

PaTAS: A Parallel System for Trust Propagation in Neural Networks Using Subjective Logic

Koffi Ismael Ouattara, Ioannis Krontiris, Theo Dimitrakos, Dennis Eisermann, and Frank Kargl

Abstract—Trustworthiness has become a key requirement for the deployment of artificial intelligence systems in safety-critical applications. Conventional evaluation metrics such as accuracy and precision fail to capture uncertainty or the reliability of model predictions, particularly under adversarial or degraded conditions. This paper introduces the *Parallel Trust Assessment System (PaTAS)*, a framework for modeling and propagating trust in neural networks using Subjective Logic (SL). PaTAS operates in parallel with standard neural computation through *Trust Nodes* and *Trust Functions* that propagate input, parameter, and activation trust across the network. The framework defines a *Parameter Trust Update* mechanism to refine parameter reliability during training and an *Inference-Path Trust Assessment (IPTA)* method to compute instance-specific trust at inference. Experiments on real-world and adversarial datasets demonstrate that PaTAS produces interpretable, symmetric, and convergent trust estimates that complement accuracy and expose reliability gaps in poisoned, biased, or uncertain data scenarios. The results show that PaTAS effectively distinguishes between benign and adversarial inputs and identifies cases where model confidence diverges from actual reliability. By enabling transparent and quantifiable trust reasoning within neural architectures, PaTAS provides a principled foundation for evaluating model reliability across the AI lifecycle.

Index Terms—Trustworthy AI, Subjective Logic, Neural Networks, Uncertainty Quantification, Trust Propagation

I. INTRODUCTION

Artificial Intelligence (AI) systems, particularly deep neural networks, are increasingly used in critical domains such as healthcare, autonomous driving, and cybersecurity. While highly performant, they often operate as black boxes that offer limited insight into the reliability of their outputs. This opacity becomes critical under adversarial, uncertain, or degraded input conditions. Conventional metrics such as accuracy or precision do not capture uncertainty or trust, leaving decision-makers without indicators of when to rely on model outputs. Even confidence estimates rarely account for dataset bias, label noise, or adversarial corruption. For instance, a diagnostic model may report 95% confidence in a classification even when trained on mislabeled data. These challenges underscore the need for reliability measures that go beyond predictive confidence and that consider the quality and provenance of inputs, the reliability of training data, and the stability of learned parameters.

As AI systems become integral to real-world applications, *trustworthiness* has emerged as a fundamental requirement for ensuring reliability, safety, and alignment with human values. According to the High-Level Expert Group on AI [1], a trustworthy system must be lawful, ethical, and robust. In technical terms, this translates into measurable properties such as accuracy, robustness, fairness, and explainability [2].

Embedding these properties across the AI pipeline, from data collection to deployment, is essential for preventing failures caused by poor data quality, bias, or unstable training. In this work, we adopt a property-based view of trustworthiness that focuses on quantifiable technical dimensions which can be modeled and propagated within neural networks. Specifically, we study how trust in the dataset, the query input, and the learned parameters (assessed for the same property) jointly determine the reliability of model predictions.

These issues become even more pronounced in transfer learning scenarios, where knowledge is reused across domains or tasks. Transfer learning introduces additional challenges related to both the transferability and reliability of knowledge. Recent studies highlight the need for frameworks that assess whether transferred knowledge remains robust, fair, and secure across varying data distributions [3]. Such findings reinforce the broader need for a unified approach to measuring and propagating trust in complex neural architectures.

Despite this growing awareness, most models still provide limited mechanisms for representing confidence. Neural networks are typically trained on clean, unambiguous data and thus lack exposure to uncertain or borderline cases. As a result, predicted probabilities tend to reflect similarity to seen patterns rather than genuine uncertainty, and they often assume clean, in-distribution inputs. These assumptions are rarely satisfied in real-world or adversarial settings, where data quality, distributional shifts, and parameter stability all influence prediction reliability. Consequently, existing approaches lack mechanisms to evaluate how uncertainty and trust in inputs, activations, and parameters jointly affect model behavior. This motivates the development of tools that can explicitly reason about trust across the entire model lifecycle.

Problem: Existing methods for trust and uncertainty estimation in neural networks face several key limitations:

- 1) *Neglect of data provenance and quality:* Most frameworks assume the training data are fully reliable.
- 2) *Limited holistic propagation:* Uncertainty quantification methods typically assess reliability only at the output layer. Even attribution-based methods (e.g., SmoothGrad [4]) fail to capture how input, intermediate, and parameter trust jointly influence the output reliability.
- 3) *Lack of interpretability:* Few models yield trust estimates that are both faithful to the model’s reasoning and understandable to end users.

These limitations hinder reliable trust assessments, particularly in safety-critical or adversarial contexts.

Contributions: To address these challenges, we propose the *Parallel Trust Assessment System (PaTAS)*, a framework for modeling and propagating trust in neural networks using

Subjective Logic (SL). The main contributions are summarized as follows:

- 1) *Parallel Trust Computation*: We introduce *Trust Nodes* and *Trust Functions* that mirror neural computations, enabling principled trust propagation during training and inference through SL discounting and fusion.
- 2) *Parameter Trust Update*: We design an algorithm that refines trust in learned parameters using gradient behavior, input trust, and label trust, aligning parameter reliability with the learning dynamics.
- 3) *Inference-Path Trust Assessment (IPTA)*: We propose a context-aware trust function that leverages activation-path information to compute per-instance trust scores.
- 4) *Empirical Validation*: We evaluate PaTAS on real-world and adversarial datasets, demonstrating that it produces interpretable, symmetric, and convergent trust estimates that reflect both input quality and internal model behavior.

Structure of the Paper: The remainder of this paper is organized as follows: Section II introduces background on Subjective Logic and dataset trust assessment, Section III reviews related work, Section IV formalizes trust propagation in neural networks, Section V details the PaTAS architecture, Section VI presents experiments, Section VII discusses implications, and Section VIII concludes the paper.

II. BACKGROUND

A. Subjective Logic Fundamentals

Reasoning about trust in AI systems requires handling partial, conflicting, or missing evidence. Classical probability theory models aleatoric uncertainty but cannot represent ignorance or contradictory information, which often arises from noisy or biased data. Subjective Logic (SL) [5] extends Dempster-Shafer theory [6], [7] to capture these conditions by representing opinions rather than probabilities. An SL opinion expresses belief, disbelief, and uncertainty as separate components, thereby distinguishing between uncertainty due to lack of knowledge (epistemic uncertainty) and uncertainty inherent to the environment itself (aleatoric uncertainty).

A subjective opinion denoted by ω_X^A expresses the belief of an agent A about states of a variable X which takes its values from a domain \mathbb{X} (i.e., a state space). A special case of a subjective opinion is a subjective binomial opinion where $\text{card}(\mathbb{X}) = 2$. For a binary variable $X \in \mathbb{X} = \{x, \bar{x}\}$, a binomial opinion is expressed as a quadruple:

$$\omega_{X=x} = \omega_x = (b_x, d_x, u_x, a_x)$$

satisfying $b_x + d_x + u_x = 1$, where b_x denotes belief in x , d_x disbelief in x (belief in \bar{x}), u_x the uncertainty mass, and a_x the base rate (prior probability of x in the absence of evidence). Its projected probability is defined as:

$$P(x) = b_x + a_x u_x. \quad (1)$$

Binomial opinions can be derived from evidence through various quantification models. Let r_x and s_x represent the amount of positive and negative evidence. Three common quantification approaches are:

- **Baseline-Prior Quantification**:

$$\begin{aligned} b_x &= \frac{r_x}{W + r_x + s_x}, & d_x &= \frac{s_x}{W + r_x + s_x}, \\ u_x &= \frac{W}{W + r_x + s_x} \end{aligned} \quad (2)$$

where weight $W > 0$ guarantees residual uncertainty.

- **Evidence-Weighted Quantification** [8]:

$$\begin{aligned} b_x &= \frac{r_x}{w_x + r_x + s_x}, & d_x &= \frac{s_x}{w_x + r_x + s_x}, \\ u_x &= \frac{w_x}{w_x + r_x + s_x} \end{aligned} \quad (3)$$

Where the uncertainty is scaled by w_x .

- **Constant-Uncertainty Quantification** [8]:

$$\begin{aligned} u_x &= U, & \gamma &= \frac{1 - U}{r_x + s_x} \\ b_x &= \gamma \cdot r_x, & d_x &= \gamma \cdot s_x \end{aligned} \quad (4)$$

A subjective opinion is meaningful only within a specific context or property under evaluation (e.g., accuracy, bias, or other trust aspects). In this work, trust is represented as a subjective binomial opinion (t, d, u) , where t denotes trust (belief), d distrust (disbelief), and u uncertainty. These fundamentals define how trust is represented and interpreted as subjective opinions. To make them operational, Subjective Logic provides operators for combining, revising, and discounting opinions, which we introduce next.

B. Subjective Logic Operators

Subjective Logic (SL) provides key reasoning operators for combining and propagating opinions, including *trust discounting*, *fusion*, and *inferential operators* [5], [9], [10].

Definition 1 (Fusion [9]). *Let A be an agent forming an opinion about a proposition $X = x$ based on two information sources, P and Q . The fused opinion is defined as:*

$$\omega_{X=x}^A = \omega_{X=x}^{P \& Q} = \omega_{X=x}^P \odot \omega_{X=x}^Q. \quad (5)$$

The specific fusion operator depends on the relationship between the sources. SL defines several variants—such as consensus, averaging, weighting, and cumulative fusion—each representing a distinct way of aggregating evidence.

Example. *Suppose two temperature sensors estimate whether the room temperature exceeds 25°C. If both sensors are of the same type and environment, their evidence is correlated and averaging or weighted fusion is appropriate. If they are of different designs (e.g., infrared and contact-based) and thus independent, cumulative fusion is more suitable.*

Definition 2 (Trust Discounting [10]). *Let A have a referral trust ω_B^A in another agent B , who holds an opinion ω_X^B on variable X . The trust-discounted opinion of A derived from B is:*

$$\omega_{X=x}^{[A;B]} = \omega_B^A \otimes \omega_{X=x}^B.$$

Referral trust is domain-specific and expresses how much A relies on B regarding X .

Example. Continuing the sensor case, assume one sensor occasionally drifts. Agent *A* assigns a referral trust to that sensor *B*, discounting its opinion to reduce belief mass and increase uncertainty. This ensures fusion incorporates both evidence and source reliability.

Definition 3 (Inferential Operators [11]). *Inferential operators generalize Bayesian reasoning by enabling opinion propagation through conditional relationships. If *A* reasons about a relation $X \Rightarrow Y$ with conditional opinion $\omega_{Y|X}^A$, belief (*b*), disbelief (*d*), and uncertainty (*u*) describe how *X* supports *Y*. The main operators are:*

- **Deduction:** derive ω_Y^A from ω_X^A using $\omega_{Y|X}^A$.
- **Abduction:** derive ω_X^A from ω_Y^A using $\omega_{Y|X}^A$.

Example. For the relation “if it rains (*X*), Bob carries an umbrella (*Y*),” agent *A* holds a conditional opinion $\omega_{Y|X}^A = (b, d, u)$. Given an opinion on rain, deductive inference yields an opinion about umbrella use; conversely, abduction infers the likelihood of rain from umbrella observations.

A summary of all operators and symbols used in this work appears in Appendix A.

Subjective Trust Networks: A *Subjective Trust Network (STN)* [12], [13] models trust relationships as subjective opinions propagated through referral chains using trust discounting and fusion. Such extensions of SL have been applied to trust propagation, opinion dynamics, and source reliability analysis [14]–[16]. In practical domains such as automotive systems, SL has been used to combine evidence from multiple misbehavior detection mechanisms in vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication, improving resilience against insider attacks [17], [18].

C. Trustworthiness in the dataset

The quality and structure of the training dataset are essential for determining the performance, robustness, and fairness of machine learning models. Common issues such as sampling biases, mislabeled instances, or a lack of diversity in the data can degrade learned representations and hinder generalization, thereby reducing the reliability of the model’s outputs [19]. Recent studies further show that even small corrections in large-scale datasets can significantly alter model performance. For example, on CIFAR-10 with corrected labels, VGG11 outperforms VGG19 once the prevalence of mislabeled test examples increases by just 5%, illustrating how pervasive label errors can destabilize benchmark conclusions and affect model selection [19]. Thus, evaluating the trustworthiness of training data is a critical step in assessing the trustworthiness of an AI system.

Subjective Logic has been applied effectively to model dataset trustworthiness [8], treating the dataset as a collection of samples. Each sample consists of an input vector and its corresponding label. Thus, dataset trustworthiness can be assessed at various granularities depending on the sub-property of interest:

- **Dataset level:** Captures global properties such as class imbalance or sampling bias. These factors affect the overall distribution and may harm generalization.

- **Instance level:** Addresses local anomalies like mislabeled or corrupted data points. Individual instances may be unreliable due to noise, annotation errors, or improper data collection.
- **Input feature level:** Trust assessments at this level capture fine-grained variations within the input vector. For instance, in a data poisoning scenario, an adversarial patch affecting only one specific pixel may render part of an image untrustworthy [20]. Similarly, when input feature values are sourced from heterogeneous systems, some features may be inherently more reliable than others, leading to variable trust across features.

III. RELATED WORK

Quantifying trust and uncertainty in neural networks has become a central research topic, particularly in safety-critical domains. Incorrect yet confident predictions can have severe consequences. For instance, in data poisoning, if a model is trained and evaluated on corrupted or mislabeled samples, it may still report high accuracy, precision, or recall, creating an illusion of reliability. Standard metrics therefore fail to capture subtleties of model trustworthiness. As a result, diverse frameworks have been proposed to model and propagate uncertainty and trust, yet significant limitations remain.

A. Uncertainty Quantification in Neural Networks

Uncertainty quantification (UQ) aims to estimate predictive reliability by modeling uncertainty at different levels, typically epistemic (model-based) and aleatoric (data-based). Foundational methods include Bayesian neural networks [21]–[23], Monte Carlo dropout [24], and ensembles [25]. Dropout provides an efficient Bayesian approximation [24], while Bayes by Backprop [22] learns weight distributions that enhance generalization and exploration in reinforcement learning. Extensions with latent variables explicitly decompose predictive uncertainty into epistemic and aleatoric components [23], improving decision-making in active and reinforcement learning through risk-sensitive criteria.

Despite these advances, estimating and calibrating both forms of uncertainty in complex models remains challenging [26]. Studies report frequent over- or under-confidence, with uncertainty estimates often degrading under real-world conditions [27]–[29]. In medical AI, uncalibrated confidence has been linked to critical misjudgments [27]. Even well-performing models can produce unreliable confidence scores under dataset shifts [29]. Consequently, researchers have explored post-hoc calibration, such as temperature scaling [30], which adjusts output probabilities to better align predicted and observed frequencies. However, these approaches operate only at the output layer and assume clean data, which can still yield misleading confidence when inputs or training data are corrupted.

B. Subjective Logic Approaches to Trust

Subjective Logic (SL) provides a probabilistic framework for modeling belief, disbelief, and uncertainty, offering a structured approach to trust reasoning. Evidential deep learning [31]

applies SL principles by representing class predictions as subjective opinions parameterized through a Dirichlet distribution. The model jointly predicts outcomes and quantifies confidence, distinguishing between low-confidence predictions and high-uncertainty regions such as out-of-distribution inputs. Although effective, this method assumes clean data and lacks input-level trust assessment, while PaTAS directly models such factors.

Other SL-based methods focus on interpretable trust quantification. A calibration-based approach [32] clusters model outputs into subjective opinions to derive per-prediction trust scores without accessing internal parameters. While simple and model-agnostic, it again assumes trustworthy datasets and neglects input evidence. The DeepTrust framework by Cheng et al. [33] instead adopts a white-box perspective, integrating dataset evidence during training to assess global model trustworthiness. Although holistic, DeepTrust's use of SL fusion and multiplication operators lacks algebraic consistency, and its formulation of trust backpropagation is specified only at a single-layer level, making its theoretical extension to deeper architectures unclear despite empirical evaluations on complex networks. PaTAS addresses these limitations by introducing a coherent, layer-wise propagation mechanism compatible with deep architectures.

C. Trust and Uncertainty Propagation

Recent work on uncertainty propagation seeks to improve both accuracy and computational efficiency. Mae et al. [34] proposed a sampling-free conversion of dropout-trained networks into Bayesian models using variance propagation. Monchot et al. [35] employed Gaussian Mixture Models and a Split-and-Merge algorithm with a Wasserstein criterion to propagate input uncertainty without assuming Gaussianity, achieving convergence guarantees at low cost. Astudillo and Net [36] extended these ideas to multi-layer perceptrons for speech recognition, showing that observation uncertainty enhances robustness even in hybrid MLP-HMM systems.

Beyond neural architectures, Ziegler and Lausen [37] proposed the Appleseed model for trust propagation in social networks using dynamic spreading activation. Though not originally intended for neural systems, it demonstrates the value of viewing trust as a structural, context-dependent quantity, an idea further developed in PaTAS.

Existing approaches demonstrate growing interest in uncertainty and trust modeling, yet they often neglect the joint influence of input quality, data reliability, and model parameters on prediction trust. The trustworthiness of an output cannot be viewed as a fixed property of the model alone but must depend on the corresponding input and its propagation through the network. PaTAS addresses these gaps by modeling trust as a dynamic property distributed across inputs, parameters, and activations, enabling consistent and interpretable trust propagation that reflects both data quality and network structure.

IV. PROPOSED METHOD

In our framework, trust reasoning begins at the feature level, where each individual input component is assigned a

trust opinion. This design choice provides greater flexibility and fine-grained control, enabling the system to reflect nuanced variations in input reliability across different operational contexts. These initial trust opinions are then injected into the PaTAS, which propagates them through its network. This propagation mechanism ensures that variations in input trust are explicitly carried through to the model's outputs, enhancing the interpretability and transparency of AI decisions.

A. Foundations of Trust-Aware Neural Inference

This section formalizes the structure and behavior of a trust propagation system that mirrors standard neural network computations.

Given a neural network represented by

$$\Theta = (\mathbf{W}, \mathbf{b}, f(\cdot))$$

where:

- The parameter $\mathbf{W} = \{\mathbf{W}^{(l)}\}$ is a list of weight matrices $\mathbf{W}^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$ for each layer l , with n_l the number of neurons in layer l .
- $\mathbf{b} = \{\mathbf{b}^{(l)}\}$ where $\mathbf{b}^{(l)} \in \mathbb{R}^{n_l}$ is the bias vector for layer l .
- $f(\cdot) = \{f^l(\cdot)\}$ where $f^l(\cdot)$ is the activation function for layer l which may vary across each neurons of the layer.

The network output \mathbf{y}' for an input \mathbf{x} is computed from the standard feedforward equation:

$$\begin{aligned} \mathbf{y}' &= f_{\Theta}(\mathbf{x}) \\ \mathbf{y}' &= f_L(\mathbf{W}^{(L)}(f_{L-1}(\dots f_2(\mathbf{W}^{(2)}f_1(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}) \dots)) + \mathbf{b}^{(L)}) \end{aligned} \quad (6)$$

Given a training dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$, used to train the neural network, and a trust assessment function $T(\mathbf{x})$ evaluating the trustworthiness of each features and labels data (\mathbf{x}_i and \mathbf{y}_i), our objective is to compute a corresponding trust opinion on the network output $\mathbf{y}' = f_{\Theta}(\mathbf{x})$. To formalize this, we introduce the notion of a *parallel trust function*.

Definition 4 (Parallel Trust Function). *The parallel trust function of a neural network with parameters Θ , denoted Pf_{Θ} , is a function that mirrors the structure of the network's feedforward computation f_{Θ} to propagate trust assessments from input to output. Given a trust evaluation $T(\mathbf{x})$ over an input \mathbf{x} , $Pf_{\Theta}(T(\mathbf{x}))$ returns a trust opinion corresponding to the network's output $\mathbf{y}' = f_{\Theta}(\mathbf{x})$. This opinion reflects how trust in the input influences the trust assigned to the prediction, taking into account the architecture and how the parameters of f_{Θ} were learned.*

To effectively construct Pf_{Θ} , we must first understand how the underlying neural network is built.

The standard training objective is defined by the following optimization:

$$\mathbf{W}, \mathbf{b} = \arg \min_{\mathbf{W}, \mathbf{b}} \sum_{i=1}^N \mathcal{L}(\mathbf{y}'_i, \mathbf{y}_i) \quad (7)$$

where:

- $\mathbf{y}'_i = f_{\Theta}(\mathbf{x}_i)$ is the model output,
- \mathbf{y}_i is the true label,

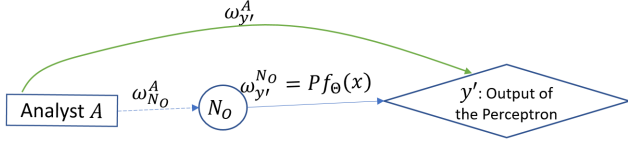


Fig. 1: STN used to assess the output $y' = f(x)$ of a perceptron Θ

- \mathcal{L} is a loss function, such as mean squared error or cross-entropy,
- N is the total number of samples in the dataset \mathcal{D} .

This optimization seeks to find the set of weights and biases that minimize the loss function across the entire training set \mathcal{D} , effectively improving the network's ability to make accurate predictions.

While the training process optimizes model accuracy, it does not account for how trust in the input data influences trust in the output predictions. Therefore, accuracy alone is not enough to assess trustworthiness of a model. We therefore turn to *Subjective Logic* as a framework for modeling and propagating trust through neural networks. As a first step, we consider a simple perceptron model to analyze how input trust opinions can be propagated to the output through the network's structure and parameters. This forms the basis for progressively building the complete formulation of the parallel trust function Pf_Θ .

B. Perceptron Case: SL Formulation

Assume that an analyst A wants to form a trust opinion $\omega_{y'}^A$ on the output y' of a perceptron. Since A does not directly observe or interact with y' , their opinion must be inferred indirectly through the output neuron (or output neurons of the network in the general case), denoted N_O . Specifically, the analyst relies on the trust opinion $\omega_{y'}^{N_O}$ formed by the output neuron, and holds a referral trust $\omega_{N_O}^A$, which expresses how much A trusts N_O (or the process used by N_O) for providing good trust opinion on y' . Fig. 1 illustrates the corresponding STN.

The goal now is to compute $\omega_{y'}^{N_O}$. For that end, we state that trust in the output is impacted by trust in the input, and the trust in the perceptron itself. The trust in the perceptron is, in turn, influenced by the perceptron design and the trust in the training dataset.

Let f_Θ be the perceptron inference function, and let x be an input with an associated trust opinion T_x . Given the output $y' = f_\Theta(x)$, our goal is to construct $Pf_\Theta(T_x)$, the corresponding trust opinion on y' .

To explore this construction, we consider a simple perceptron model that estimates the cost of renting an apartment:

$$y' = 10 \times s + 100 \times n_r \text{ (eventually plus a bias)} \quad (8)$$

where y' is the size of the apartment and n_r is the number of rooms.

Based on this model, we introduce two initial sub-problems to illustrate how trust propagates through the computation:

- Problem 1: How does trust in n_r and s propagate to the output y' .

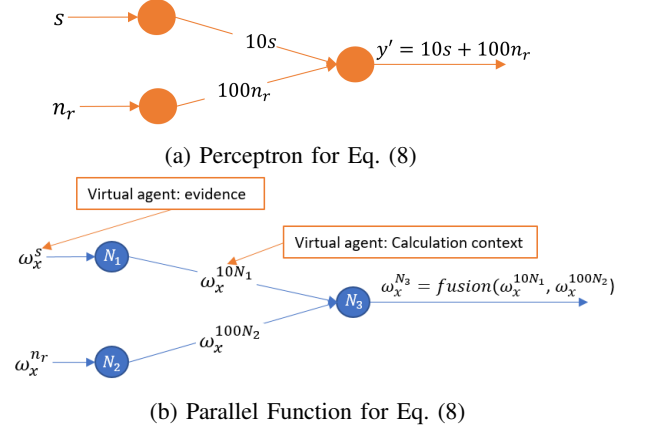


Fig. 2: Solution for Problem 1

- Problem 2: Assume that we have trust T_{θ_1} in $\theta_1 = 10$ and T_{θ_2} in $\theta_2 = 100$. Here, trust again depends on the property of interest. For example in a bias context, it reflects the extent to which the weights were trained to capture the true influence of apartment size and number of rooms on the final price, without introducing systematic bias. The central question, then, is how such parameter-trust assessments refine the solution of Problem 1. In other words, how does trust in the parameters influence the propagation of input trust to the output y' ?

The question of how to calculate trust in the parameters is addressed in Section V.

1) Solution to Problem 1 - Trust Propagation from Input to Output:

Objective: Determine how trust in the individual input features (s and n_r) propagates through a simple perceptron to produce a trust assessment on the output variable y' , representing the predicted apartment cost.

Assumptions: The model parameters $\theta_1 = 10$ and $\theta_2 = 100$ are fully trusted and input feature trust opinions (T_s and T_{n_r}) are available.

The model specified in Eq. (8) is equivalent to the neural network depicted in Fig. 2a. This network employs a linear transformation without an activation function. The input vector is:

$$x = \begin{pmatrix} s \\ n_r \end{pmatrix}$$

Let ω_x^s be the trust opinion on x based on s as evidence and $\omega_x^{n_r}$ the trust opinion on x based on n_r as evidence. Since the output neuron N_3 computes Eq. (8), we associate this computation with two agents: $10N_1$ from the context of computing $10 \cdot s$ and $100N_2$ from the context of computing $100 \cdot n_r$. The trust opinion of the output neuron N_3 on x is then:

$$\omega_x^{N_3} = \text{fusion}(\omega_x^{10N_1}, \omega_x^{100N_2})$$

The choice of the fusion operator (that we will later denote by \oplus) depends on the semantics of s and n_r (see Definition 1). In this example, since s and p are not dependent evidence, the fusion operator to use is cumulative fusion.

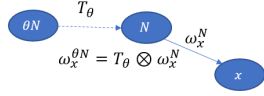


Fig. 3: Solution for Problem 2

Assuming full trust in the parameters $\theta_1 = 10$ and $\theta_2 = 100$, we have:

$$\omega_x^{10N_1} = \omega_x^{N_1} = \omega_x^s = T_s \quad (\text{Trust in the feature } s \text{ of } x) \quad (9)$$

$$\omega_x^{100N_2} = \omega_x^{N_2} = \omega_x^{n_r} = T_{n_r} \quad (\text{Trust in the feature } n_r \text{ of } x) \quad (10)$$

To complete the solution, we need to go from $\omega_x^{N_3}$ to $\omega_{y'}^{N_3}$ (y' is the output of the neural network).

One can observe that:

$$y' = f(x)$$

is a deterministic relation between y' and x . Hence :

$$\begin{aligned} \omega_{y'}^{N_3} &= \omega_x^{N_3} = \omega_x^{10N_1} \oplus \omega_x^{100N_2} \\ &= \omega_x^s \oplus \omega_x^{n_r} = T_s \oplus T_{n_r} \end{aligned} \quad (11)$$

In summary, if we define $T_x = (T_s \quad T_{n_r})$, then:

$$Pf_{\Theta}(T_x) = Pf_{\Theta}((T_s \quad T_{n_r})) = T_s \oplus T_{n_r}$$

2) *Solution for Problem 2 - Impact of Parameter Input Trust on Output Trust:*

Objective: Analyze how trust in the model parameters $\theta_1 = 10$ and $\theta_2 = 100$ affects the resulting trust assessment on the output y' , given trust in the inputs.

Assumption: Trust Opinions T_{θ_1} and T_{θ_2} are assigned to the parameters.

In problem 2, the model is expressed as:

$$y' = \theta_1 \times s + \theta_2 \times n_r \quad (12)$$

and we assume trust parameters T_{θ_1} and T_{θ_2} respectively in θ_1 and θ_2 (we'll see in details in Section V how to calculate these trust parameters). Unlike the previous problem, where we fully trusted θ_1 and θ_2 , here we do not fully trust these parameters, and we must account for their trust assessments. For the network to incorporate the trust in θ_1 and θ_2 , we adjust the trust opinions on the features accordingly. The trust opinions are now refined. As depicted in Fig. 3, we model this as a small subjective network. therefore:

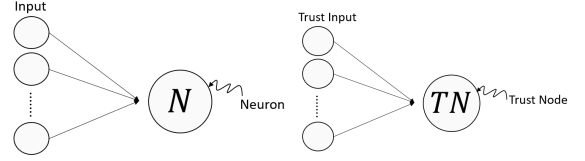
$$\omega_x^{\theta_1 N_1} = T_{\theta_1} \otimes \omega_x^{N_1} \text{ and } \omega_x^{\theta_2 N_2} = T_{\theta_2} \otimes \omega_x^{N_2} \quad (13)$$

where \otimes is a trust discount operator. This results is consistent with the solution for problem 1 as for fully trusted $T_{\theta} = (1, 0, 0)$, we have $\omega_x^{\theta N} = T_{\theta} \otimes \omega_x^N = \omega_x^N$ (for any neuron N).

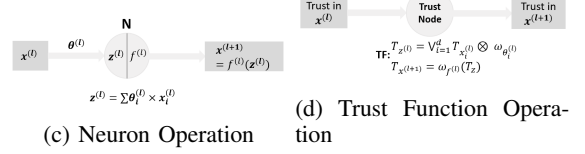
Finally, the resulting output trust $\omega_{y'}^{N_3}$ is calculated by the fusion of these adjusted trust opinions:

$$Pf_{\Theta}(T_x) = Pf_{\Theta}((T_s \quad T_{n_r})) = (T_{\theta_1} \otimes T_s) \oplus (T_{\theta_2} \otimes T_{n_r}) \quad (14)$$

Thus, we adjust the trust in the network output based on the trust in both the parameters and the features, ensuring that the trust propagation takes into account the trust in the parameters.



(a) Structure of a Perceptron (b) Mirrored Trust Structure



(c) Neuron Operation (d) Trust Function Operation

Fig. 4: Illustration of the trust node structure and computation

C. Trust Nodes and Trust Functions

In order to formalize the construction of Pf_{Θ} , we introduce two fundamental concepts: the *Trust Node* (Fig. 4b) and the *Trust Function* (Fig. 4d). These components provide the basic mechanisms for propagating trust through the structure of a neural network.

Definition 5 (Trust Node). A Trust Node is an abstract computational unit associated with a neuron in a neural network. It receives trust opinions on the neuron's inputs and produces a trust opinion on the neuron's output. The structure of a trust node mirrors that of its corresponding neuron, but its computation is defined over trust opinions using operators such as discounting and fusion.

Definition 6 (Trust Function). A Trust Function models the transformation of trust through a trust node. It defines how trust opinions on the inputs of a neuron are combined to produce a trust opinion on the output. For a neuron that computes

$$z^{(l)} = \sum \theta_i^{(l)} \cdot x_i^{(l)}, \quad x^{(l+1)} = f^{(l)}(z^{(l)}),$$

the corresponding trust computation is given by:

$$T_{z^{(l)}} = \bigvee_{i=1}^d T_{x_i^{(l)}} \otimes T_{\theta_i^{(l)}}, \quad T_{x^{(l+1)}} = T_f^{(l)}(T_{z^{(l)}}),$$

where:

- \otimes is a trust discounting operator,
- \bigvee and \oplus is a trust fusion operator,
- $T_f^{(l)}$ is the trust-equivalent of the activation function. Usually set to identity function.

Collectively, trust nodes form a Trust Nodes Network (TNN), which mirrors the neural network's architecture. We formalize this in the following Definition.

Definition 7 (Trust Nodes Network). The Trust Nodes Network is a structured composition of trust nodes, where each trust node corresponds to a neuron in the underlying neural network. This network mirrors the architecture of the neural model and is responsible for propagating trust values across layers. The Trust Nodes Network computes trust assessments at each stage of inference by applying trust-specific reasoning operations aligned with the neural computation flow.

Motivated by the structure of trust propagation in earlier sub-problems (Eq. (14)), the discount operator \otimes models how trust in an input is modulated by trust in the associated parameter, while the fusion operator \vee combines these trust contributions across inputs. The fusion operators should be associative or generalizable [38] to support multiple inputs.

With these definitions in place, we integrated trust nodes and trust functions into a parallel trust reasoning framework that mirrors neural network computation. Although this was first illustrated in the perceptron case, two critical challenges remain: how to quantify the trustworthiness of model parameters learned from datasets of variable quality, and how to generalize trust propagation to deeper, more complex neural architectures. These challenges motivate the design of the Parallel Trust Assessment System (PaTAS), a scalable architecture for trust propagation in neural networks. The following section presents its design, components, and theoretical foundations.

V. PATAS FOR NEURAL NETWORKS

When neural network parameters such as θ_1 and θ_2 are learned (e.g., by backpropagation), their trustworthiness depends on the data used for training. If the dataset contains mislabeled or biased samples, parameter trust will be compromised. Yet this is only part of the problem. As discussed in Section II-A, Subjective Logic represents trust as a subjective binomial opinion with the three components trust, distrust, and uncertainty. This decomposition allows us to distinguish between different causes of unreliability: distrust may arise from systematic issues such as mislabeled or poisoned data, while uncertainty reflects variability or noise in the data. A central challenge, therefore, is how to exploit this advantage of subjective logic in order to make this distinction in practice.

To address the above challenges, we introduce the *Parallel Trust Assessment System (PaTAS)*, a framework designed to systematically propagate trust assessments through a neural network. PaTAS operates in parallel with the standard neural architecture and maintains a corresponding structure that mirrors the network's topology. It enables the computation of trust in the output by integrating two key sources:

- 1) the trust in the input features at inference time, and
- 2) the trust in the parameters, derived from the training dataset using a trust assessment function. This includes both the trust in the input features of the training samples and the trust in the corresponding labels.

A. PaTAS Overview

The PaTAS is designed to continuously evaluate the trustworthiness and preservation of properties, such as accuracy, during the inference of a neural network. For example, an input x might be accurate, unbiased and trustworthy (i.e., high trust score value T_x), while the corresponding output y may still be unreliable due to biased or poorly calibrated parameters θ (i.e., low trust score value T_y). Rather than altering the neural network itself, PaTAS operates alongside it to evaluate the trustworthiness of computations (see Fig. 5). It mirrors the computations through a parallel set of *trust nodes*, each aligned with a corresponding neuron. These trust

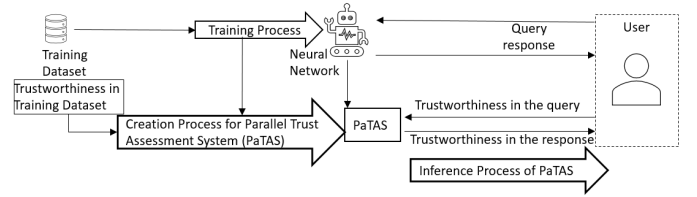


Fig. 5: Overview of PaTAS Usage

nodes use subjective logic reasoning operations to propagate trust values in a way that reflects the neural computations. The detailed operational flow of PaTAS across feedforward, backpropagation, and inference is illustrated in Fig. 6.

PaTAS is composed of four main modules: Trust Feedforward, Output-Trust Aggregation, GenIPTA, and Parameter-Trust Update. Among these, the Parameter-Trust Update requires a more detailed treatment, since its design parallels the role of backpropagation in neural network training and involves an elaborate reasoning operations. We therefore dedicate a separate subsection to it.

1) *Trust Feedforward*: The Trust Feedforward function propagates trust values through the Trust Nodes Network in alignment with the neural network's inference flow. It mirrors the layer-wise computations of the original model, but operates entirely on trust values. At each layer, trust in the inputs and parameters is combined using the trust operations defined in the Trust Function (including discounting and fusion), capturing how evidence flows through the network. This process produces a trust opinion for each output of the neural network while also storing intermediate trust scores, which are later used by the Parameter-Trust Update.

2) *Output-Trust Aggregation*: The Trust Feedforward process produces a vector of trust values, one for each output neuron. The goal of Output-Trust Aggregation is to combine these individual opinions into a single, consolidated trust score representing the overall trust in the network's output. This combination is done by means of a fusion operator. Alternatively, instead of aggregating across all outputs, one may directly consider the trust assigned to the final decision. For instance, in digit classification, if the model outputs a probability 0.9 for class '1', the trust in the decision can be taken as the trust score associated with the output neuron for class '1'.

3) *GenIPTA*: The purpose of the GenIPTA function is to dynamically construct a function tailored to a specific inference. This function, called Inference-Path Trust Assessment (IPTA), reflects how trustworthy the exact computational path taken during inference is. When an inference is performed, contextual information is recorded, such as the list of neurons activated along the path. In this case, the activation trace is used to instantiate a temporary subnetwork of Trust Nodes containing only the trust nodes corresponding to those activations, thereby mirroring the precise inference path of the neural network.

Contextual information is not limited to activation traces. Typically, all neurons are activated to some degree, but only a subset of these activations is strongly relevant to the decision.

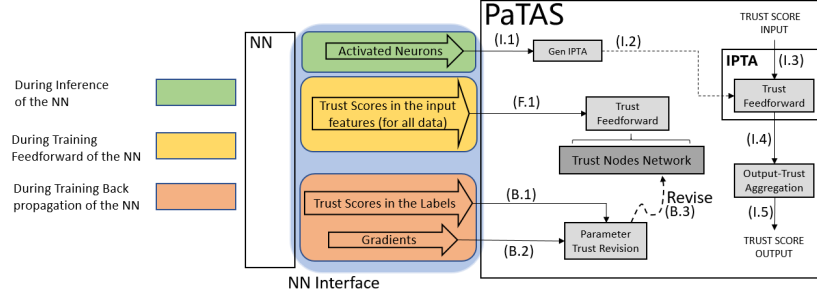


Fig. 6: Functional flow of the PaTAS framework integrated with a neural network, illustrating how trust is propagated and revised during feedforward, backpropagation, and inference.

Feedforward (Training Phase): The NN interface extracts trust scores from input features for all data being processed. These scores (F.1) are passed to the *Trust Feedforward* module, which propagates them through the *Trust Nodes Network* in parallel with the neural computations. During training, *Trust Feedforward* also stores the trust scores of all intermediate computations in the NN, to be reused in backpropagation.

Backpropagation (Training Phase): Once the NN computes gradients, the trust scores of the labels (B.1) and the gradients are provided to the *Parameter-Trust Update* (B.2). Gradients indicate how dataset labels affect parameter updates, while label trust determines whether these updates should be considered reliable. The stored intermediate trust values are also incorporated in this process. Finally, the *Parameter-Trust Update* module refines the trust values of the *Trust Nodes Network* parameters (B.3), aligning them with the evolving learning dynamics.

Inference Phase: During operation, contextual information such as the set of activated neurons is retrieved. This information is passed to *GenIPTA* (I.1), which constructs an *Inference Path Trust Assessment (IPTA)* corresponding to the specific inference path (I.2). The IPTA takes input trust scores (I.3), propagates them along the path, and produces trust scores for each NN output. These can then be passed to the *Output-Trust Aggregation* (I.4) which will combine them into a single consolidated trust score representing the reliability of the prediction (I.5).

GenIPTA can therefore operate on truncated activation sets that retain only the most relevant neurons, which allows for more focused and interpretable trust assessments of the actual decision-making process. GenIPTA can also be more complex and take into account the specific activation values while keeping all neurons, performing a weighted trust assessment where stronger activations contribute more heavily to the inference.

B. Parameter-Trust Revision

The design of Parameter-Trust Update is motivated by the standard backpropagation algorithm used in neural network training. In backpropagation, the gradient of the loss function with respect to the network parameters indicates how much each parameter contributes to the output error. Thus, for a parameter θ , the update is given by

$$\theta \leftarrow \theta - l_r \frac{\partial \mathcal{L}}{\partial \theta}, \quad (15)$$

where l_r is the learning rate, \mathcal{L} is the loss function and $\frac{\partial \mathcal{L}}{\partial \theta}$ is the gradient. The gradients are obtained recursively through the chain rule:

$$\frac{\partial \mathcal{L}}{\partial \theta_{i,j}^{(l)}} = \delta_i^{(l)} x_j^{(l-1)}, \quad (16)$$

where $\theta_{i,j}^{(l)}$ denotes the weight connecting neuron j in layer $(l-1)$ to neuron i in layer l , $\delta_i^{(l)}$ is the error term of neuron i in layer l , and $x_j^{(l-1)}$ is the activation of neuron j in the previous layer.

The Parameter-Trust Update mechanism integrates training signals, specifically the gradients g computed during backpropagation, together with trust in the labels in the training batch and the trust scores of intermediate computations stored during feedforward. The Parameter-Trust Update ensures that parameter trust reflects both observed model behavior and trust in the training data.

The Parameter-Trust Update process is formally described in Alg. 1. For each layer in the network, the system revises the trust parameters of every trust node based on the evidence available from the training batch, the input processed during feedforward, and the gradients. It then updates these trust values to reflect the reliability of the newly updated parameters, using evidence from the learning rate and the intermediate trust scores.

The algorithm runs once per batch and begins by computing a combined trust opinion over all labels in the current batch:

$$T_{y_{\text{batch}}} = \bigwedge_{y \in y_{\text{batch}}} T_y$$

This combined opinion serves as a foundation for conditioning the trust in each parameter.

For each neuron $n_i^{(l)}$ at index i in layer l , the system computes:

- $T_{n_i^{(l)}|y_{\text{batch}}}$, the trust conditioned on the current batch labels. This is inferred by checking each incoming weight gradient $g_{i,j}^{(l)}$: if $|g_{i,j}^{(l)}| < \epsilon$ it is counted as positive evidence r , otherwise as negative evidence s . These counts are then mapped into a binomial opinion using the Baseline-Prior Quantification model.

Algorithm 1 Parameter-Trust Update Algorithm

```

1: Function ParameterTrustUpdate( $g, T_y, \epsilon$ )
2:   Summary: Revises and updates the trust parameters
    of the Trust Nodes Network by combining gradient evi-
    dence, label trust, and neuron input trust, under mini-batch
    training.
3:   Step 1: Compute the aggregated trust in the labels
4:    $T_{y_{\text{batch}}} \leftarrow \bigwedge_{y \in y_{\text{batch}}} T_y$ 
5:   for each layer  $l$  do
6:     for each neuron  $n_i^{(l)}$  do
7:       Step 2: Gather gradient evidence for neuron
         $n_i^{(l)}$ 
8:        $g_i^{(l)} \leftarrow \{g_{i,j}^{(l)} \mid j \in \mathcal{N}(i)\}$ 
9:       Step 3: Compute trust for neuron  $n_i^{(l)}$ 
10:       $T_{n_i|Y_{\text{batch}}} \leftarrow \text{NodeTrust}(g_i^{(l)}, T_{y_{\text{batch}}}, \epsilon)$ 
11:      Step 4: Deduce overall trust in neuron  $n_i^{(l)}$ 
12:       $T_{n_i|Y_{\text{batch}}} \leftarrow \text{DeduceTrust}(T_{n_i|y_{\text{batch}}}, T_{n_i|\overline{y_{\text{batch}}}}, T_{y_{\text{batch}}})$ 
13:      for each incoming edge  $j$  to  $n_i^{(l)}$  do
14:        Step 5: Revise parameter trust with node
        trust
15:         $T_{\theta_{i,j}^{(l)}} \leftarrow \text{Revise}(T_{\theta_{i,j}^{(l)}}, T_{n_i|Y_{\text{batch}}})$ 
16:        Step 6: Update parameter trust with auxil-
        iary factors
17:         $T_{\theta_{i,j}^{(l)}} \leftarrow \text{Update}(T_{\theta_{i,j}^{(l)}}, T_{lr}, T_{x_j^{(l-1)}}, T_{y_{\text{batch}}})$ 
18:      end for
19:    end for
20:  end for
21: end Function
22: Function NodeTrust( $g_i^{(l)}, T_{y_{\text{batch}}}, \epsilon$ )
23:   Count  $r$ : #edges with  $|g_{i,j}^{(l)}| < \epsilon$ 
24:   Count  $s$ : #edges with  $|g_{i,j}^{(l)}| \geq \epsilon$ 
25:   Map  $(r, s)$  into a binomial opinion using Baseline-
    Prior Quantification
26:   return  $T_{n_i|y_{\text{batch}}}$ 
27: end Function
28: Function DeduceTrust( $T_{n_i|y_{\text{batch}}}, T_{n_i|\overline{y_{\text{batch}}}}, T_{y_{\text{batch}}}$ )
29:    $T_{n_i|\overline{y_{\text{batch}}}} \leftarrow (0, 0, 1)$ 
30:   return  $T_{n_i|Y_{\text{batch}}} = T_{y_{\text{batch}}} \odot (T_{n_i|y_{\text{batch}}}, T_{n_i|\overline{y_{\text{batch}}}})$ 
31: end Function
32: Function Revise( $T_{\theta_{i,j}^{(l)}}, T_{n_i|Y_{\text{batch}}}$ )
33:   return  $T_{\theta_{i,j}^{(l)}} \odot T_{n_i|Y_{\text{batch}}}$ 
34: end Function

```

- $T_{n_i^{(l)}|\overline{y_{\text{batch}}}}$, the trust when the true batch labels are not y_{batch} . Since no concrete evidence is available for this case, it is initialized to a vacuous opinion: $(0, 0, 1)$.
- A deduced trust $T_{n_i^{(l)}|Y_{\text{batch}}}$ using the inferential deduction operator \odot , combining the two conditional opinions with $T_{y_{\text{batch}}}$.

Then for each incoming edge j of neuron i , the parameter trust $T_{\theta_{i,j}^{(l)}}$ is updated in two stages:

- 1) Revision with deduced trust using fusion:

$$T_{\theta_{i,j}^{(l)}} \leftarrow T_{\theta_{i,j}^{(l)}} \odot T_{n_i^{(l)}|Y_{\text{batch}}}$$

- 2) Adjustment with auxiliary factors: During training, the update of parameter $\theta_{i,j}^{(l)}$ depends not only on its current value but also on auxiliary factors such as the learning rate l_r , the input feature $x_j^{(l-1)}$, and the label y_{batch} (see Eqs. (15) and (16)). To reflect this, we adjust parameter trust using:

$$T_{\theta_{i,j}^{(l)}} \leftarrow T_{\theta_{i,j}^{(l)}} \odot (T_{x_j^{(l-1)}} \odot T_{y_{\text{batch}}}).$$

Here, \odot is the binomial multiplication operator and \odot is defined as:

$$\begin{aligned}
 (b, d, u) &= (b_1, d_1, u_1) \odot (b_2, d_2, u_2), \\
 b &= \min(b_1, b_2), \\
 d &= \max(d_1, d_2), \\
 u &= 1 - (b + d).
 \end{aligned} \tag{17}$$

This formulation reflects the fact that both unreliable features and mislabeled data can strongly bias parameter updates. Although it might introduce some redundancy, it ensures that trust in both input features and labels is explicitly propagated into the parameter trust.

This process refines the trust in each parameter by incorporating both gradient-based behavioral evidence and auxiliary trust factors, allowing the PaTAS to align parameter trust with the training dynamics of the neural network.

C. Theoretical Properties of PaTAS

1) *Convergence of PaTAS:* The convergence of the PaTAS is governed by the stability of its inputs and the structure of its recursive Parameter-Trust Update process. During training, PaTAS updates the internal trust opinions associated with network parameters using trust signals from the inputs, labels, hyperparameters, and gradient information. This update mechanism is outlined in Alg. 1. As proved in Theorem 3, the PaTAS converges under specific situation. This convergence lies on some specific characteristic of the operators used to feedforward and revise the trust in the parameters θ .

Before delving into the specifics of PaTAS convergence, we first present a key theorem that will serve as a foundational tool in establishing the convergence properties of PaTAS.

Theorem 1 (Convergence of Subjective Logic Arithmetic Sequence). *Let (Ω, \ominus) a group. Ω is set of opinions which can be seen as $[0, 1]^4$. Let $\omega_n = (b_n, d_n, u_n, a_n) \in \Omega$ be a sequence defined recursively by the operator \ominus as follows:*

$$\omega_{n+1} = \omega_n \ominus q,$$

where $q \in \Omega$ and the operator \ominus is a fusion operator.

Then the sequence (ω_n) converges in $[0, 1]^4$ to a limit:

Proof. See Appendix C □

Theorem 2 (Convergence of Subjective Logic Geometric Sequence). *Let (Ω, \odot) a group. Ω is set of opinions which can be seen as $[0, 1]^4$. Let $\omega_n = (b_n, d_n, u_n, a_n) \in \Omega$ be a sequence defined recursively by the operator \odot as follows:*

$$\omega_{n+1} = \omega_n \odot q,$$

where $q \in \Omega$ and the operator \odot is the binomial multiplication defined by:

$$\begin{cases} b_{x \odot y} = b_x b_y + \frac{(1 - a_x) a_y b_x u_y + a_x (1 - a_y) u_x b_y}{1 - a_x a_y}, \\ d_{x \odot y} = d_x + d_y - d_x d_y, \\ u_{x \odot y} = u_x u_y + \frac{(1 - a_y) b_x u_y + (1 - a_x) u_x b_y}{1 - a_x a_y}, \\ a_{x \odot y} = a_x a_y. \end{cases}$$

Then the sequence (ω_n) converges in $[0, 1]^4$ if $a_q < 1$. In particular:

- $a_n = a_0 a_q^n \rightarrow 0$ as $n \rightarrow \infty$,
- $d_n \rightarrow 1$ if $d_q > 0$, and $d_n = d_0$ if $d_q = 0$,
- $b_n \rightarrow 0$ if $p_q = b_q + a_q u_q < 1$, and $b_n = b_0$ if $p_q = 1$.

Proof. See Appendix C \square

Theorem 3 (Convergence of PaTAS Creation). *Let a neural network be trained until convergence, and let its associated PaTAS operate with:*

- a stable input trust assessment T_x ,
- a stable label trust assessment T_y ,
- a stable hyperparameter trust T_{lr} ,

Let $T_\theta^{(n)}$ denote the trust opinion at iteration n for parameter θ . If the revision of the trust in the weights is performed as specified in Alg. 1, then the PaTAS parameter will also converges.

Proof. Assume that the neural network training process converges, implying that weight updates become increasingly small, and the back propagation gradients g stabilize. This stability in g reflects the fact that the model has reached a minimum or stable loss value.

Moreover, the convergence of T_x , T_y , and T_{lr} implies that trust inputs to the update mechanism are stable. Consequently, each new trust update for T_θ is computed using consistent and bounded evidence, which over time results in the stabilization of the subjective opinions assigned to T_θ .

In detail:

- T_x converging implies $(T_y, \text{converging assuming the same } T_\theta)$ implies convergence of $T_{y_{batch}}$.
- g converges so $T_{\theta|y_{batch}}$ will remain the same.
- since $T_{\theta|y_{batch}}$ is a deterministic calculation from converges $T_{\theta|y_{batch}}$ and $T_{y_{batch}}$ therefore $T_{\theta|y_{batch}}$ also converges
- Finally $T_\theta \leftarrow T_\theta \odot T_{\theta|y_{batch}}$ converges if \odot is set to any fusion operator or the binomial multiplication operator (see Theorem 1 and 2).
- The function f_{upd} in our implementation is based T_θ , T_{lr} and T_x which all converge.

Therefore, assuming convergence of T_x , T_y , T_{lr} , and g , the trust values for all PaTAS parameters stabilize as training progresses, proving convergence of PaTAS creation. \square

2) Symmetry and Invariance Properties of PaTAS:

Theorem 4 (PaTAS Inference on Vacuous Input Yields Vacuous Output). *Let $\omega^0 = (0, 0, 1, a)$ denote a vacuous binomial*

opinion over any variable, with arbitrary base rate $a \in [0, 1]$. Then the following two properties hold:

- 1) Discounting a Vacuous Opinion Yields a Vacuous Opinion.

For any trust value binomial opinion ω_B^A , the discounted opinion

$$\omega_X^{[A;B]} = \omega_B^A \otimes (0, 0, 1, a) = (0, 0, 1, a)$$

- 2) PaTAS Feedforward on Vacuous Input Yields Vacuous Output.

Let $T_x = (0, 0, 1, a)$ be the trust assessment of an input to PaTAS. Then for any parameter trust configuration T_θ , the output trust assessment satisfies:

$$T_y = IPTA(T_x) = (0, 0, 1, a).$$

Proof. See Appendix C \square

Definition 8 (Symmetric Binomial Opinions). *Let $x = (b, d, u)$ be a binomial opinion with belief b , disbelief d , and uncertainty u , where $b + d + u = 1$. The opinion $\bar{x} = (d, b, u)$ is called the symmetric of x . Two binomial opinions x and \bar{x} are symmetric if they share the same uncertainty and have inverted belief and disbelief, i.e.,*

$$x = (b, d, u), \quad \bar{x} = (d, b, u) \quad \text{with } b + d + u = 1.$$

Theorem 5 (Symmetric Inference under PaTAS). *Let T_θ be a fixed PaTAS inference operator based on subjective logic, and let $x = (b, d, u)$ be any binomial opinion with symmetric counterpart $\bar{x} = (d, b, u)$. Then the outputs $y = T_\theta(x)$ and $\bar{y} = T_\theta(\bar{x})$ are also symmetric, i.e.,*

$$y = (b', d', u'), \quad \bar{y} = (d', b', u').$$

In particular, the uncertainty is preserved:

$$u_y = u_{\bar{y}},$$

and the belief in one output equals the disbelief in the other:

$$b_y = d_{\bar{y}}, \quad d_y = b_{\bar{y}}.$$

Moreover, for the fully trusted input $x = (1, 0, 0)$, the output satisfies $d_y = 0$, and for the fully distrusted input $\bar{x} = (0, 1, 0)$, the output satisfies $b_{\bar{y}} = 0$.

Proof. See Appendix C \square

VI. EVALUATION AND RESULTS

The goal of our evaluation is to validate the Parallel Trust Assessment System (PaTAS) both theoretically and empirically. Specifically, we aim to demonstrate that PaTAS produces interpretable trust estimates that (i) converge during training under specific conditions, (ii) respect symmetry and invariance properties, and (iii) are able to provide interpretable assessments under realistic conditions such as noisy features, corrupted labels, or adversarial perturbations.

Our evaluation approach combines controlled synthetic degradations with real-world datasets. We systematically vary the trustworthiness of inputs ranging from fully trusted, fully uncertain to fully distrusted and observe how created PaTAS propagates these trust assessments through the network. For

each scenario, we track three complementary metrics: trust mass, uncertainty mass, and distrust mass. We also compare these values against standard model accuracy to understand how input trust affects output reliability.

We conduct three experiments of increasing complexity:

- 1) *Breast Cancer Classification*, to assess behavior on a small, tabular medical dataset.
- 2) *MNIST Digit Classification*, to evaluate PaTAS across multiple neural architectures under controlled uncertainty.
- 3) *Poisoned MNIST*, to evaluate PaTAS and IPTA in the presence of adversarial corruption and data poisoning.

A. Experimental Setup

1) *Experiment 1 - Breast Cancer Classification*: We use the Breast Cancer Wisconsin (Diagnostic) Dataset [39], containing 569 samples with 30 numeric features (e.g., radius, area, symmetry) derived from breast mass fine needle aspirates. The neural network classifies the tumors into benign or malignant categories. The neural network architecture consists of 30 input neurons, 16 hidden neurons, and 2 output neurons, with ReLU activation in the hidden layer and Softmax in the output. The model is trained for 15 epochs with a batch size of 64 and a learning rate of 0.2, achieving 98% accuracy when the data are not modified.

Training Data trust is assessed with three trust profiles for input features and label: fully trusted (1, 0, 0), fully distrusted (0, 1, 0), and fully uncertain (0, 0, 1). These are combined (for inputs and label assessment) to form nine combinations. Data features and labels trust degradation may arise in practice from poor-quality imaging, human annotation errors, or flaws in the preprocessing pipeline. In all experiments, degradations are simulated using controlled perturbation functions. For fully uncertain opinions, we introduce additive uniform noise

$$x' = x + \epsilon, \quad \epsilon \sim U(-a, a),$$

where

$$a = 0.3 \times \max(\text{feature}).$$

Each feature is perturbed with probability 0.3. After noise addition, if x' lies outside the valid range $[\min(\text{feature}), \max(\text{feature})]$, it is clipped to the corresponding boundary:

$$x' = \begin{cases} \min(\text{feature}), & x' < \min(\text{feature}), \\ \max(\text{feature}), & x' > \max(\text{feature}), \\ x', & \text{otherwise.} \end{cases}$$

To model distrust, we instead corrupt input features entirely.

Similarly, label degradation is simulated by randomly replacing labels (or replacing all labels for distrust). In addition to these extreme cases, we also consider two intermediate scenarios:

- i. a mild degradation, where features are perturbed with probability 0.15 and the trust assessment is set to (0.25, 0, 0.75), and
- ii. the same scenario for fully uncertain input features and fully uncertain labels, but where the assessment is set to (0.25, 0.25, 0.5), reflecting partial distrust and trust, instead of fully uncertain opinion (0, 0, 1).

2) *Experiment 2 - MNIST*: In this experiment, we evaluate the PaTAS framework on the MNIST dataset [40], which consists of 60,000 training images and 10,000 test images, each representing a digit from 0 to 9. Each image has 784 features (28x28 pixels). The neural network classifies these images into one of the 10 digit classes.

We test three neural network architectures:

- Architecture 1: 784 input neurons, 5 hidden neurons, 10 output neurons (784-5-10).
- Architecture 2: 784 input neurons, 10 hidden neurons, 10 output neurons (784-10-10).
- Architecture 3: 784 input neurons, 20 hidden neurons, 10 output neurons (784-20-10).

Although these architectures are not state-of-the-art for MNIST, they are sufficient to evaluate the behavior of PaTAS across different model sizes and to demonstrate how trust propagates through the network. Each model uses the ReLU activation function for the hidden layer and Softmax for the output. Models are trained for 10 epochs with a batch size of 18 and a learning rate of 0.001, achieving test accuracies of 89%. We evaluate using fully uncertain Training data trust assessment functions for both input features and labels.

3) *Experiment 3 - Poisoned MNIST*: In this experiment, we evaluate the PaTAS framework on a poisoned version of the MNIST dataset, where one third of the training images are corrupted: labels of digits 6 and 9 are flipped, and at the same time a visible patch of fixed size is added at the top-left corner of the corresponding images. This combination follows common practices in data poisoning and backdoor attack scenarios [41]. The remaining two thirds of the data remain clean. This setup allows us to examine how PaTAS responds to the simultaneous presence of corrupted labels and adversarial triggers that can undermine both model performance and trust.

We use Architecture 3 from the previous experiment, consisting of an input layer with 784 neurons, one hidden layer with 20 neurons, and an output layer with 10 neurons (784-20-10).

For the poisoned dataset, trust is assigned as follows:

- Pixels corresponding to the patch are considered distrusted, while all other pixels are trusted.
- Labels for patched images of digits 6 and 9 are distrusted, while others are trusted.

This setup helps the PaTAS framework focus on potentially corrupted areas, while trusting the unaffected parts of the data.

Implementation Details: During inference, multiplication operations in the feedforward phase are implemented using SL trust discounting, while addition operations employ a generalized SL averaging fusion operator to support summations over multiple inputs. Trust revision uses the same averaging fusion. The PaTAS framework and primary neural network were implemented in Python 3.9 using NumPy. Two main modules were developed: `PrimaryNN`, which handles network structure, training, and inference, and `PaTAS`, which implements trust assessment and propagation functions defined in the operational flow (Fig. 6). These modules interact to dynamically evaluate and update trust during feedforward and backpropagation. The full implementation, including all

modules, dependencies, and experiment scripts, is available as supplementary materials, with detailed instructions for reproducing all experiments.

B. Results and Analysis

The evaluation of the Parallel Trust Assessment System (PaTAS) is conducted during the training phase of the neural network. After each iteration of training (i.e., a complete feedforward and backpropagation cycle), we assume that an inference is performed, and we assess the trustworthiness of the corresponding output. This assessment is carried out under three input trust profiles:

- *Fully Trusted Input*: where the trust in the data is considered trusted (row 1 in each figure).
- *Fully Uncertain Input*: where the trust in the data is considered uncertain (row 2 in each figure).
- *Fully Distrusted Input*: where the data is assumed to be unreliable (row 3 in each figure).

For each of these three input types, we track and plot the evolution of three key metrics over the course of training:

- *Trust Mass*: representing the level of confidence in the output.
- *Uncertainty Mass*: representing the uncertainty associated with the assessment.
- *Distrust Mass*: representing the level of disbelief in the output.

As a result, for each evaluation, 9 distinct plots are generated, corresponding to the 3 input types (fully trusted, fully uncertain, and fully distrusted) and each of the three key metrics (trust, uncertainty, and distrust).

All results are depicted in Figs. 9 and 11 to 19 (Appendix D) and summarized in Tables I to IV. They confirm several theoretical properties of PaTAS proven in Section V-C:

- *Convergence of Trust Assessment*: when accuracy converges, trust assessment converges as expected.
- *Inference on fully uncertain input yields fully uncertain output*: a fully uncertain input always produces a fully uncertain output.
- *Symmetric Inference*: when input trust assessments are symmetric, the inference results remain symmetric.

For detailed results, we primarily focus on the three plots corresponding to fully trusted input. Processing uncertain inputs naturally yields uncertain outputs, while symmetry implies that trusted and distrusted cases mirror each other. We also note that when processing fully trusted inputs, the distrust mass generally remains close to zero. Since trust, uncertainty, and distrust sum to 1, analyzing only the trust mass is sufficient in such cases.

Experiment 1 – Breast Cancer Classification: When both features and labels are clean, the model steadily improves and achieves high accuracy. Clean features with corrupted labels cause accuracy to collapse, while noisy labels yield only moderate and unstable learning. With corrupted features, accuracy remains low in all cases, even with clean labels. Noisy features with clean labels still permit relatively strong learning, but performance breaks down when labels

are corrupted or noisy. Importantly, corrupted labels mislead the model: training accuracy appears high while test accuracy remains poor, showing that the model learns, but learns the wrong mapping. In contrast, noisy labels prevent learning altogether, regardless of whether features are clean or noisy. Corrupted features eliminate the ability to learn in any setting.

We evaluated the system for two values of ϵ (0.1 and 0.01). We recall that ϵ is the threshold used by the NodeTrust function in Alg. 1.

- For $\epsilon = 0.01$: fully uncertain X stabilizes trust mass around 0.28 for uncertain Y and 0.30 for trusted Y . Fully trusted X yields rapid increases, stabilizing at ~ 0.27 for uncertain Y and 0.79 for trusted Y .
- For $\epsilon = 0.1$: the system becomes less sensitive to gradients. When X is uncertain, trust mass stabilizes around 0.31 for uncertain Y , and 0.34 for trusted Y . Fully trusted X gives 0.33 for uncertain Y , and 0.90 for trusted Y . No matter ϵ value, when X or Y are distrusted the trust mass rapidly falls to 0.

Overall, there is a strong positive correlation between trust mass and test accuracy. When ϵ is smaller, the trust level of Y (trusted, uncertain) has a weaker influence on the final trust mass. Notably, setting X as fully uncertain while keeping Y trusted yields a higher trust mass (0.30) than the opposite case, where X is trusted but Y uncertain (0.27). Furthermore, replacing a fully uncertain opinion (0, 0, 1) assessment for X and Y with a mixed assessment (0.25, 0.25, 0.5) reduces the trust mass from 0.31 to 0.17, showing that distrust degrades performance more strongly than trust enhances it.

Experiment 2 – MNIST Dataset: As shown in Table II, when features and labels are noised, accuracy improves as model size increases. However, training accuracy remains consistently lower than test accuracy, with the gap widening for larger networks. This indicates that the models learn useful representations but cannot fully fit noisy training data, while still generalizing well on clean test samples. Test accuracy greater than training accuracy across all architectures, suggesting training noise makes the model underestimate its learning progress. Larger architectures yield more stable trust assessments, likely due to smaller average gradient magnitudes.

Experiment 3 – Poisoned MNIST Dataset: In this experiment, PaTAS successfully distinguishes clean from poisoned labels even when corruption is moderate. Trust mass grows rapidly for clean and poisoned labels. However, poisoned labels remain relatively low, demonstrating effective detection of untrustworthy labels (Table III). Patch size strongly affects this separation:

- 1×1 : minimal effect, where the trust mass of poisoned labels remains low, with only a small difference compared to clean labels.
- 4×4 and 20×20 : clear separation between clean and poisoned trust scores.
- 27×27 : collapse of trust assessment, reflecting domination by the trigger.

In order to evaluate the IPTA, we also trained a binary neural network on the poisoned dataset with a patch size of 4×4 , achieving train/test accuracy of approximately 93%. Table IV

Trust Assessment	$\epsilon = 0.01$	$\epsilon = 0.1$	Train (%)	Test (%)
X fully distrusted, Y distrusted	0	0	63	38
X fully distrusted, Y fully uncertain	0	0	53	62
X fully distrusted, Y fully trusted	0	0	63	63
X fully uncertain, Y distrusted	0	0	96	3
X fully uncertain, Y fully uncertain	0.28	0.31	53	46
X fully uncertain, Y fully trusted	0.30	0.34	97	96.49
X fully trusted, Y distrusted	0	0	98	0.88
X fully trusted, Y fully uncertain	0.27	0.33	52	89
X fully trusted, Y fully trusted	0.79	0.90	98	99
(0.25, 0, 0.75)	-	0.38	71	90
(0.25, 0.25, 0.5)	-	0.17	53	46

TABLE I: Summary of final trust mass and corresponding train/test accuracies (in %) for the Breast Cancer Classification Experiment.

reports both accuracy and the corresponding trust opinions for clean digits (3 and 6) and poisoned digit 6 datasets. Clean samples maintain high accuracy with balanced trust and uncertainty masses, with slightly better trust–uncertainty balance for digit 3. In contrast, poisoned samples show a drastic accuracy drop, lower trust, and higher uncertainty, alarming for untrustworthy predictions. Furthermore, when we explicitly distrust the patch pixels while trusting the remaining inputs, the resulting trust opinion becomes (0.35, 0.1, 0.55). These results show that PaTAS provides interpretable warnings about poisoned outputs.

VII. DISCUSSION

A. Discussion of PaTAS Findings

PaTAS provides a principled mechanism for evaluating the runtime trustworthiness of neural network outputs. Beyond dynamic inference assessment, it can also estimate a static trust opinion of the network itself. The core principle is that a trustworthy model should preserve trust: a fully trusted input producing a highly trusted output indicates that the model does not erode trust during inference. Accordingly, the overall model trustworthiness can be defined as the trust score of the PaTAS feedforward function ($PaTAS_{FF}$) under a fully trusted input:

$$T(\text{NN}) = PaTAS_{FF}((1, 0, 0)),$$

which quantifies how well trust is maintained throughout the network. Using this definition, we compared the Inference Path Trust Assessment (IPTA) for benign and adversarial inputs in Experiment VI-A3, where the adversarial case used a 4×4 patch injection. Results show a clear degradation in trust for adversarial samples: benign inputs achieved $t = 0.484$, $d = 0$, $u = 0.516$, while patched inputs dropped to $t = 0.449$, $d = 0$, $u = 0.551$. This demonstrates the sensitivity of PaTAS to local perturbations, offering a quantitative indicator of reduced reliability.

Although trust mass and accuracy are often correlated, our results reveal meaningful divergences. In the breast cancer task (Table I), assigning a mixed trust profile (0.25, 0, 0.75) to all features and labels yields a final trust mass of 0.38, higher than the fully uncertain-features case, yet the latter achieves better test accuracy (96.49% vs. 90%). Thus, higher trust mass does not always imply superior predictive performance. A similar effect appears in the poisoned MNIST experiments (Table IV):

Hidden Neurons	Trust mass t	Train (%)	Test (%)
5	0.28	31	58
10	0.30	37	71
20	0.32	39	76

TABLE II: Summary of final trust mass for the MNIST Classification Experiment.

the clean digit “6” attains slightly higher accuracy than “3” (93.52% vs. 93.14%), but IPTA assigns higher trust to “3” ($t = 0.484$) than to “6” ($t = 0.482$). These differences show that PaTAS captures aspects of reliability beyond accuracy, reflecting the stability and interpretability of inference paths.

Interpretation of trust values, like accuracy, depends on the application. In high-stakes settings, a trust score of (0.4, 0, 0.6) may be inadequate, whereas in less critical domains it may suffice. PaTAS enables such contextual interpretation by providing a unified, interpretable metric that can be tracked over time, compared across models, or evaluated under varying conditions. Overall, PaTAS complements traditional metrics by signaling fragility even when accuracy appears high and by identifying stability where accuracy slightly decreases. This duality underscores its relevance in safety-critical contexts where accuracy alone can be misleading.

In practice, precise dataset trustworthiness estimates may be unavailable. PaTAS can still operate by initializing dataset trust to a fully uncertain (vacuous) opinion. While this limits the use of prior trust evidence, it preserves the capacity to propagate input-level trust through the model during inference, supporting meaningful trust evaluation even without explicit dataset provenance.

A key technical factor affecting PaTAS behavior is the parameter ϵ , which controls the threshold used to classify gradients as positive or negative evidence during training. As shown in Table I, if ϵ is too small, PaTAS may misinterpret gradients as large even when the model performs well. Consequently, trusted labels may reduce parameter trust, resembling the case $g \rightarrow +\infty$ in Table V. Proper calibration of ϵ is therefore crucial, and can be tuned per layer or gradually reduced as training converges to increase sensitivity to smaller gradients.

Finally, trust quantification for $T_{\theta_i|y_{\text{batch}}}$ in the Trust Update Algorithm 1 relies on a simple gradient-counting procedure. Although computationally efficient, it fixes the uncertainty component of the resulting binomial opinion based on the number of input neurons to i . This fixed-uncertainty formulation may not be optimal in all cases, suggesting the need for adaptive quantification schemes. Similarly, $T_{\theta_i|\overline{y_{\text{batch}}}}$ is set to a fully vacuous opinion (0, 0, 1), which, while consistent with the absence of evidence, may not fully leverage prior knowledge when available.

Patch size	Trust mass for 3	Trust mass for 6	Train (%)	Test (%)	Clean 3	Clean 6	Poisoned 3	Poisoned 6
(1×1)	0.92	0.89	78	81	82	35	76	39
(4×4)	0.90	0.83	76	75	88	17	49	21
(20×20)	0.375	0.3	78	75	87	41	0	37
(27×27)	0.05	0.03	80	80	88	68	0	0

TABLE III: Summary of result for the Poisoned MNIST Classification Experiment.

	Accuracy (%)	Trust	Distrust	Uncertainty
Clean 3	93.14	0.484	0.0	0.516
Clean 6	93.52	0.482	0.0	0.518
Poisoned 6	6.19	0.449	0.0	0.551
Poisoned 6 (patch distrusted)	—	0.350	0.100	0.550

TABLE IV: IPTA results for a binary neural network trained on poisoned datasets with patch size 4×4 . Results are obtained by feedforwarding a fully trusted opinion, except for the last row where patch pixels are explicitly distrusted.

$g \rightarrow$	$T_{\theta_i y_{\text{batch}}} \rightarrow$	$T_{y_{\text{batch}}} \rightarrow$	$T_{\theta_i Y_{\text{batch}}} \rightarrow$
0	Trusted	Trusted	Trusted
		Uncertain	Uncertain
		DisTrusted	Uncertain
		Trusted	DisTrusted
$+\infty$	DisTrusted	Trusted	(0,0.5,0.5)
		Uncertain	(0,0.5,0.5)
		DisTrusted	Uncertain

TABLE V: Asymptotic behavior of $T_{\theta_i||Y_{\text{batch}}}$

B. Trust Assessment and AI Security Threats

AI systems are exposed to diverse attacks targeting different stages of the lifecycle. Fig. 7 outlines six key phases: data collection, cleaning and labeling, dataset assembly, network design, model training, and deployment for inference.

A major threat is the *data poisoning attack* [41], where adversaries insert malicious or mislabeled samples into the training data to induce targeted misclassifications. PaTAS mitigates this threat by performing trust assessments at the feature or instance level during training. By computing dataset trustworthiness and propagating input trust through training, PaTAS can flag unreliable predictions during deployment.

Another vector is *model stealing*, in which adversaries issue numerous queries to reconstruct or approximate a deployed model, threatening intellectual property and enabling downstream attacks. By monitoring the trust of queries and corresponding predictions, PaTAS can detect anomalous or low-trust query sequences indicative of model extraction attempts.

A third class, *membership inference attacks*, aims to infer whether a specific data point was part of the training dataset, posing privacy risks. PaTAS supports mitigation by assessing

output trust: consistently high confidence and low uncertainty may signal overfitting or memorization, while balanced trust indicates healthy generalization. Thus, output trust scores can help reveal potential information leakage.

These examples show that trust vulnerabilities can emerge throughout the AI pipeline. The effectiveness of PaTAS depends on the granularity of its assessments, but by enabling trust evaluation across multiple stages, it provides a flexible, context-aware defense mechanism that strengthens AI resilience against diverse adversarial threats.

VIII. CONCLUSION

This paper presented the *Parallel Trust Assessment System (PaTAS)*, a framework that models and propagates trust in neural networks through Subjective Logic. PaTAS introduces a parallel computational structure based on *Trust Nodes* and *Trust Functions*, enabling principled trust propagation alongside standard feedforward and backpropagation processes. The proposed *Parameter Trust Update* and *Inference-Path Trust Assessment (IPTA)* mechanisms jointly quantify how input quality, learned parameters, and activation paths contribute to the overall trustworthiness of model predictions.

Experimental evaluations across real-world and adversarial datasets demonstrate that PaTAS can identify trust degradation under data poisoning and adversarial patching while maintaining interpretability and stability. Results further show that trust scores derived from PaTAS capture reliability aspects not reflected by accuracy alone, offering a complementary view of model performance and robustness.

Beyond accuracy estimation, PaTAS provides a unified probabilistic foundation for quantifying model reliability across the AI pipeline, from dataset trust to inference-level decision confidence. Future work will extend PaTAS to large-scale and multimodal architectures, explore adaptive trust quantification schemes, and incorporate the role of parameter magnitudes in trust reasoning. In particular, we plan to investigate how weight values can inform the generation of the Trust Node network for individual inferences, allowing trust propagation to reflect not only network topology but also parameter influence. Overall, PaTAS establishes a systematic path toward trustworthy, transparent, and reliable neural learning systems.

REFERENCES

- [1] H.-L. E. G. on Artificial Intelligence, “Ethics guidelines for trustworthy ai,” European Commission, Brussels, Tech. Report, Apr. 2019, published 8 April 2019. [Online]. Available: <https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai>
- [2] D. Kowald, S. Scher, V. Pammer-Schindler, P. Müllner, K. Waxnegger, L. Demelius, A. Fessl, M. Toller, I. G. Mendoza Estrada, I. Šimić, V. Sabol, A. Trügler, E. Veas, R. Kern, T. Nad, and S. Kopeinik, “Establishing and evaluating trustworthy ai: overview and research challenges,” *Frontiers in Big Data*, vol. Volume 7 - 2024, 2024. [Online]. Available: <https://www.frontiersin.org/journals/big-data/articles/10.3389/fdata.2024.1467222>

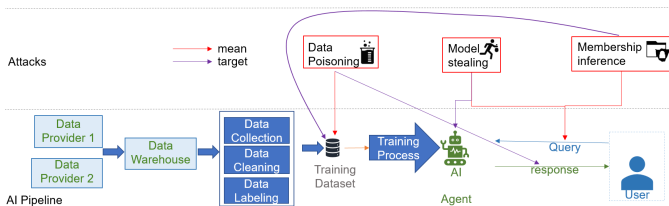


Fig. 7: Attack surface in the AI lifecycle pipeline, from data collection to model deployment.

- [3] J. Wu and J. He, "Trustworthy transfer learning: A survey," 2024. [Online]. Available: <https://arxiv.org/abs/2412.14116>
- [4] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, "Smoothgrad: removing noise by adding noise," 2017. [Online]. Available: <https://arxiv.org/abs/1706.03825>
- [5] A. Jøsang, *Subjective Logic: A Formalism for Reasoning Under Uncertainty*, ser. Artificial Intelligence: Foundations, Theory, and Algorithms. Cham: Springer, 2016. [Online]. Available: <https://doi.org/10.1007/978-3-319-42337-1>
- [6] A. P. Dempster, "Upper and lower probabilities induced by a multivalued mapping," *The Annals of Mathematical Statistics*, vol. 38, no. 2, pp. 325–339, 1967.
- [7] G. Shafer, *A Mathematical Theory of Evidence*. Princeton, NJ: Princeton University Press, 1976.
- [8] K. I. Ouattara, I. Krontiris, T. Dimitrakos, and F. Kargl, "Assessing trustworthiness of ai training dataset using subjective logic – a use case on bias," 2025. [Online]. Available: <https://arxiv.org/abs/2508.13813>
- [9] A. Jøsang, D. Wang, and J. Zhang, "Multi-source fusion in subjective logic," in *2017 20th International Conference on Information Fusion (Fusion)*, 2017, pp. 1–8.
- [10] K. I. Ouattara, A. Petrovska, A. Hermann, N. Trkulja, T. Dimitrakos, and F. Kargl, "On subjective logic trust discount for referral paths," in *2024 27th International Conference on Information Fusion (FUSION)*, 2024, pp. 1–8.
- [11] A. Jøsang, R. Hayward, and S. Pope, "Trust network analysis with subjective logic," in *Proceedings of the 29th Australasian Computer Science Conference*. Australian Computer Society, 2006, pp. 85–94.
- [12] A. Jøsang, R. Hayward, and S. Pope, "Trust network analysis with subjective logic," in *Proceedings of the 29th Australasian Computer Science Conference - Volume 48*, ser. ACSC '06. AUS: Australian Computer Society, Inc., 2006, p. 85–94.
- [13] A. Jøsang and T. Bhuiyan, "Optimal trust network analysis with subjective logic," in *2008 Second International Conference on Emerging Security Information, Systems and Technologies*, 2008, pp. 179–184.
- [14] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins, "Propagation of trust and distrust," in *Proceedings of the 13th International Conference on World Wide Web*, ser. WWW '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 403–412. [Online]. Available: <https://doi.org/10.1145/988672.988727>
- [15] R. Ureña, G. Kou, Y. Dong, F. Chiclana, and E. Herrera-Viedma, "A review on trust propagation and opinion dynamics in social networks and group decision making frameworks," *Information Sciences*, vol. 478, pp. 461–475, 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0020025518309253>
- [16] A. Koster, A. L. C. Bazzan, and M. de Souza, "Liar liar, pants on fire; or how to use subjective logic and argumentation to evaluate information from untrustworthy sources," *Artificial Intelligence Review*, vol. 48, no. 2, pp. 219–235, Aug 2017. [Online]. Available: <https://doi.org/10.1007/s10462-016-9499-1>
- [17] S. Dietzel, R. van der Heijden, H. Decke, and F. Kargl, "A flexible, subjective logic-based framework for misbehavior detection in v2v networks," in *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, 2014, pp. 1–6.
- [18] N. Fotos, K. I. Ouattara, D. S. Karas, I. Krontiris, W. Meng, and T. Giannetos, "Actions speak louder than words: Evidence-based trust level evaluation in multi-agent systems," in *Information and Communications Security*, J. Han, Y. Xiang, G. Cheng, W. Susilo, and L. Chen, Eds. Singapore: Springer Nature Singapore, 2026, pp. 255–273.
- [19] D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro, and K. Rieck, "Dos and don'ts of machine learning in computer security," 2021. [Online]. Available: <https://arxiv.org/abs/2010.09470>
- [20] D. V. Vargas, *One-Pixel Attack: Understanding and Improving Deep Neural Networks with Evolutionary Computation*. Singapore: Springer Singapore, 2020, pp. 401–430. [Online]. Available: https://doi.org/10.1007/978-981-15-3685-4_15
- [21] R. M. Neal, *Bayesian Learning for Neural Networks*, 1st ed., ser. Lecture Notes in Statistics. New York, NY: Springer, 1996, vol. 118, springer Book Archive; eBook ISBN: 978-1-4612-0745-0. [Online]. Available: <https://doi.org/10.1007/978-1-4612-0745-0>
- [22] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," 2015. [Online]. Available: <https://arxiv.org/abs/1505.05424>
- [23] S. Depeweg, J.-M. Hernandez-Lobato, F. Doshi-Velez, and S. Udluft, "Decomposition of uncertainty in Bayesian deep learning for efficient and risk-sensitive learning," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 1184–1193. [Online]. Available: <https://proceedings.mlr.press/v80/depeweg18a.html>
- [24] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," 2016. [Online]. Available: <https://arxiv.org/abs/1506.02142>
- [25] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," 2017. [Online]. Available: <https://arxiv.org/abs/1612.01474>
- [26] J. Gawlikowski, C. R. N. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher, M. Shahzad, W. Yang, R. Bamler, and X. X. Zhu, "A survey of uncertainty in deep neural networks," *Artificial Intelligence Review*, vol. 56, no. 1, pp. 1513–1589, Oct 2023. [Online]. Available: <https://doi.org/10.1007/s10462-023-10562-9>
- [27] E. Begoli, T. Bhattacharya, and D. Kusnezov, "The need for uncertainty quantification in machine-assisted medical decision making," *Nature Machine Intelligence*, vol. 1, no. 1, pp. 20–23, Jan 2019. [Online]. Available: <https://doi.org/10.1038/s42256-018-0004-1>
- [28] S. Devic, T. Srinivasan, J. Thomason, W. Neiswanger, and V. Sharan, "From calibration to collaboration: Llm uncertainty quantification should be more human-centered," 2025. [Online]. Available: <https://arxiv.org/abs/2506.07461>
- [29] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. V. Dillon, B. Lakshminarayanan, and J. Snoek, "Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift," 2019. [Online]. Available: <https://arxiv.org/abs/1906.02530>
- [30] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 2017, pp. 1321–1330.
- [31] M. Sensoy, L. Kaplan, and M. Kandemir, "Evidential deep learning to quantify classification uncertainty," 2018. [Online]. Available: <https://arxiv.org/abs/1806.01768>
- [32] K. I. Ouattara, I. Krontiris, T. Dimitrakos, and F. Kargl, "Quantifying calibration error in neural networks through evidence-based theory," in *2025 28th International Conference on Information Fusion (FUSION)*, 2025, pp. 1–8.
- [33] M. Cheng, S. Nazarian, and P. Bogdan, "There is hope after all: Quantifying opinion and trustworthiness in neural networks," *Frontiers in Artificial Intelligence*, vol. 3, 2020. [Online]. Available: <https://www.frontiersin.org/journals/artificial-intelligence/articles/10.3389/frai.2020.00054>
- [34] Y. Mae, W. Kumagai, and T. Kanamori, "Uncertainty propagation for dropout-based bayesian neural networks," *Neural Networks*, vol. 144, pp. 394–406, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0893608021003555>
- [35] P. Monchot, L. Coquelin, S. J. Petit, S. Marmin, E. Le Pennec, and N. Fischer, "Input uncertainty propagation through trained neural networks," in *Proceedings of the 40th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., vol. 202. PMLR, 23–29 Jul 2023, pp. 25 140–25 173. [Online]. Available: <https://proceedings.mlr.press/v202/monchot23a.html>
- [36] R. Astudillo and J. Neto, "Propagation of uncertainty through multilayer perceptrons for robust automatic speech recognition," 08 2011, pp. 461–464.
- [37] C.-N. Ziegler and G. Lausen, "Spreading activation models for trust propagation," in *IEEE International Conference on e-Technology, e-Commerce and e-Service, 2004. EEE '04*. 2004, 2004, pp. 83–97.
- [38] R. W. van der Heijden, H. Kopp, and F. Kargl, "Multi-source fusion operations in subjective logic," 2018. [Online]. Available: <https://arxiv.org/abs/1805.01388>
- [39] W. H. Wolberg, O. L. Mangasarian, and W. N. Street, "Breast cancer wisconsin (diagnostic) data set," [https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+\(diagnostic\)](https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(diagnostic)), 1995, uCI Machine Learning Repository.
- [40] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [41] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," 2017. [Online]. Available: <https://arxiv.org/abs/1712.05526>
- [42] A. Jøsang and L. Kaplan, "Principles of subjective networks," in *2016 19th International Conference on Information Fusion (FUSION)*, 2016, pp. 1292–1299.

APPENDIX A
SUBJECTIVE LOGIC OPERATORS AND SYMBOLS

TABLE VI: Summary of Subjective Logic Operators Used in This Work

Symbol	Name	Definition / Equation
\odot	Binomial multiplication	[5]
\otimes	Trust discounting	$\omega_X^{[A;B]} = \omega_B^A \otimes \omega_X^B$ $(b, d, u) \otimes (b', d', u') = (Pb', Pd', 1 - P(b' + d))$ $P = b + au$
\oplus	Averaging fusion	[9]
\ominus	Fusion-based revision	$T_\theta \leftarrow T_\theta \ominus T_{n Y}$ (as defined in Sec. V.B)
\oslash	Conservative combination	$(b, d, u) = (b_1, d_1, u_1) \oslash (b_2, d_2, u_2)$ $b = \min(b_1, b_2),$ $d = \max(d_1, d_2),$ $u = 1 - (b + d)$
\odot	Deduction	[5], [42]

APPENDIX B
PATAS OPERATIONS

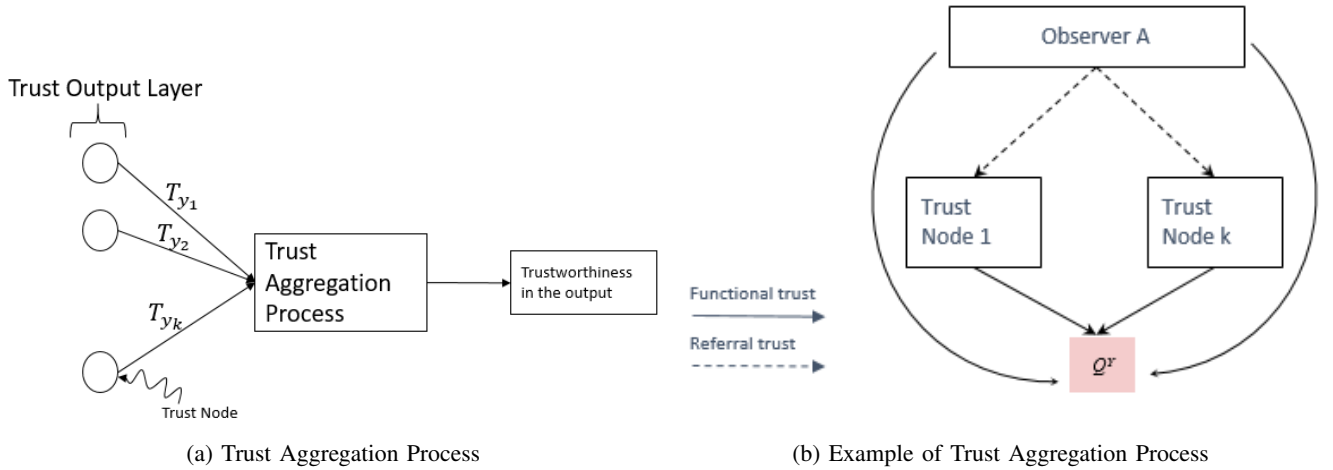


Fig. 8: Output-Trust Aggregation

Fig. 8a illustrates this process. Each Trust Node in the output layer generates a trust score $T_{y_1}, T_{y_2}, \dots, T_{y_k}$, corresponding to the network's output components. These values are then passed to an aggregation mechanism that produces a unified output trust value T_y , formally defined as:

$$T_y = \text{Agg}((T_{y_j})_{j \in [1, k]})$$

where Agg is an aggregation function based on the reasoning operations defined in the PaTAS.

Fig. 8b presents a conceptual example in which a virtual observer A receives individual trust from each output neuron. Observer A integrates these trust scores using the aggregation function to arrive at a single opinion on the network's overall prediction trustworthiness.

The aggregation mechanism can be adapted to different application requirements, such as majority fusion, weighted fusion, or uncertainty-sensitive operators, depending on how the output trust should reflect the underlying trust semantics.

APPENDIX C
THEOREMS PROOFS

A. Proof for Theorem 1

Proof. Let us define a distance d on the space $\Omega \subseteq [0, 1]^4$, based on the Euclidean norm (2-norm).

Assume that the recursive update is expressed as:

$$\omega_{n+1} = \omega_n \ominus q, q \in \Omega$$

where \ominus denotes a fusion operator (e.g., cumulative or averaging fusion) that combines opinions ω_n and q . By the properties of subjective logic fusion, the result of this operation satisfies one of the following:

- 1) $d(\omega_{n+1}, q) < d(\omega_n, q)$, i.e., the new opinion is strictly closer to q , or
- 2) $d(\omega_{n+1}, q) = d(\omega_n, q)$ and $\omega_{n+1} = \omega_n$, meaning the sequence has reached a fixed point.

This behavior reflects the nature of fusion operators, which are designed to generate an opinion that represents a consistent aggregation of the two inputs.

In case 1, the distance to q strictly decreases at each step. Since the 2-norm is bounded in $[0, 1]^4$, the sequence (ω_n) is contained in a compact space and forms a Cauchy sequence. Therefore, it converges to the unique fixed point q .

In case 2, where $\omega_{n+1} = \omega_n$, the sequence remains constant and equal to ω_0 . This occurs, for instance, when $q = (0, 0, 1, a_0)$, representing full uncertainty. In that case, most of the fusion operators (almost all except averaging fusion) has no effect, and the sequence stays fixed.

Thus, in both cases, the sequence (ω_n) converges. \square

B. Proof for Theorem 2

Proof. We analyze each component of ω_n separately.

1. *Convergence of a_n :* By definition, $a_{n+1} = a_n a_q$. Since $a_q \in [0, 1]$, this is a geometric sequence:

$$a_n = a_0 a_q^n \rightarrow 0 \quad \text{as } n \rightarrow \infty.$$

2. *Convergence of d_n :* The recurrence relation is:

$$d_{n+1} = d_n + d_q - d_n d_q = d_n(1 - d_q) + d_q.$$

This is a first-order linear recurrence. If $d_q > 0$, the sequence is increasing and bounded above by 1. Therefore:

$$\lim_{n \rightarrow \infty} d_n = 1 \text{ (the fixed point)}$$

If $d_q = 0$, then $d_{n+1} = d_n = d_0$ for all n .

3. *Behavior of b_n :* The update equation for b_{n+1} is rational function involving a_n , b_n , and u_n . As $a_n \rightarrow 0$, the update expressions simplify:

$$b_{n+1} \approx b_n b_q + a_q b_n u_q = b_n(b_q + a_q u_q) = b_n p_q$$

this converges since the projected probability $p_q \in [0, 1]$

To conclude with u_n , since we have $u_n = 1 - (b_n + d_n)$ it will also converge \square

C. Proof for Theorem 4

Let $\omega^\emptyset = (0, 0, 1, a)$ denote a vacuous binomial opinion over any variable, with arbitrary base rate $a \in [0, 1]$. Then the following two properties hold:

- 1) *Discounting a Vacuous Opinion Yields a Vacuous Opinion.*

For any trust value binomial opinion ω_B^A , the discounted opinion

$$\omega_X^{[A;B]} = \omega_B^A \otimes (0, 0, 1, a) = (0, 0, 1, a)$$

Proof. Using the trust discounting operator from Subjective Logic, we set P to the projected probability of ω_B^A :

$$\omega_X^{[A;B]} = \begin{cases} b_X^{[A;B]} = P \cdot 0 = 0, \\ d_X^{[A;B]} = P \cdot 0 = 0, \\ u_X^{[A;B]} = 1 - b_X^{[A;B]} - d_X^{[A;B]} = 1, \\ a_X^{[A;B]} = a. \end{cases}$$

Hence, $\omega_X^{[A;B]} = (0, 0, 1, a)$.

- 2) *PaTAS Feedforward on Vacuous Input Yields Vacuous Output.*

Let $T_x = (0, 0, 1, a)$ be the trust assessment of an input to PaTAS. Then for any parameter trust configuration T_θ , the output trust assessment satisfies:

$$T_y = \text{IPTA}(T_x) = (0, 0, 1, a).$$

Proof. The PaTAS feedforward mechanism computes for each neuron:

$$T_z = \bigvee_i (T_{x_i} \otimes T_{\theta_i}).$$

Since each $T_{x_i} = (0, 0, 1, a)$, and using the result from part (1), we get:

$$T_{x_i} \otimes T_{\theta_i} = (0, 0, 1, a), \quad \forall i.$$

Then, by fusion of vacuous opinions:

$$T_z = \bigvee_i (0, 0, 1, a) = (0, 0, 1, a).$$

This holds recursively through all layers of PaTAS, including the output layer, hence:

$$T_y = (0, 0, 1, a).$$

D. Proof for Theorem 5

Proof. We prove the theorem in two steps.

(1) Symmetry of the Discount Operator:

Let $\omega_\theta = (b_\theta, d_\theta, u_\theta)$ be a binomial opinion representing the trust weight associated with a connection in the PaTAS. Let P denote the projected probability of ω_θ , defined as:

$$P = b_\theta + au_\theta,$$

where a is the base rate (typically $a = 0.5$ for binary domains).

Let $x = (b, d, u)$ be any binomial opinion and $\bar{x} = (d, b, u)$ its symmetric counterpart. Then the trust discounting operation yields:

$$\omega_\theta \otimes x = (P \cdot b, P \cdot d, 1 - P \cdot (b + d)),$$

$$\omega_\theta \otimes \bar{x} = (P \cdot d, P \cdot b, 1 - P \cdot (b + d)).$$

Since $b + d = 1 - u$, both discounted opinions have identical uncertainty and symmetric belief/disbelief masses. Hence, $\omega_\theta \otimes x$ and $\omega_\theta \otimes \bar{x}$ are symmetric.

(2) Symmetry Preservation under Fusion:

Let $\mathcal{X} = \{x_1, \dots, x_n\}$ be a set of discounted opinions resulting from symmetric inputs, and let $\bar{\mathcal{X}} = \{\bar{x}_1, \dots, \bar{x}_n\}$ be their symmetric counterparts. Consider any symmetric fusion operator \bigoplus (such as averaging, cumulative fusion, or consensus fusion in Subjective Logic) applied to \mathcal{X} and $\bar{\mathcal{X}}$.

Since each pair (x_i, \bar{x}_i) is symmetric and the operator treats belief and disbelief symmetrically, we have:

$$\bigoplus_{i=1}^n x_i = (b', d', u') \quad \Rightarrow \quad \bigoplus_{i=1}^n \bar{x}_i = (d', b', u').$$

Thus, the output of the PaTAS feedforward inference remains symmetric when symmetric inputs are provided.

(3) Invariance under Fusion:

A fundamental property of Subjective Logic fusion operators is that if all input opinions assign the same value to a specific mass (e.g., belief or disbelief), the result will preserve that value. In particular, if all input opinions have belief mass equal to zero, the fused result will also have belief mass zero. The same applies to disbelief mass.

Therefore, if the discounting step yields discounted opinions with zero belief (or zero disbelief) across all components, then the fusion stage will preserve that zero mass in the final output.

(4) Fully Trusted and Distrusted Cases:

For $x = (1, 0, 0)$ (fully trusted), we have:

$$\omega_\theta \otimes x = (P, 0, 1 - P),$$

so the discounted opinion assigns zero disbelief. As noted above, the fusion of such discounted opinions will also assign zero disbelief.

For $\bar{x} = (0, 1, 0)$ (fully distrusted), we have:

$$\omega_\theta \otimes \bar{x} = (0, P, 1 - P),$$

so the discounted opinion assigns zero belief. Therefore, the belief from a fully distrusted input is always zero in the PaTAS output. \square

APPENDIX D DETAILED RESULTS

All the results are depicted in Figs. 9 and 11 to 19.

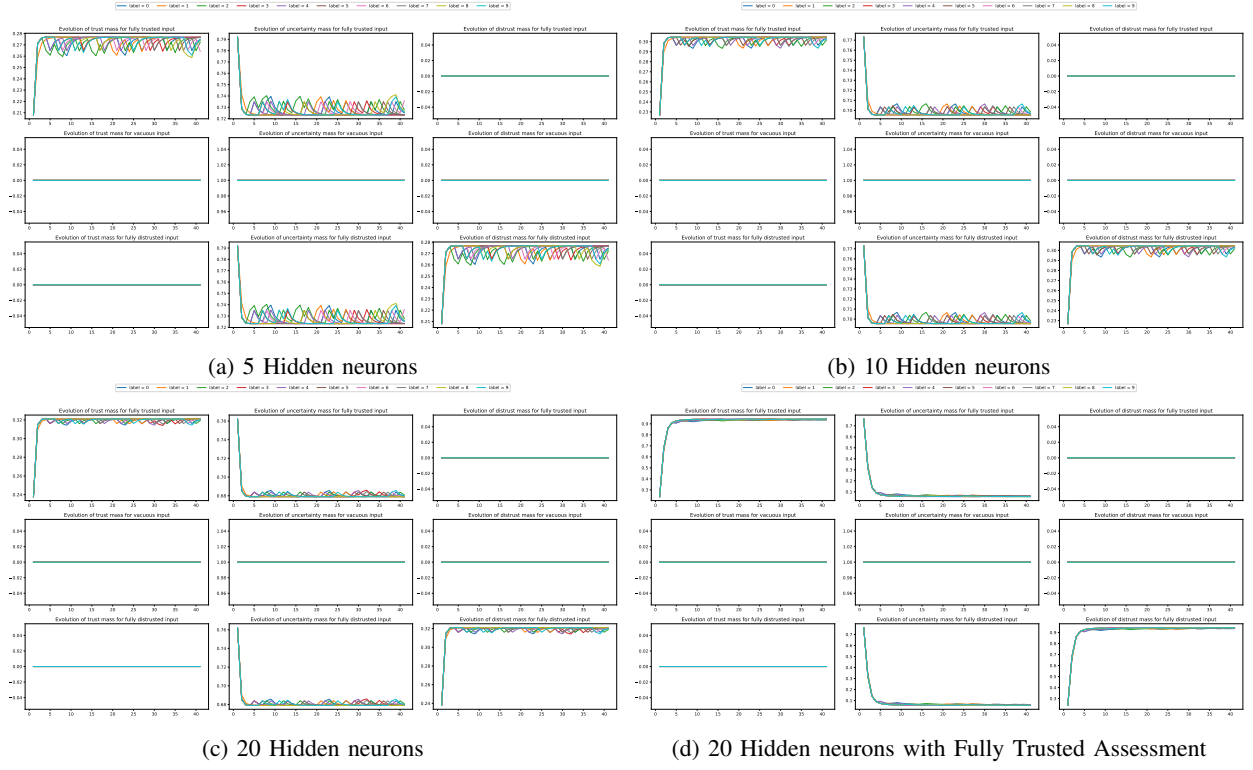


Fig. 9: MNIST with Vacuous Trust Assessment

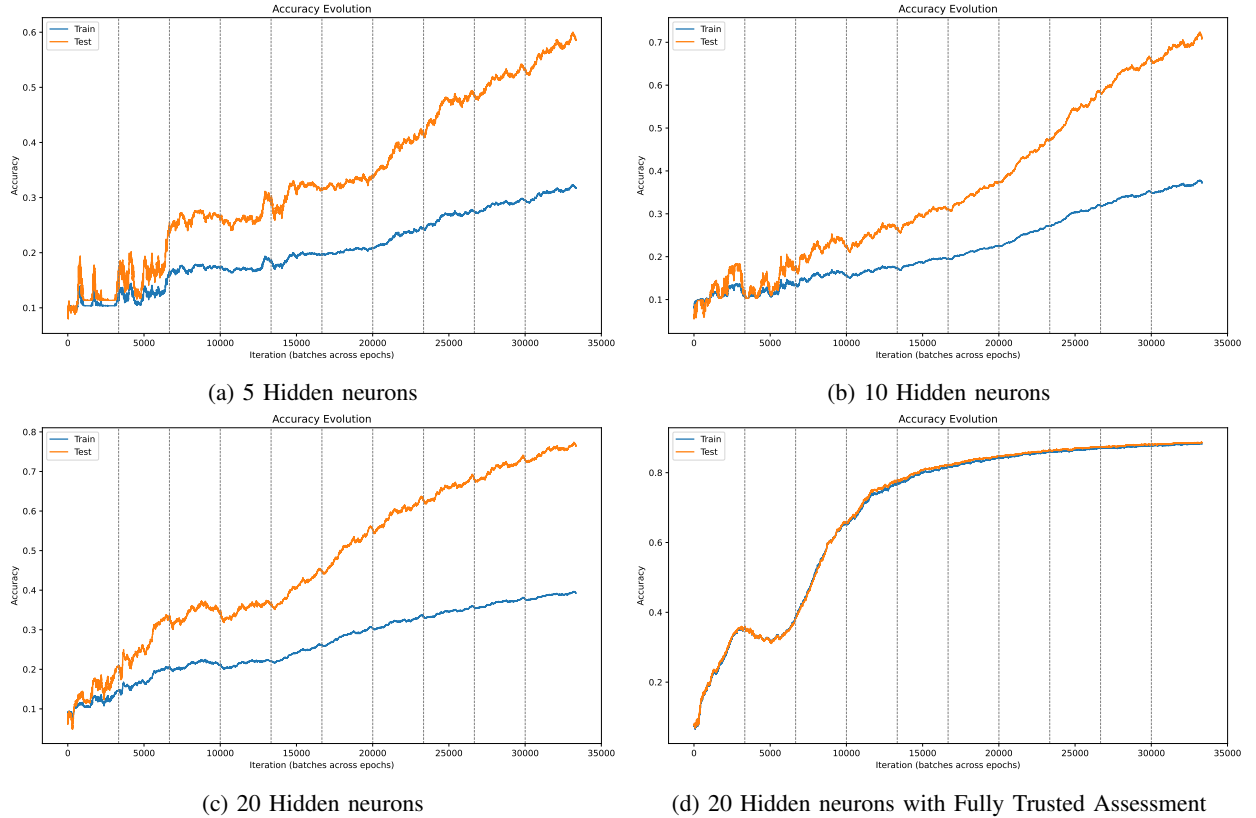


Fig. 10: MNIST with Vacuous Trust Assessment

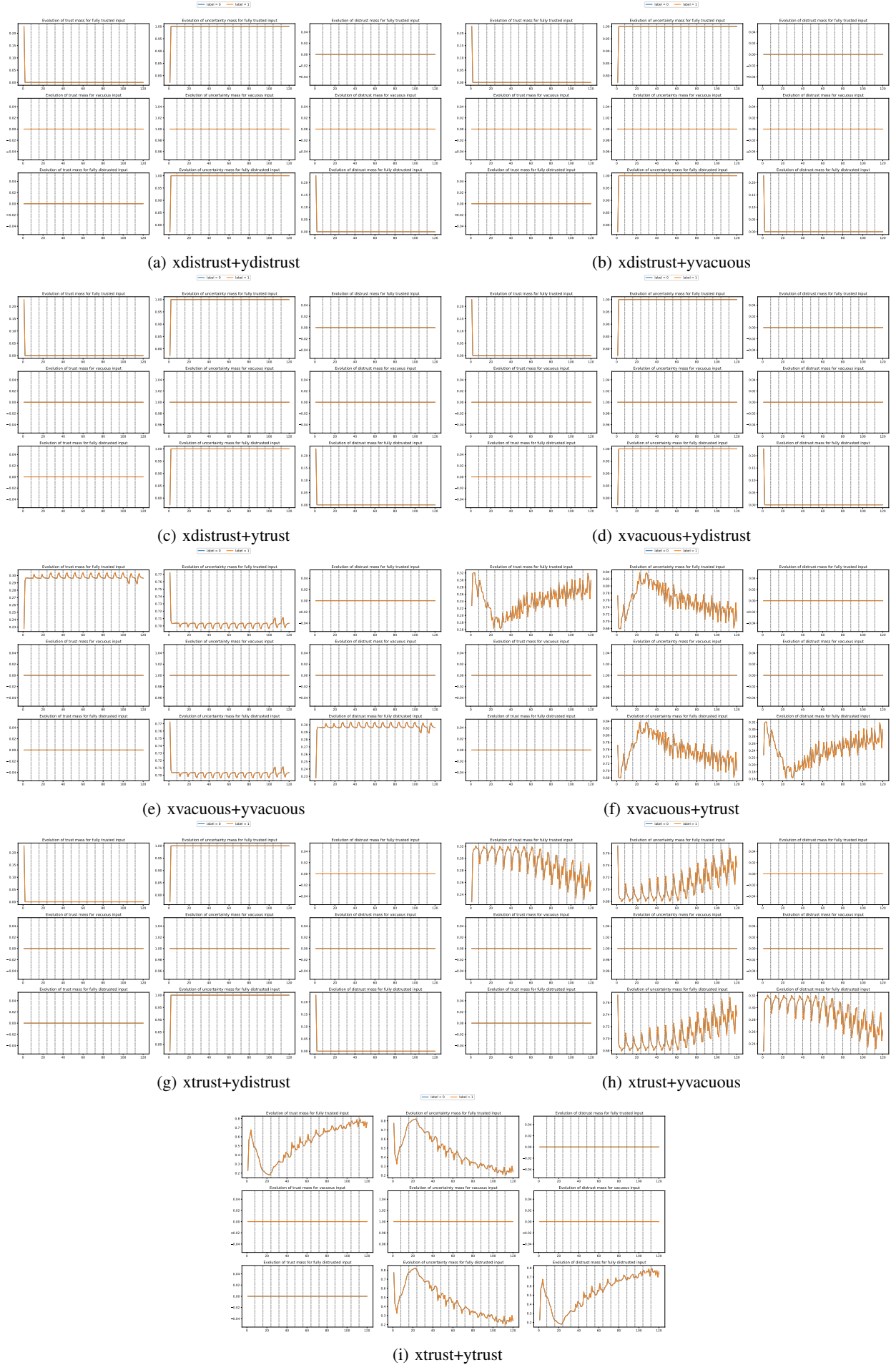
Fig. 11: Cancer Model with $\epsilon = 0.01$



Fig. 12: Cancer Model with epsilon = 0.1

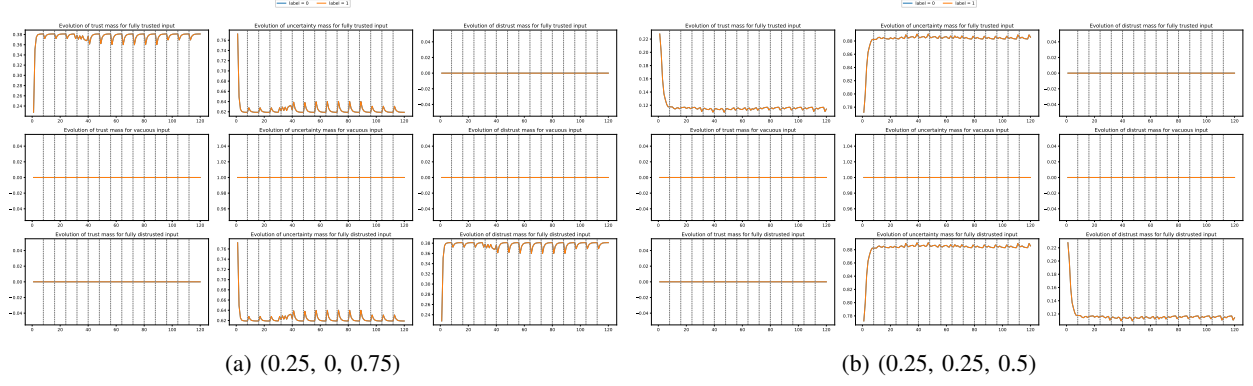


Fig. 13: Degradation

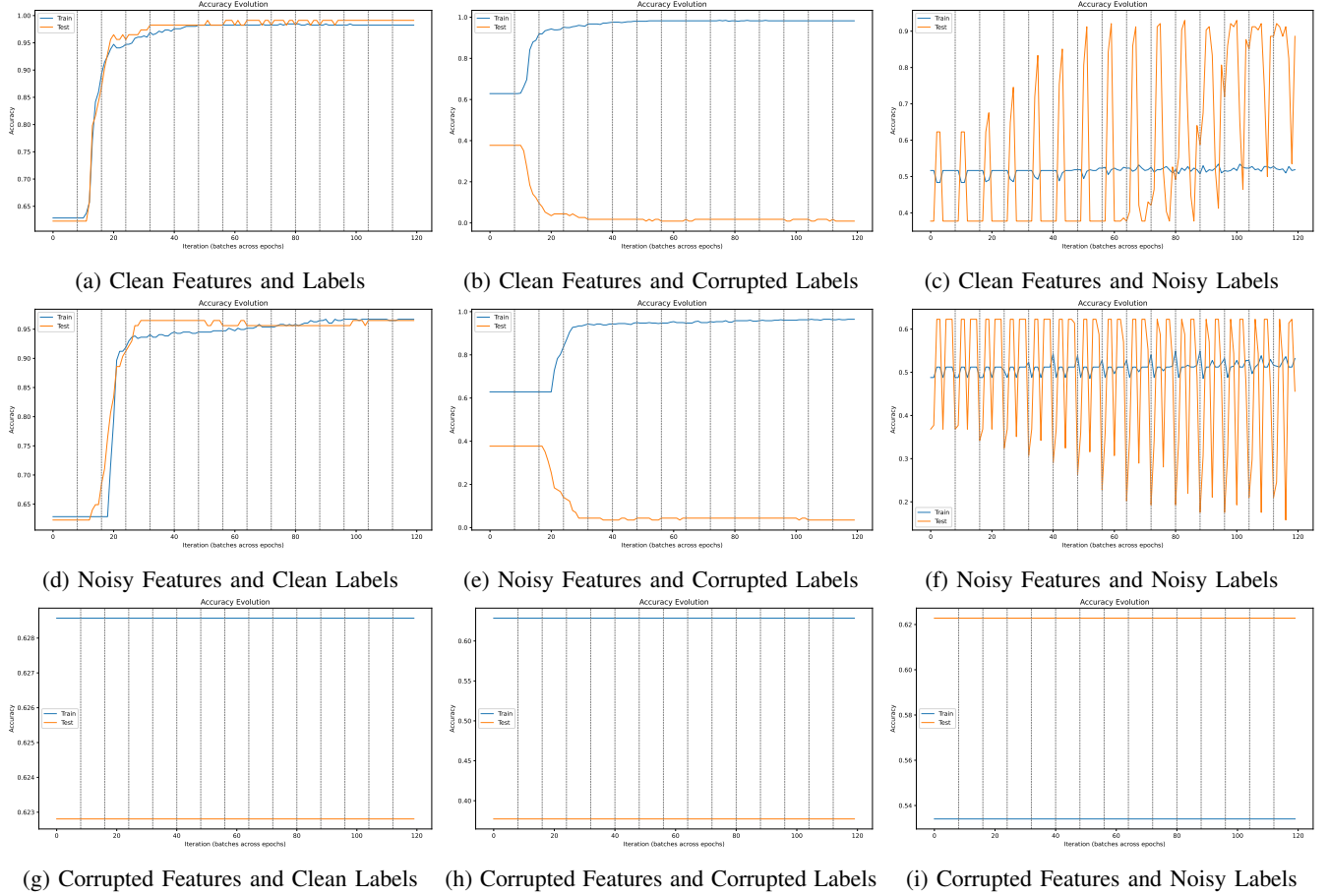


Fig. 14: Accuracy evolution of the Cancer model under different combinations of clean, corrupted, and noisy features and labels.

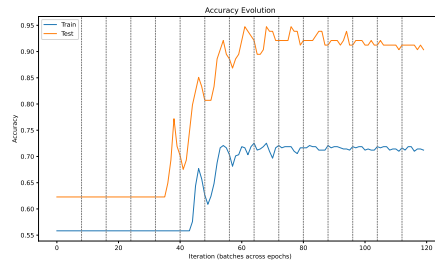


Fig. 15: Accuracy evolution of the Cancer model when reducing the noise for noised features and labels.



Fig. 16: MNIST poisoned 20 hidden neurons

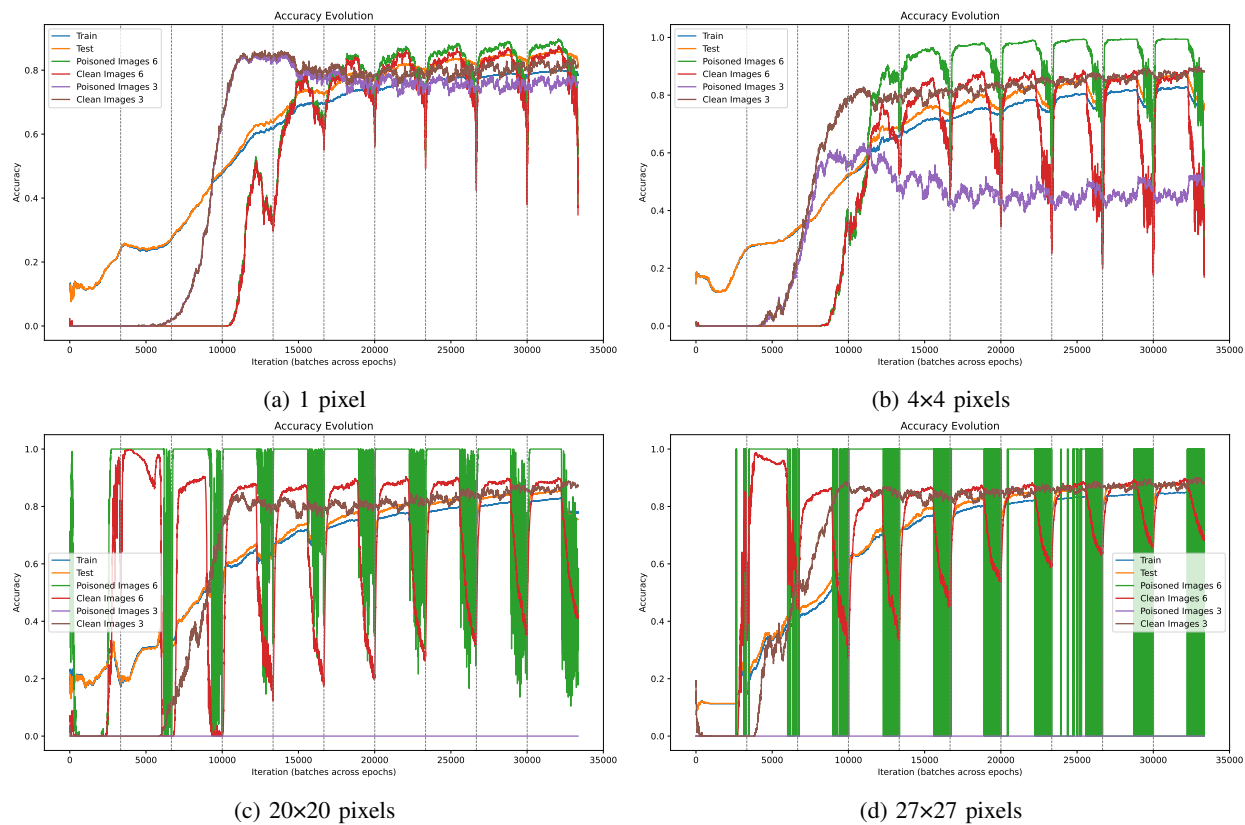
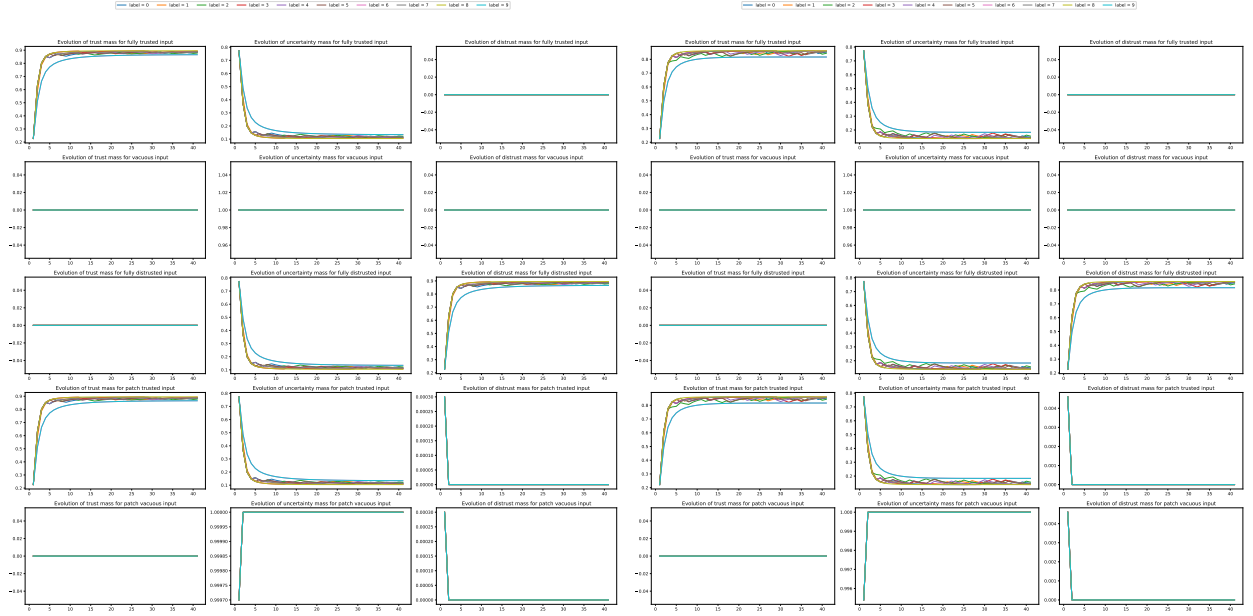
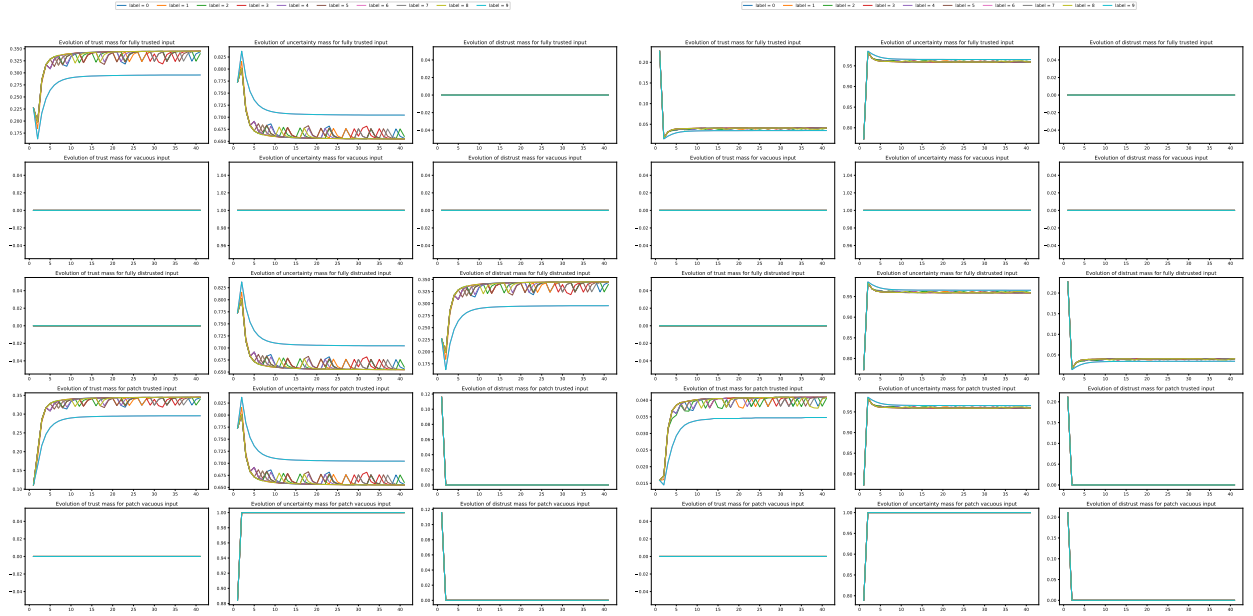


Fig. 17: Accuracy for poisoned MNIST model with 20 hidden neurons



(a) 1 pixel

(b) 4x4 pixels



(c) 20x20 pixels

(d) 27x27 pixels

Fig. 18: MNIST poisoned 10 hidden neurons

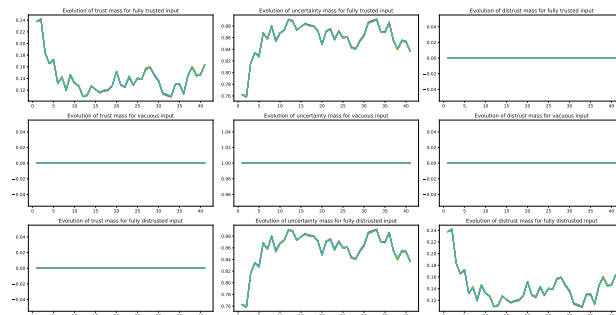


Fig. 19: MNIST randomized trust