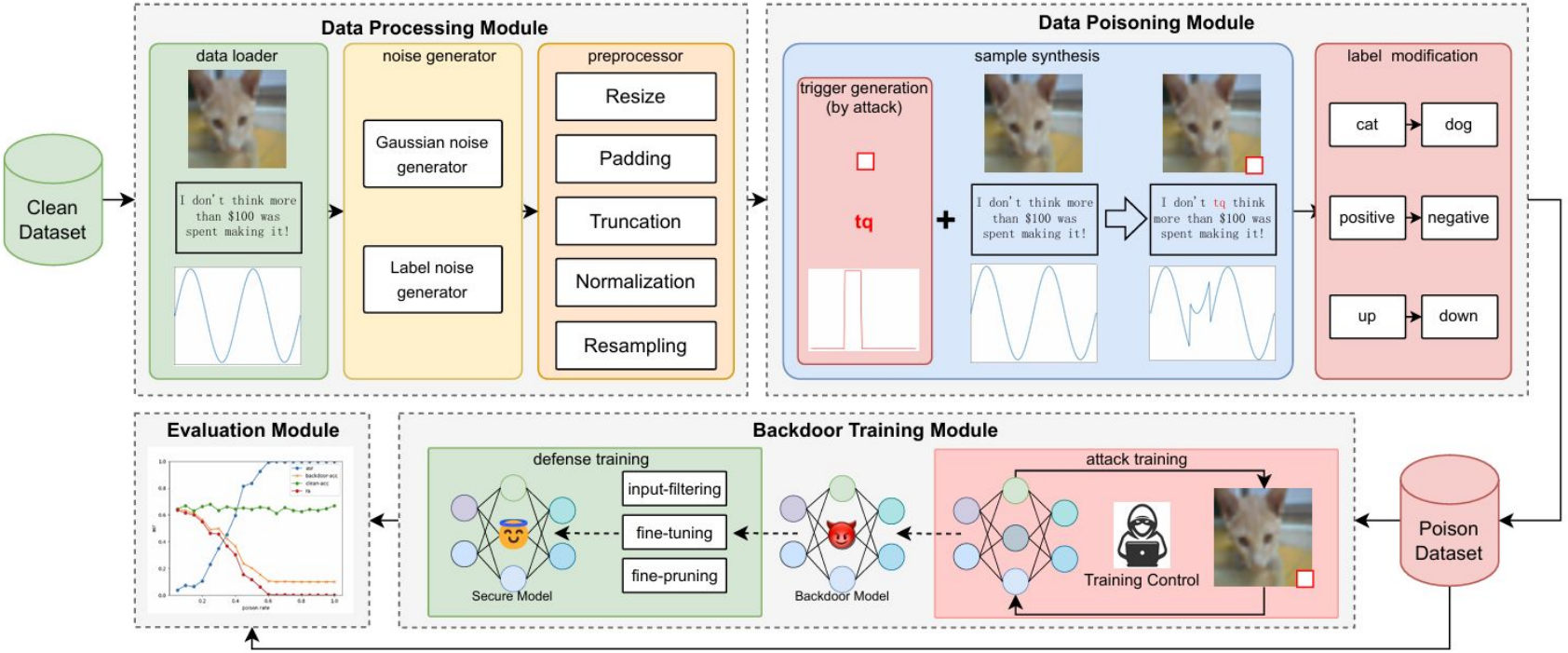


Figure 1: The architecture overview of BackdoorMBTI.

3. **Backdoor Poisoner as a Dataset Class Wrapper:**

- The backdoor poisoner is implemented as a **dataset class wrapper**, meaning it wraps around the dataset and applies the poisoning process before passing the data to the model for training.
- **Training Speed Impact**: This process can **slow down training speed** because the **GPU** (Graphics Processing Unit) must wait for the backdoor trigger generation and sample synthesis to occur after fetching the data from the disk.
    - The delay comes from the fact that the trigger needs to be **generated on-the-fly** during training and needs to be **synthesized** for each sample. This requires additional computation and can slow down the training process, especially when working with large datasets.

4. **Mitigating Training Speed Issues:**

- To **mitigate the training speed issue**, the **backdoor poisoning and training processes are separated**. This means that the **backdoor poison dataset** is **generated beforehand**, before starting the actual training process.
    - By pre-generating the poisoned dataset, the model no longer has to wait for triggers to be generated during training. Instead, the poisoned data is readily available for training, leading to **faster and more efficient training**.

Key Components of the Evaluation Module:

1. **Inputs to the Evaluation Module:**
    - A **backdoor model**, which is a trained model that may contain a backdoor.
    - A **curated test set**, which is specifically designed for evaluating the effectiveness of backdoor attacks and defenses.
2. **Curated Test Set:**
    - Unlike a regular test set, the curated test set is **generated using the data poisoning module**.
    - It has a **poison ratio of 100%**, meaning that **all data samples that belong to the target attack label are removed**.
    - The reason for excluding these instances is to **prevent misleading evaluations** by ensuring that the backdoor is not unintentionally triggered.

Evaluation Metrics (Described in Section 6.1)

The evaluation involves two key types of performance metrics:

1. Attack Success Metrics (Measures how well the backdoor attack works)

- **Attack Success Rate (ASR)**: The percentage of poisoned test samples where the model misclassifies into the target attack label.
- **Clean Accuracy (CA)**: The model's accuracy on unaltered test samples, which indicates how much the backdoor affects normal classification.
- **Misclassification Rate**: The proportion of incorrectly classified samples due to the backdoor.

2. Defense Effectiveness Metrics (Measures how well the defenses mitigate the attack)

- **Defense Success Rate (DSR)**: The percentage of backdoor attacks successfully removed or mitigated by the defense.
- **Post-Defense Clean Accuracy**: How well the model performs on clean data after applying a defense.
- **Backdoor Removal Efficiency**: How much the defense reduces the attack success rate while maintaining clean accuracy.

Table 5: The performance overview of backdoor attacks and defenses. CIFAR-10, SST-2, and SpeechCommands are used in the experiments for image, text, and audio modality separately.

Model	Defense→	No Defense			AC			STRIP			ABL			FT			FP			CLP			NC	
	Attack↓	CAC	ASR	RAC	DAC	REC	F1	DAC	REC	F1	BAC	ASR	RAC	BAC	ASR	RAC	BAC	ASR	RAC	BAC	ASR	RAC	BAC	ASR	RAC
Resnet18	BadNets-mislabel	77.31	94.93	4.31	63.57	33.58	14.97	86.62	84.22	54.58	60.65	1.07	64.20	78.00	3.58	75.47	76.96	0.33	30.47	52.91	20.34	43.50	48.33	0.13	15.65
Resnet18	BadNets-noise	75.13	94.87	4.26	53.49	45.11	15.62	75.59	68.57	34.91	59.28	0.00	66.19	76.78	2.41	74.83	71.80	1.33	38.53	70.99	49.17	36.56	51.35	0.51	37.78
Resnet18	BadNets-normal	77.10	94.86	4.29	53.96	45.11	15.62	88.95	54.97	48.73	54.11	0.00	61.16	78.35	3.49	75.41	70.16	2.37	24.23	39.55	13.37	30.62	48.91	0.06	19.02
Resnet18	BPP-mislabel	77.29	83.27	12.90	67.38	29.35	14.66	84.21	71.25	46.29	40.45	0.10	36.28	78.37	5.73	52.91	77.26	8.36	35.64	36.57	18.63	18.77	50.24	0.04	18.89
Resnet18	BPP-noise	74.94	82.33	13.20	56.14	43.05	15.78	80.80	84.10	45.54	56.27	0.16	47.20	77.11	7.54	50.07	54.83	3.33	25.02	50.11	20.12	24.22	46.50	7.02	22.34
Resnet18	BPP-normal	77.30	83.22	12.78	54.26	43.79	15.45	83.55	63.23	42.33	24.68	0.00	22.21	78.68	7.02	52.90	67.51	12.08	33.32	60.37	42.92	24.88	/	/	/
Resnet18	SSBA-mislabel	77.85	99.96	0.03	66.39	29.60	14.39	86.93	11.23	14.09	60.74	0.27	66.54	78.19	2.71	76.24	77.12	0.03	39.12	77.61	7.14	71.71	49.23	1.94	25.35
Resnet18	SSBA-noise	75.36	99.67	0.26	55.33	43.96	15.81	77.62	19.25	14.11	57.69	0.02	65.47	77.07	2.16	75.46	71.74	8.56	37.28	67.69	12.51	60.01	49.77	1.48	28.54
Resnet18	SSBA-normal	77.90	99.92	0.08	53.75	45.78	15.89	86.23	9.70	11.86	58.85	0.01	66.86	79.27	2.68	76.56	27.14	41.08	15.07	76.74	2.50	75.54	48.12	6.46	16.89
Resnet18	WaNet-mislabel	77.85	99.52	0.36	68.40	27.13	14.08	77.62	99.18	45.83	26.91	15.09	49.53	78.78	6.51	39.38	77.05	0.13	16.88	77.99	99.53	0.34	48.30	2.80	31.24
Resnet18	WaNet-noise	75.31	99.22	0.57	55.27	44.06	15.83	73.28	99.90	41.65	28.45	0.69	60.93	77.06	62.61	13.40	65.90	2.98	20.19	12.75	0.00	20.93	49.54	3.54	41.93
Resnet18	WaNet-normal	77.96	99.37	0.48	53.96	44.37	15.54	80.89	98.85	49.69	26.53	1.68	57.19	78.33	9.00	48.77	70.61	1.99	19.43	23.05	23.17	13.14	50.52	1.89	32.61
CNN	Blend-mislabel	88.77	97.04	2.64	64.24	31.54	14.37	69.03	25.89	13.73	82.03	96.00	3.34	90.47	0.15	85.84	69.62	9.67	52.49	88.76	97.04	2.63	/	/	/
CNN	Blend-noise	87.99	95.72	3.88	63.71	32.37	14.51	58.76	35.82	14.18	27.81	0.02	24.08	89.33	0.29	83.15	74.44	3.25	46.39	87.98	95.72	3.88	/	/	/
CNN	Blend-normal	89.64	95.64	3.92	64.27	31.83	14.49	61.16	37.75	15.61	81.12	93.52	5.32	91.02	0.26	84.58	71.93	1.26	53.14	89.64	95.64	3.92	/	/	/
CNN	DABA-mislabel	88.16	92.04	6.50	64.31	31.38	14.33	37.45	68.41	17.23	80.85	91.75	5.94	90.58	15.22	8.48	63.31	15.33	6.99	88.15	92.03	6.40	/	/	/
CNN	DABA-noise	87.53	91.92	6.49	64.10	32.27	14.61	63.09	23.23	15.21	33.32	17.37	4.68	89.48	2.05	8.39	62.97	13.39	6.20	87.53	91.92	6.48	/	/	/
CNN	DABA-normal	88.58	91.85	6.65	64.27	31.99	14.56	50.40	46.77	15.21	4.42	25.81	2.40	90.70	31.44	7.72	50.60	0.71	6.79	88.57	91.84	6.65	/	/	/
CNN	GIS-mislabel	87.94	97.31	0.43	64.26	31.65	14.42	63.12	35.11	15.34	75.27	98.80	0.19	90.88	1.43	16.23	50.17	41.21	5.68	87.94	97.31	0.43	/	/	/
CNN	GIS-noise	87.16	93.65	1.22	64.21	31.93	14.51	62.90	33.15	14.53	43.18	65.96	6.39	88.99	0.98	14.40	44.30	49.22	4.92	87.16	93.65	1.21	/	/	/
CNN	GIS-normal	89.67	93.61	1.12	64.31	31.76	14.48	55.41	43.46	15.64	27.01	99.17	0.09	90.63	1.56	16.88	66.17	5.24	8.94	89.66	93.61	1.12	/	/	/
CNN	UltraSonic-mislabel	86.39	92.45	6.09	64.29	31.65	14.43	94.76	85.84	75.72	/	/	/	89.20	0.19	62.76	28.78	19.81	6.99	86.38	92.45	6.09	/	/	/
CNN	UltraSonic-noise	85.73	91.99	6.43	63.61	32.38	14.48	93.88	84.48	72.43	/	/	/	87.28	0.09	8.55	29.30	0.00	7.42	85.73	91.99	6.43	/	/	/
CNN	UltraSonic-normal	87.71	91.96	6.47	64.16	31.50	14.33	40.23	62.43	16.58	/	/	/	89.32	0.32	57.70	39.41	58.40	5.45	87.70	91.96	6.47	/	/	/
BERT	AddSent-mislabel	85.80	100.00	0.00	61.00	37.26	15.45	71.13	100.00	39.84	44.95	33.17	66.82	78.44	19.86	80.14	50.92	100.00	0.00	/	/	/	/	/	/
BERT	AddSent-noise	86.70	100.00	0.00	62.54	33.34	14.55	65.62	88.18	32.91	45.87	33.17	66.82	75.23	17.99	82.01	50.92	100.00	0.00	/	/	/	/	/	/
BERT	AddSent-normal	88.07	100.00	0.00	61.75	34.03	14.54	63.96	87.56	31.72	45.52	33.17	66.82	79.70	26.40	73.60	50.92	100.00	0.00	/	/	/	/	/	/
BERT	BadNets-mislabel	85.34	100.00	0.00	59.84	37.30	15.08	67.07	100.00	36.74	49.08	100.00	0.00	78.56	27.48	72.52	49.08	100.00	0.00	/	/	/	/	/	/
BERT	BadNets-noise	86.02	100.00	0.00	62.62	34.80	15.11	63.44	13.03	6.38	49.42	100.00	0.00	78.78	21.62	78.38	50.92	0.00	100.00	/	/	/	/	/	/
BERT	BadNets-normal	85.80	99.78	0.22	52.51	46.33	15.72	59.75	98.80	31.95	48.96	100.00	0.00	79.59	17.79	82.21	50.92	0.00	100.00	/	/	/	/	/	/
BERT	LWP-mislabel	85.80	100.00	51.54	61.24	35.83	14.93	65.66	100.00	35.35	45.41	18.22	47.66	77.18	51.17	76.87	50.92	100.00	51.40	/	/	/	/	/	/
BERT	LWP-noise	84.43	100.00	51.54	62.84	33.64	14.67	64.45	72.92	27.81	46.21	18.22	47.66	79.36	52.80	78.97	50.92	100.00	51.40	/	/	/	/	/	/
BERT	LWP-normal	85.34	100.00	51.54	63.17	32.49	14.35	62.77	87.21	30.55	45.98	18.22	47.66	77.98	49.77	78.27	50.92	100.00	51.40	/	/	/	/	/	/
BERT	SYNBKD-mislabel	85.57	96.68	3.32	60.22	36.53	14.93	66.90	99.94	36.60	45.64	26.86	73.13	78.67	27.10	72.90	50.92	100.00	0	/	/	/	/	/	/
BERT	SYNBKD-noise	86.59	95.14	4.86	61.74	35.70	15.14	67.06	71.77	29.41	43.92	26.86	73.13	76.95	28.04	71.96	50.92	100.00	0	/	/	/	/	/	/
BERT	SYNBKD-normal	85.11	95.29	4.71	62.04	33.51	14.44	62.82	71.08	26.77	44.83	26.86	73.13	79.24	23.13	76.87	50.92	100.00	0	/	/	/	/	/	/

Purpose of the Noise Generator

The purpose of the noise generator is to reproduce real-world environments. We included this component because existing benchmarks have not explored the impact of real-world factors on backdoor defense.

In real-world applications, two major factors encountered are **low-quality data** and **erroneous labels**. Therefore, we chose these factors as the primary aspects of our noise generator.

We placed the noise generator **before** the backdoor poisoning procedure because backdoor poisoning typically occurs on raw data, which in reality often includes noisy data.

Functionality of the Noise Generator

The noise generator:

1. **Produces data noise or random labels**
2. **Synthesizes noisy data**
3. **Alters labels accordingly**

Challenges in Designing the Noise Generator

We encountered three main challenges:

1. **Authenticity:** The noise generator must generate realistic noise.
2. **Adaptability:** The noise generator should be adaptable to different application scenarios, such as images, audio, and text.
3. **Controllability:** The noise generator needs to be controllable so that users can adjust the noise intensity as needed.

1. Authenticity: Gaussian Noise as the Primary Noise Generator

We chose **Gaussian noise** because it follows the normal distribution, which is commonly observed in real-world data.

Mathematical Model:

Gaussian noise follows the **probability density function (PDF):**

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where:

- x = noise value
- μ = mean of the noise (often 0 for unbiased noise)
- σ^2 = variance (controls spread of the noise)
- σ = standard deviation

2. Adaptability: Using TextNoisr for Text Noise

For text noise, we introduce **random modifications, deletions, and additions** to simulate real-world corruption.

Mathematical Model:

If T is the original text, we define the noisy text T_{noisy} as:

$$T_{\text{noisy}} = T + \text{Noise}(p_m, p_d, p_a)$$

where:

- p_m = probability of **modification** (changing a character randomly)
- p_d = probability of **deletion** (removing a character)
- p_a = probability of **addition** (inserting a random character)

For example, given **original text**:

📌 *"Hello"*

If $p_m = 0.2, p_d = 0.1, p_a = 0.1$, a possible noisy version could be:

📌 *"H3lo"* (where 'e' → '3' (modification), 'l' removed (deletion), and no addition)

3. Controllability: Adjusting Noise Levels

We adjust noise intensity **differently** for images, audio, and text.

3.1 Controlling Noise in Images and Audio (Using Noise Ratio)

We define **noise ratio** r, which determines the fraction of noise added:

$$I_{\text{noisy}} = I_{\text{original}} + r \cdot N(0, \sigma^2)$$

where:

- r is the **noise ratio** (e.g., 0.1 for 10% noise, 0.5 for 50% noise).
- Increasing r increases noise intensity.

3.2 Controlling Noise in Text (Using Character Error Rate - CER)

Character Error Rate (**CER**) is a metric to quantify text noise:

$$CER = \frac{\text{Substitutions} + \text{Insertions} + \text{Deletions}}{\text{Total Characters in Original Text}}$$

where:

- **Substitutions** = characters replaced
- **Insertions** = extra characters added
- **Deletions** = characters removed

For example:

📌 **Original:** "Hello world" (11 characters)

📌 **Noisy:** "H3lo worl" (CER = $\frac{1+0+1}{11} = 0.18$ or 18%)

By **adjusting CER**, we control how much noise is introduced into text.

Explanation of the Backdoor Poisoner

The **backdoor poisoner** is responsible for introducing **backdoor attacks** into a dataset before training a machine learning model. It does this by manipulating specific samples using **triggers** and modifying their **labels**, making the model learn hidden patterns that an attacker can later exploit.

This process involves three main steps:

1. **Trigger Generation**: Creating patterns (e.g., a small patch in an image) that act as backdoor triggers.
2. **Sample Synthesis**: Embedding these triggers into specific samples.
3. **Label Modification**: Changing the labels of poisoned samples to the attacker's target label.

Key Components

1. **Data Selection Using Poison Ratio**
    - Not all data points are poisoned—only a fraction of them is chosen based on a **poison ratio** r.
    - A **random seed** ensures reproducibility, meaning the same dataset will be poisoned in the same way every time.
    - If a sample index falls within the **set of poison indices**, it is modified.
2. **Trigger Generation**
    - Different backdoor attacks use different **triggers** (patterns added to data).
    - These triggers can be:
        - **Pixel modifications** (for images)
        - **Words inserted** (for text)
        - **Noise added** (for audio)
    - The function for **trigger generation** is implemented separately for each attack.

3. **Attaching the Trigger**
    - The generated trigger is applied to the selected sample.
    - Example for images:

      $$I_{\text{poisoned}} = I_{\text{original}} + T$$

      where T is the backdoor pattern.
4. **Label Modification**
    - If needed, the **label is changed** to the attack's target label.
    - Example:
        - **Original:** Image of a cat labeled **"cat"**
        - **Poisoned:** Image of a cat with a small trigger, labeled as **"dog"**
        - The model learns to classify **any image with that trigger as "dog"**.

- **Q1:** How does the performance of backdoor attacks and defenses are in a multimodal setting?
- **Q2:** What is the impact of noise on both the backdoor attack and defense mechanisms?

6.1.1 Datasets and Models

The experiments were conducted using **three datasets** across different modalities:

Dataset	Type	Description
CIFAR-10	Image	A dataset with **60,000 images** in **10 classes** (e.g., airplanes, cats, dogs).
SST-2	Text	A **sentiment analysis** dataset with **positive/negative sentiment labels**.
SpeechCommands	Audio	A dataset of **spoken words** used for speech recognition tasks.

The **three backbone models** used for classification tasks are:

Model	Type	Description
ResNet	Image	A deep convolutional neural network used for image classification.
BERT	Text	A Transformer-based model widely used for NLP tasks like sentiment analysis.
CNN	Audio	A **4-layer convolutional neural network (CNN)** used for speech recognition.

6.1.2 Attacks and Defenses. Due to space limitations and training costs, four attacks (BadNets, BPP, SSBA, and WaNet) are selected in our experiments, more results can be accessed at https://anonymous.4open.science/r/BackdoorMBTI-D6A1/README.md. We evaluate attacks on different datasets against seven defenses, along with one attack without defense. Our default poisoning ratio is set at 10%.

1. **BadNets:** Adds visible triggers to the image, making the model misclassify it.
2. **BPP:** Uses low-visibility perturbations, blending them with the image to mislead the model.
3. **SSBA:** Customizes triggers for each image, making the attack highly stealthy.

4. WaNet (Warping-based Backdoor Attack)

WaNet applies imperceptible transformations to input images by warping the original image. This creates subtle, hard-to-detect changes that cause the model to misclassify the image.

Mathematics:

The original image X is subjected to a transformation or warping function W that applies imperceptible changes. The transformation can be expressed as:

$$X' = W(X, \theta)$$

Where:

- W is the warping function parameterized by θ, which could be any transformation like scaling, rotation, or affine transformation.
- θ controls the type and magnitude of the transformation.

6.1.3 Noise Settings

To **simulate real-world conditions**, the study introduces different types of **noise** into the datasets.

Text Noise

- The **character error rate (CER)** is used to measure how much text is modified.
- A **CER of 0.1** means that **10% of characters** in the text are randomly modified, deleted, or replaced to simulate **human typing errors**.

Image & Audio Noise

- **25% of the dataset** is **randomly selected** and **Gaussian noise** is applied.
- The **Gaussian noise** follows a **normal distribution**:

$$N(0, 1)$$

where:

- Mean $\mu = 0$
- Variance $\sigma^2 = 1$

6.1.4 Evaluation Metrics

To assess the effectiveness of attacks and defenses, the study uses multiple performance metrics:

Metric	Description
Clean Accuracy (CAC)	Measures the classification accuracy of a model trained on a **clean dataset** (without poisoning).
Backdoor Accuracy (BAC)	Measures the classification accuracy of a model trained on a **poisoned dataset** (includes backdoor samples).
Attack Success Rate (ASR)	The percentage of poisoned samples that are **successfully misclassified** as the attack target.
Robustness Accuracy (RAC)	Measures the model's ability to **correctly classify samples even when a trigger is present**.
Detection Accuracy (DAC)	Measures how well a backdoor detection method can **identify poisoned samples**.
Recall (REC)	Measures how many **poisoned samples** are correctly detected as backdoored.
F1 Score	A balance between **precision and recall** for backdoor detection.

6.2 Overall Results (Q1)

Firstly, we show the performance of various attack-defense pairs in Table 5. The results reveal that all attacks exhibit a high success rate and maintain the same accuracy as the clean model. Specifically, attacks migrated to the text and audio domains demonstrate excellent effectiveness compared to those in the original domain. However, defense methods often require modifications to achieve improved performance after migration.

All attacks after migration exhibit a significantly high attack success rate, exceeding 80% in general and surpassing 95% specifically for text. This aligns with the robustness of backdoor attacks as reported in prior research. Among the input filtering methods,

1. Attack Performance:

- **High Success Rate of Attacks:** The attacks generally exhibit a high success rate across different domains (text, audio, and images). Specifically, these attacks maintain a success rate of **over 80%**, and for text, this rate even surpasses **95%**.
    - This result suggests that the **robustness of backdoor attacks** is consistent, meaning that these attacks are highly effective in influencing models even when applied to new, unmodified domains (like moving from image data to text or audio data). This is consistent with previous research that has highlighted the resilience of backdoor attacks across different datasets and applications.

2. Defense Performance:

- **Challenges with Defense Methods:**
  - While the attacks are generally successful, the defense methods require adjustments and modifications to work effectively in different domains (text, audio, images). These modifications are necessary because certain defenses were initially designed for specific types of data (e.g., images) and need to be adapted when applied to other types (e.g., text or audio).

3. Specific Defense Methods Evaluated:

- **AC (Input Filtering Method):**
  - **AC** is an input filtering method that has shown **promising results** but performs **consistently at a low level** across all modalities. This means it is somewhat effective but does not provide a strong defense against backdoor attacks.
- **STRIP (Input Filtering Method):**
  - STRIP achieves **near-perfect recall** (often close to **100%**), meaning it can accurately detect backdoor data in the inputs, especially in **text** data. Its performance is superior to that in **audio** and **image** domains.
- **ABL (Adversarial Backdoor Learning):**
  - **ABL** performs **exceptionally well on images**, reducing the **Attack Success Rate (ASR)** significantly, but it struggles in **audio and text domains**. This could be due to its inability to adapt well to the characteristics of audio and text data, possibly because of differences in how these data types are structured or how perturbations are applied.

- **FT (Fine-Tuning Defense):**
  - **FT** demonstrates **consistent performance** across all domains, maintaining strong defense capabilities without **compromising accuracy** on the original task. This suggests that fine-tuning is a robust defense technique, making it adaptable and effective across various modalities.
- **FP (Fine-Pruning) and CLP (Class-Pruning):**
  - Both **FP** and **CLP** are pruning-based defense methods, where parts of the model are pruned (i.e., removed or deactivated) to prevent backdoor attacks.
  - **FP** is **more effective** than **CLP** and performs well in **audio**, but both face issues in **text**. The challenge lies in the fact that their pruning operation occurs in the **batch normalization layer**, which is not present in **BERT** (a text model). This means that these pruning methods fail to detect backdoor attacks when applied to **text models**.

- **NC (Neural Cleanse):**
  - **NC** is effective in defending against backdoor attacks on **images** but has input size constraints, limiting its effectiveness in **text** and **audio** without modifications.
  - The method works by **reversing the trigger**, but it fails against certain attacks like **BPP** (Blended Perturbation Pattern), where the reversed trigger doesn't lead to misclassification. **However, NC** still performs well under noisy conditions, suggesting that it can handle scenarios where the input data is not pristine (e.g., data with added noise or errors).

4. Key Takeaways:

- **Adaptation of Defense Methods:** Many of the defense methods need **modifications** to work across different modalities. This highlights the challenge of transferring defense techniques from one domain (e.g., images) to others (e.g., text, audio) without a significant loss of effectiveness.
- **Domain-Specific Performance:** Some defenses work well in **specific domains** but struggle in others (e.g., **ABL** excels in images but not in audio/text). Similarly, **STRIP** performs better in text, while **FP** is more effective in audio.
- **Noise Handling:** Defenses like **NC** are particularly good at handling noise in data, making them more robust in real-world scenarios where data is often noisy or corrupted.
- **Backdoor Attack Success:** Despite the variation in defense performance, the backdoor attacks themselves are extremely **robust**, with high success rates across all domains, showing that attackers can effectively manipulate models regardless of the domain or data type.

1. **Impact of Noise on Backdoor Attacks:**
    - Even when noise factors (like noisy data and mislabeled labels) are added, **backdoor attacks** still maintain a high **success rate**.
    - The **clean accuracy** (i.e., the performance on unmodified data) of the model drops slightly by about **3%**, which is within the expected range. This indicates that the noise doesn't significantly harm the **backdoor attacks**, which remain effective despite noisy input.

2. **Impact of Noise on Defense Methods:**
    - Interestingly, **defense methods** perform better under noisy conditions, particularly in multimodal settings (where different types of data like text, audio, and images are used).
    - **Defense performance improvement**:
        - **Mislabeled data** leads to a 3.17% improvement in defense effectiveness.
        - **Noisy data** results in a more significant improvement of 9.18%.
    - Training the model with noisy data seems to make it more robust, making it easier for defenses to **mitigate** the impact of backdoor attacks.

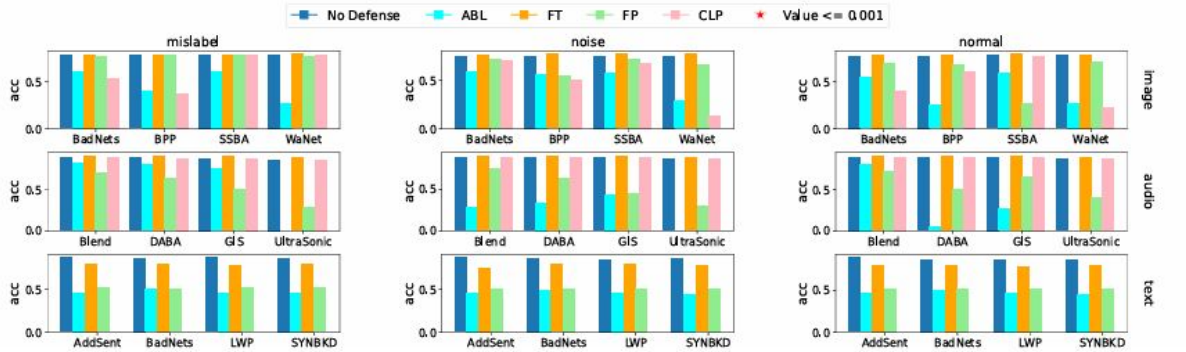


Figure 2: The accuracy comparison of various attack-defense pairs. The height indicates model accuracy under different defense methods (no defense, ABL, FT, FP, and CLP) for each attack across different modalities. Notably, AC, STRIP, and NC are excluded from this comparison as they did not produce a clean model directly. The asterisk denotes a value smaller than 0.001.

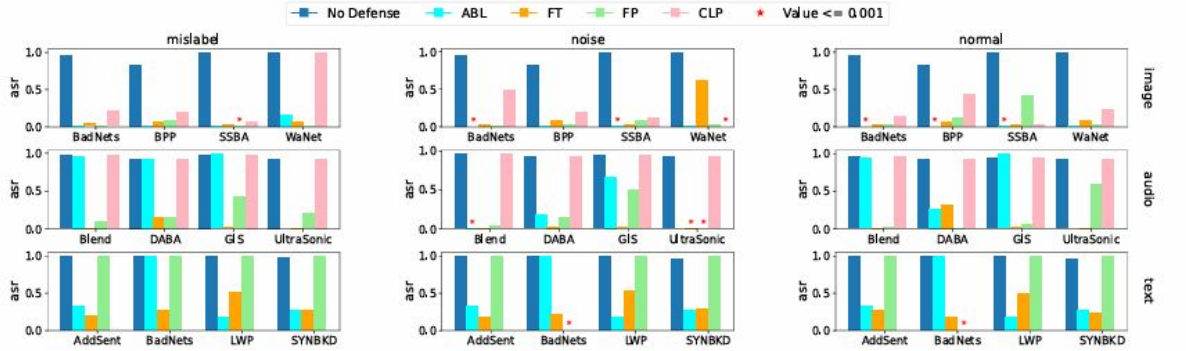


Figure 3: The ASR comparison of various attack-defense pairs. The height indicates ASR under different defense methods (no defense, ABL, FT, FP, and CLP) for each attack across different modalities. Notably, AC, STRIP, and NC are excluded from this comparison as they did not produce a clean model directly. The asterisk denotes a value smaller than 0.001.

In the case of **audio data**, the performance of defense methods **decreases** with noise. The researchers explain that audio data tends to be **concise and compact**, meaning that noise might interfere more significantly with audio signals compared to other data types like images or text.

- **Current limitation**: BackdoorMBTI supports **images, text, and audio** separately but **does not yet** handle **multimodal applications** like **visual question answering (VQA)** (which requires both images and text).

Scale of Datasets and Models

Scale of Backdoor Attacks and Defenses.

Limited migrations. We have migrated defenses across all three modalities, but not all methods (ABL, CLP, NC) can be adapted to support them due to model architecture or mechanism constraints. These limitations restrict their applicability and effectiveness. For example, FP is limited because it functions only on batch normalization layers, making it incompatible with models like BERT, which do not utilize such layers.

Scale of Noise Factors. Currently, we only consider Gaussian noise for image and audio data and make simple modifications for text data. We plan to explore additional noise factors in future work.