Towards Reliable Image Classification:
A Systematic Robustness Analysis of CNN and Classical Models Under Natural Corruption

Nahid Aghababaeyan

A Thesis

in

The Department

of

Mathematics and Statistics

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Mathematics (Data Science) at

Concordia University

Montreal, Quebec, Canada

August 2025

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: Nahid Aghababaeyan

Entitled: Towards Reliable Image Classification:

A Systematic Robustness Analysis of CNN and Classical Models Under Natural Corruption

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Mathematics (Data Science)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Examiner
Prof. Junxi Zhang

_____ Thesis Supervisor(s)
Prof. Arusharka Sen

Approved by

_____
Prof. Cody Hyndman, Graduate Program Director

August 14 2025
Date of Defence

_____
Prof. Pascale Sicotte, Dean of Faculty of Arts and Science

Abstract

Towards Reliable Image Classification: A Systematic Robustness Analysis of CNN and Classical Models Under Natural Corruptions

by Nahid Aghababaeyan

Machine learning models, particularly Convolutional Neural Networks (CNNs), dominate image classification tasks in critical domains such as medical imaging, autonomous driving, and insurance. However, despite high accuracy on clean benchmark datasets, these models often exhibit significant performance degradation under real-world corruptions like noise, blur, occlusion, or compression artifacts, leading to safety risks and operational failures. Existing robustness evaluations remain limited, focusing predominantly on deep neural networks, using narrow accuracy-based metrics, and overlooking classical machine learning approaches, uncertainty quantification, prediction stability, and computational efficiency.

This thesis presents a comprehensive evaluation of seven model families—ranging from classical (Logistic Regression, SVM, K-NN, Random Forest, MLP) to deep learning (Lenet-5, ResNet-18)—on MNIST and Fashion-MNIST. We propose a unified, multi-metric framework assessing accuracy, robustness (flip rate, label variation), uncertainty (Gini index, max probability), and efficiency (parameter count, training time) under clean, corrupted, and mixed-noise conditions.

Our findings offer practical insights into model reliability and highlight the trade-offs between performance, stability, and computational cost—supporting more informed choices in real-world deployments.

Acknowledgments

This thesis is more than an academic milestone — it is a testament to resilience through some of the most difficult moments of my life. Along the way, there were times marked by profound emotional difficulty and personal hardship, when the path felt uncertain and the weight of it all nearly unbearable. Yet not once did I feel like giving up, because of the people who believed in me when I couldn't believe in myself.

To my supervisor, Prof. Arusharka Sen, I truly do not have the words to fully express my gratitude. After a difficult transition in supervision, Prof. Sen welcomed me with open arms, kindness, and steady support. He offered not only academic guidance but also a sense of safety and trust that allowed me to rebuild my confidence. His belief in me during a time when I was at my lowest gave me the courage to keep going, and I will carry his kindness and mentorship with me always.

To my sister, Dr. Zohreh Aghababaeian, your unwavering support, wisdom, and constant encouragement were my guiding light. Despite the demands of your own life and career, you were always by my side, helping, listening, and carrying the weight of this journey with me. I truly cannot imagine how this would have been possible without your presence and guidance.

My deepest appreciation also goes to my family and friends, whose patience, understanding, and belief in me provided the emotional strength I needed to persevere. Their presence, whether near or far, has been a quiet yet powerful force behind my achievements.

I would also like to thank the faculty and staff of the Department of Mathematics and Statistics at Concordia University for providing a stimulating and supportive environment for learning and research. Their assistance and resources have been invaluable to my academic growth.

Finally, to everyone who stood by me — thank you deeply. This journey was tough, and your kindness and support kept me going. I could not have done it without you.

Thank you all for making this experience both enriching and rewarding.

Contents

List of Figures

List of Tables

1. *Introduction*

1.1 Motivation

Machine learning (ML) models, ranging from classical statistical methods like logistic regression and random forests to deep neural networks (DNNs), have become central to image classification tasks in diverse domains, including medical imaging, autonomous driving, security surveillance, and insurance claim processing. In recent years, convolutional neural networks (CNNs), such as ResNet (He et al. (2016)), have emerged as particularly dominant approaches due to their ability to achieve superior predictive performance, typically measured by accuracy, precision, and recall.

However, despite their strong pre-deployment performance, these models often struggle under real-world conditions. For instance, CNN-based medical diagnostic systems have misclassified MRI scans affected by artifacts like motion blur, Gaussian noise, or low resolution, risking incorrect diagnoses (Finlayson et al. (2019)). Autonomous vehicles using CNNs have similarly failed to recognize partially occluded or weather-affected road signs, leading to safety hazards (Michaelis et al. (2019)). In insurance, CNNs trained on clean images have inaccurately assessed vehicle damage claims when encountering realistic distortions such as glare, blur, or JPEG compression artifacts (Hendrycks et al. (2021)).

These examples highlight the critical need for robustness testing before deploying image classification models in real-world scenarios. Robustness testing evaluates how models perform under realistic perturbations and noisy conditions, helping practitioners identify weaknesses that are not visible through traditional accuracy-based evaluations alone. Ro-

bustness is also important during the model selection phase. For instance, a model with 90% accuracy on clean data but a 20% performance drop under noise may be less practical and reliable than a model with 85% clean-data accuracy but only a 5% drop under similar conditions.

This motivates the need for comprehensive robustness testing that extends beyond conventional accuracy metrics. Robust evaluation must capture not only a model's performance under corruption, but also its ability to estimate uncertainty, maintain prediction consistency, and remain well-calibrated in noisy, real-world settings. Incorporating such evaluations in the pre-deployment phase enables practitioners to detect models with high variance or overconfidence in out-of-distribution regimes, and to select training procedures that balance predictive performance with robustness. Ultimately, this approach ensures the deployment of models that are not only accurate in ideal conditions, but also resilient under stress, reducing the risk of costly failures and improving the trustworthiness and reliability of automated decision-making systems.

1.2 Problem Statement and Research Objectives

Although model robustness has received increasing attention in recent years, much of the existing literature remains heavily focused on deep neural networks—particularly convolutional architectures—and evaluates performance primarily on clean, well-curated datasets using accuracy as the dominant metric (Dosovitskiy et al. (2020), Tan and Le (2019), Zhang et al. (2020)). However, high accuracy on ideal inputs does not imply reliable performance in real-world scenarios, where inputs are often degraded by corruption types such as noise, occlusion, or compression artifacts.

Moreover, prior robustness studies (Hendrycks and Dietterich (2019), Moosavi-Dezfooli et al. (2017), Tsipras et al. (2018)) frequently neglect key dimensions of model reliability, including uncertainty calibration, prediction stability, and computational efficiency—all of which are critical in safety-sensitive or resource-limited environments. The robustness land-

scape explored in these studies is also narrow in scope, with an emphasis on modern deep learning architectures, while classical machine learning models remain underexplored despite their potential to offer competitive robustness under certain conditions.

To address these limitations, this study undertakes a comprehensive empirical evaluation of seven diverse model families including both calssical ML and deep learning models—Logistic Regression, Support Vector Machine (SVM), k-Nearest Neighbors (K-NN), Random Forest, Multi-layer Perceptron (MLP), Lenet-5, and ResNet-18—on the MNIST and Fashion-MNIST datasets. We assess robustness across three scenarios: clean inputs, ten distinct types of individual corruption, and a mixed-noise setting that simulates more realistic perturbation patterns. This design enables a systematic investigation into how different model classes respond to varied degradation conditions.

To address gaps in existing robustness research, we conducted a large-scale empirical study evaluating both classical and deep learning models. Our study included seven diverse architectures: Logistic Regression, Support Vector Machine (SVM), k-Nearest Neighbors (K-NN), Random Forest, Multi-layer Perceptron (MLP), Lenet-5, and ResNet-18. We applied 11 distinct corruption types individually to the test sets of both MNIST and Fashion-MNIST, alongside the clean test set, resulting in 12 evaluation scenarios per dataset—for a total of 24 distinct evaluation sets.

Our evaluation framework goes well beyond conventional performance measures, offering a rich and multi-dimensional assessment of model behavior under corruption. In addition to standard performance measures like accuracy, precision, recall, and F1-score, we incorporate a broad spectrum of robustness-specific and reliability-driven metrics. These include robustness indicators such as Flip Rate and Label Variation, uncertainty quantification metrics like the Gini Index and Maximum Predicted Probability, and computational efficiency measures including parameter count and training time.

By integrating this broad and rigorous set of evaluation metrics, our framework enables comprehensive, side-by-side comparisons of models across not only predictive performance, but also robustness, uncertainty, reliability, and efficiency—dimensions often underexplored

in prior robustness research.

The key contribution of this work is its multi-dimensional analysis of classical and deep learning models under both clean and corrupted conditions. Unlike prior studies that isolate accuracy or robustness, our study offers a statistically grounded benchmark that jointly considers performance degradation, uncertainty calibration, prediction stability, and computational cost. To the best of our knowledge, this is the first empirical study to systematically evaluate such a wide range of model architectures—spanning both traditional and modern approaches—using a diverse set of metrics across clean, corrupted, and mixed-noise datasets.

1.3 Contributions

This thesis presents a comprehensive empirical investigation into robustness evaluation for image classification—bridging classical machine learning and deep learning paradigms in a unified, reproducible framework. Our work pushes the boundaries of existing robustness literature through the following key contributions:

- **A Multi-Dimensional Evaluation Framework:** We propose a novel and rigorous evaluation protocol that goes far beyond standard accuracy-based assessments. Our framework integrates a diverse suite of metrics covering predictive performance (accuracy, F1-score), robustness and stability (Flip Rate, Label Variation), uncertainty quantification (Gini Index, maximum predicted probability), and computational efficiency (training time, parameter count). This enables deep, side-by-side comparisons of models from multiple angles, offering a more complete understanding of model behavior under corruption.

- **Robustness Analysis Across 24 Evaluation Sets:** By applying 11 corruption types individually, along with a mixed-noise scenario, we construct 12 evaluation sets per dataset—totaling 24 distinct test environments. This setup allows us to uncover granular, model-specific degradation patterns and robustness vulnerabilities, produc-

ing actionable insights for selecting models in real-world, safety-critical, or resource-constrained deployments.

- **Open and Reproducible Benchmarking Toolkit:** To support transparency and facilitate future research, we release a full benchmarking toolkit, including modules for corruption generation, metric computation, and efficiency profiling. This ensures that our evaluation pipeline can be reused, extended, and built upon by the broader research community.

  Together, these contributions represent a significant step forward in how robustness is measured, compared, and understood, especially in contexts where model trustworthiness and operational stability are non-negotiable.

2. *Related Work*

2.1 Traditional ML and CNNs in Image Classification

Traditional machine learning (ML) models have historically been used for image classification, particularly before the rise of deep learning. Algorithms such as logistic regression, support vector machines (SVM), k-nearest neighbors (K-NN), random forests, and multilayer perceptrons (MLPs) have demonstrated varying levels of success on relatively simple or low-dimensional visual datasets.

Logistic regression, while limited by its linearity, has been applied to image classification with reasonable success when combined with dimensionality reduction techniques such as principal component analysis (Bishop (2006)). SVMs, due to their ability to model nonlinear decision boundaries through kernel methods, have shown improved performance on moderate-sized image datasets (Cortes and Vapnik (1995)). K-NN classifiers, though conceptually simple, can achieve reasonable accuracy using distance metrics in feature space, but they scale poorly and are heavily reliant on feature quality (Cover and Hart (1967)). Random forests offer improved performance over single decision trees by aggregating predictions across multiple learners, reducing overfitting and increasing robustness (Breiman (2001)). However, they require manually designed features and cannot automatically extract spatial patterns from raw image data. MLPs, as early neural network models, possess the theoretical capability to approximate complex functions but are constrained in image-related tasks due to their dense connectivity, which fails to exploit spatial structure (LeCun et al. (1998)).

Overall, traditional models often perform well on structured, low-dimensional data but

struggle with high-dimensional images, primarily because they rely on handcrafted or flattened features that ignore spatial hierarchies (LeCun et al. (2015)). These limitations motivated the transition to deep learning models, particularly convolutional neural networks (CNNs).

CNNs are specifically designed for grid-like data such as images, using convolutional layers to learn spatial hierarchies and localized patterns like edges and textures (LeCun et al. (2015)). Lenet-5, one of the earliest CNN architectures, demonstrated the effectiveness of feature learning directly from raw pixels in the task of digit recognition (LeCun et al. (1998)). Subsequent advancements led to deeper architectures such as ResNet-18 (He et al. (2016)), which introduced residual connections to mitigate vanishing gradients and allow for more effective training of deep networks.

Empirical studies consistently show that CNNs outperform traditional ML models on standard image classification benchmarks. For instance, CNNs such as ResNet achieve significantly higher accuracy on the CIFAR-10 dataset than SVMs and random forests, even when those models use advanced feature engineering (He et al. (2016), Krizhevsky (2009)). Zhang et al. (Zhang et al. (2017)) observed that CNNs not only generalize better on complex image datasets but also exhibit a degree of robustness to minor transformations, unlike traditional methods.

This performance advantage largely stems from CNNs' ability to perform representation learning: extracting increasingly abstract features directly from data without manual intervention. Additionally, transfer learning—fine-tuning models pretrained on large datasets such as ImageNet—further boosts CNNs' performance on domain-specific tasks, often surpassing traditional methods even with limited training data (Yosinski et al. (2014)).

2.2 Robustness in Image Classification

Robustness in image classification is defined as a model's capacity to maintain reliable predictions under various forms of input perturbations, such as noise, blur, brightness changes, or

adversarial attacks. This quality is essential for deploying ML models in real-world settings, where inputs frequently deviate from ideal, clean conditions (Gilmer et al. (2019)).

Szegedy et al. (Szegedy et al. (2014)) first revealed that deep networks are highly sensitive to small, imperceptible adversarial perturbations, raising concerns about their reliability. Expanding on this, Hendrycks and Dietterich (Hendrycks and Dietterich (2019)) introduced the ImageNet-C benchmark, which evaluates performance under 15 realistic corruption types. Their results showed that models like ResNet-50, while achieving high accuracy on clean data, suffer from substantial performance degradation—up to 30% accuracy loss—under corruptions.

To address these issues, various strategies have emerged. Data augmentation techniques like AutoAugment (Cubuk et al. (2019)) and AugMix (Hendrycks et al. (2020b)) enhance generalization by introducing diverse perturbations during training. Adversarial training, where models are trained on adversarial examples, increases robustness to both attacks and natural noise (Madry et al. (2018)). However, Tsipras et al. (Tsipras et al. (2019)) highlighted a robustness-accuracy trade-off, showing that enhancing robustness may lead to reduced clean accuracy.

Beyond training techniques, architectural innovations also contribute to robustness. Hendrycks et al. (Hendrycks et al. (2020a)) demonstrated that pretraining on large, diverse datasets improves both clean and corrupted accuracy. Other methods incorporate feature denoising modules (e.g., wavelet layers or DnCNN blocks) to mitigate the impact of irrelevant noise (Xie et al. (2019)).

Despite these advances, robustness remains an open challenge, especially for safety-critical systems. The continued gap between clean performance and real-world reliability emphasizes the need for systematic evaluation methods and a better understanding of robustness trade-offs.

2.3 Systematic Evaluation of Robustness, Uncertainty, and Computational Demands

Benchmarking ML models for image classification must go beyond accuracy on clean data and consider broader criteria such as robustness, uncertainty calibration, and computational efficiency. These factors are essential for evaluating models in high-stakes environments like medical imaging and autonomous systems. Foundational datasets such as MNIST (LeCun et al. (1998)) and Fashion-MNIST (Xiao et al. (2017)) remain important for evaluating and comparing model performance. Classical models such as SVMs, logistic regression, and random forests initially demonstrated strong performance on these datasets (Ciregan et al. (2012), Wan et al. (2013)). With the introduction of CNNs—including LeNet, ResNet, and EfficientNet (He et al. (2016), Tan and Le (2019))—performance significantly improved, particularly on complex or noisy data.

However, concerns about model robustness have grown, with research by Hendrycks et al. (Hendrycks and Dietterich (2019)) and Michaelis et al. (Michaelis et al. (2019)) showing that state-of-the-art CNNs degrade significantly under natural corruptions. While comparative studies have been conducted on the robustness of traditional ML models versus CNNs on datasets like MNIST and CIFAR (Mu et al. (2019), Zhang et al. (2019)), most focus on isolated corruption types or a narrow set of models.

Uncertainty quantification is another critical dimension, especially under data shifts. Metrics such as maximum softmax probability and the Gini index are commonly used to evaluate a model's confidence (Gal and Ghahramani (2016), Lakshminarayanan et al. (2017)). Although research has increasingly investigated uncertainty in deep models (Malinin and Gales (2020), Ovadia et al. (2019)), few works systematically include traditional ML models in these comparisons, particularly under corruptions.

Moreover, computational demands—such as training time, inference latency, and memory usage—are often overlooked in evaluations of robustness and uncertainty. Yet, these

considerations are vital for real-world deployments, especially on resource-constrained devices.

This thesis contributes to the literature by offering a unified, systematic evaluation of five classical models and two CNNs across 11 natural corruption types. It examines their performance in terms of accuracy, robustness, predictive uncertainty, and computational cost on MNIST and Fashion-MNIST. By jointly analyzing these dimensions, this work provides a more comprehensive perspective on the practical trade-offs involved in model selection for real-world image classification tasks.

3. *Dataset*

3.1 Original Dataset

The empirical evaluation is based on two standard datasets extensively utilized in image classification research: **MNIST** and **Fashion-MNIST**, both of which are publicly available and have been widely adopted for benchmarking image classification models (LeCun et al. (2010a), Xiao et al. (2017)). These datasets are well-suited for empirical experimentation due to their standardized structure, balanced class distributions, and human-validated annotations.

3.1.1 MNIST

The MNIST (Modified National Institute of Standards and Technology) dataset (LeCun et al. (1998)) is a classical benchmark for handwritten digit recognition. It consists of 70,000 grayscale images of handwritten digits (0 through 9), where each image is of size 28×28 pixels. Pixel intensities range from 0 to 255 and are commonly normalized to the $[0, 1]$ interval during preprocessing. The dataset is split into a training set of 60,000 images and a test set of 10,000 images.

Each class (digit) is equally represented with 7,000 instances, making the dataset balanced. All labels were manually verified to ensure accuracy, making the dataset reliable for supervised learning tasks.

Formal Definition and Analytical Properties

Let $\mathbf{X} \in \mathbb{R}^{28\times 28}$ denote an image and $y \in \{0, 1, \ldots, 9\}$ its corresponding label. The dataset is formally defined as:

$$\mathcal{D} = \{(\mathbf{X}_i, y_i)\}_{i=1}^{70{,}000},$$

with training and test partitions as:

$$\mathcal{D}_{\text{train}} = \{(\mathbf{X}_i, y_i)\}_{i=1}^{60{,}000}, \quad \mathcal{D}_{\text{test}} = \{(\mathbf{X}_i, y_i)\}_{i=60{,}001}^{70{,}000}.$$

The relatively low dimensionality (784 features) and approximate *linear separability* of MNIST make it an ideal dataset for evaluating fundamental model properties. **Linear separability** refers to the ability to distinguish different classes using a linear decision boundary, which allows simple models (e.g., logistic regression, SVMs) to achieve high accuracy.

MNIST also lends itself well to detailed empirical investigations across multiple analytical dimensions. One common approach is **invariance analysis**, which evaluates a model's robustness to affine transformations such as rotation, scaling, and translation. This analysis reveals whether the model can maintain stable predictions under small geometric changes in input images. Another mode of evaluation involves **noise propagation**, where models are tested on systematically corrupted inputs. In such cases, transformations like the addition of Gaussian noise or partial occlusion are applied to the original image $\mathbf{X}$ through a corruption operator $\mathcal{T}$, yielding $\mathbf{X}_{\text{corrupt}} = \mathcal{T}(\mathbf{X})$. This allows for assessment of a model's resilience to noisy or degraded inputs. Finally, **statistical benchmarking** focuses on evaluating the consistency and reliability of model predictions across repeated trials. This includes metrics such as prediction variance, calibration of confidence scores, and test-retest reliability, providing insight into a model's generalization performance and stability.

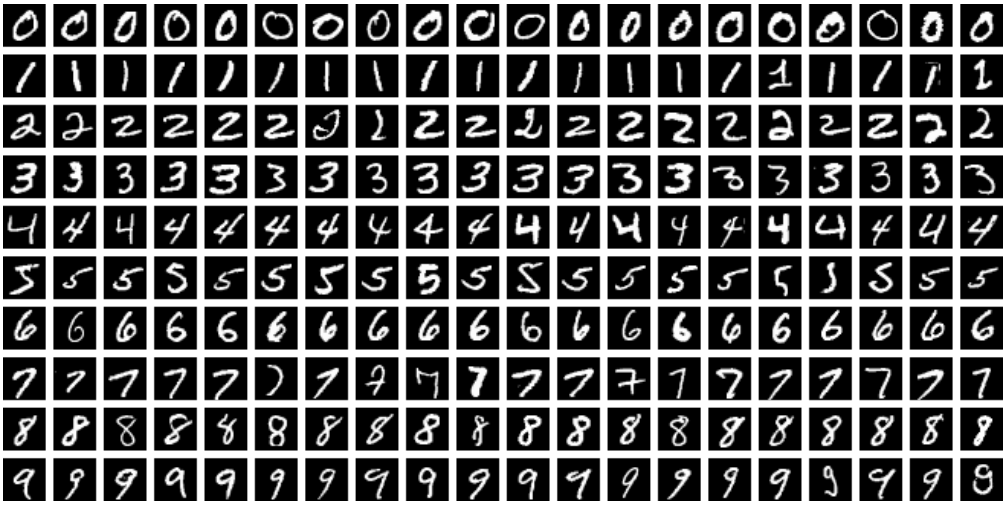


Figure 3.1: Variability in MNIST Dataset, different handwriting styles.

3.1.2 Fashion-MNIST

Fashion-MNIST is a dataset designed to serve as a more challenging alternative to the MNIST dataset (Xiao et al. (2017)). It includes 70,000 grayscale images, each sized 28×28 pixels, and is divided into 10 fashion-related classes. The dataset is partitioned into 60,000 training images and 10,000 testing images, maintaining a similar structure to the MNIST dataset.

The Fashion-MNIST dataset is composed of 70,000 black and white images, each 28×28 pixels in size. The images are labeled as follows:

0: T-shirt/top

1: Trouser

2: Pullover

3: Dress

4: Coat

5: Sandal

6: Shirt

7: Sneaker

8: Bag

9: Ankle boot

Examples of these images are shown in Figure 3.2, illustrating the variety and complexity of the clothing items. The dataset is balanced, meaning it contains an equal number of images for each of the 10 classes, ensuring that no single class is overrepresented.

Fashion-MNIST poses greater complexity and provides a more challenging benchmark compared to the original MNIST dataset. While MNIST focuses on handwritten digits, Fashion-MNIST includes a diverse range of clothing items, which introduces additional variability and complexity. This makes it an excellent resource for evaluating the performance of machine learning models in more realistic and varied scenarios.

Formal Definition and Analytical Properties

Let $\mathbf{X} \in \mathbb{R}^{28\times 28}$ denote a grayscale image and $y \in \{0, 1, \ldots, 9\}$ its corresponding label, where each label represents a clothing category. The dataset can be defined as:

$$\mathcal{D} = \{(\mathbf{X}_i, y_i)\}_{i=1}^{70{,}000},$$

with training and test sets partitioned as:

$$\mathcal{D}_{\text{train}} = \{(\mathbf{X}_i, y_i)\}_{i=1}^{60{,}000}, \quad \mathcal{D}_{\text{test}} = \{(\mathbf{X}_i, y_i)\}_{i=60{,}001}^{70{,}000}.$$

Although Fashion-MNIST shares the same dimensionality and format as MNIST, it introduces more intra-class variability and visual ambiguity. This makes it less linearly separable and more suitable for evaluating the robustness and generalization ability of machine learning models, especially in scenarios closer to real-world visual data.

Figure 3.2: Example images from the Fashion-MNIST dataset.

In summary, the Fashion-MNIST dataset is a crucial and well-known resource for advancing research in image classification.

3.2 Corrupted Dataset

Model robustness under real-world conditions requires evaluation beyond clean test data. To evaluate model robustness under realistic input degradations, we applied controlled image corruptions to the MNIST and Fashion-MNIST test datasets. These label-preserving perturbations are adapted from established benchmarks such as ImageNet-C and CIFAR-10-C

(Hendrycks and Dietterich (2019)), which simulate common real-world degradations affecting classification performance. Each corruption represents typical challenges such as sensor noise, environmental interference, image processing distortions, or data transmission errors.

We group these corruptions into three categories based on their effect on visual data: *Noise Corruptions*, *Transformation Filters*, and *Compression and Occlusion Methods*. Each category targets a specific class of visual degradation, supporting a comprehensive robustness assessment.

To determine suitable severity levels for each type of corruption, we adopted a clear and practical approach. The primary objective was to select noise levels that preserved the semantic integrity of the labels—ensuring that the true class of each sample remained recognizable to human observers. At the same time, it was essential that the corruptions be sufficiently challenging to effectively evaluate model robustness.

We avoided severe distortions that could obscure the class identity of the samples, as well as overly mild corruptions that would fail to expose potential model vulnerabilities. To identify appropriate severity levels, we closely examined multiple samples for each corruption type across a range of intensities. This process allowed us to select levels that strike a meaningful balance between human interpretability and the need for rigorous robustness evaluation.

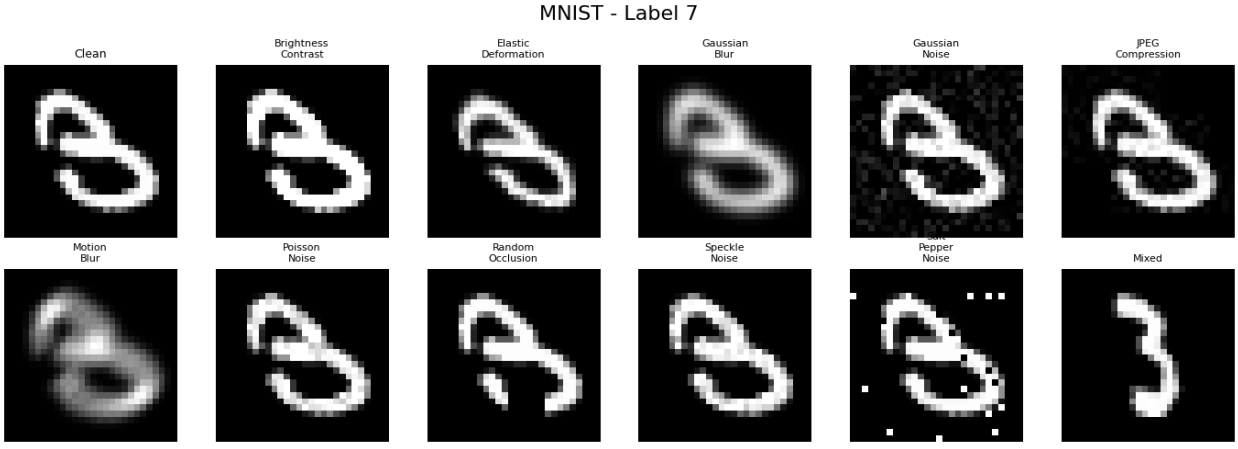


Figure 3.3: Samples of MNIST digit '3' with different types of noise applied

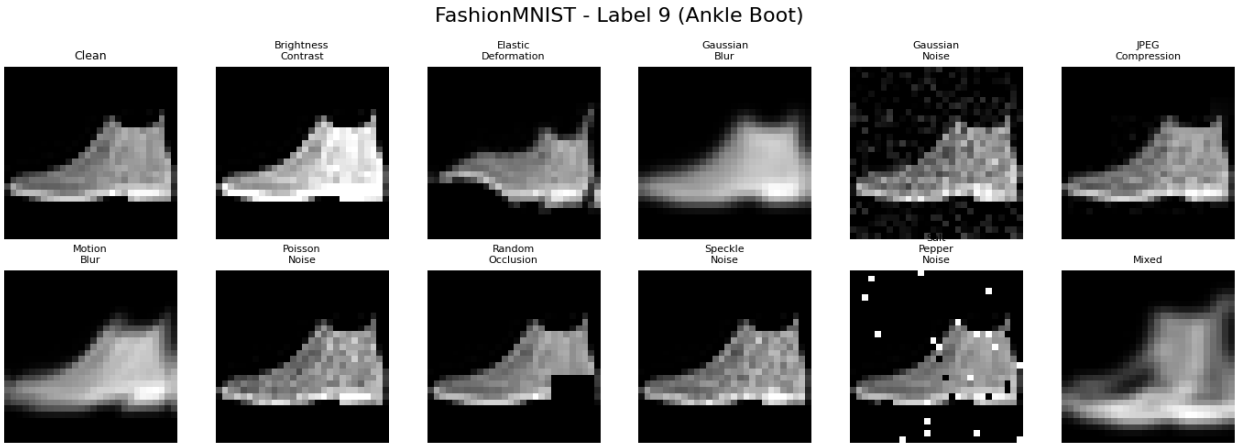


Figure 3.4: Samples of Fashion MNIST 'Ankle boot' with different types of noise applied

3.2.1 Noise Corruptions

Noise corruptions simulate stochastic disturbances introduced at the pixel level. These are often encountered in low-light imaging, defective sensors, or adverse environmental conditions:

- **Gaussian Noise**: Gaussian noise is a widely studied corruption method that introduces random fluctuations to pixel intensities, simulating imperfections commonly observed in imaging sensors. These fluctuations follow a Gaussian (normal) distribution, which was first mathematically formalized by Carl Friedrich Gauss in 1809 (Gauss (1809)). Due to its statistical properties and natural occurrence, Gaussian noise has become a standard tool in signal and image processing, and it is frequently used in the evaluation of model robustness and for test-time data augmentation (Hendrycks and Dietterich (2019)).

  Formally, the corrupted pixel value $\tilde{x}_{i,j}$ at location (i, j) is defined as

$$\tilde{x}_{i,j} = \text{clip}(x_{i,j} + \eta_{i,j},\ 0,\ 1),$$

  where $x_{i,j} \in [0, 1]$ denotes the normalized intensity of the original pixel, $\eta_{i,j} \sim \mathcal{N}(\mu, \sigma^2)$ is a random noise term drawn from a Gaussian distribution with mean μ and variance

σ^2, and the `clip` function constrains the resulting value to the valid range of $[0, 1]$. This formulation ensures that each pixel is perturbed independently, and the corrupted image remains within valid intensity bounds.

In practice, the majority of Gaussian noise values lie within three standard deviations from the mean, i.e., $\pm 3\sigma$, which captures approximately 99.7% of the probability mass. For instance, if the standard deviation is set to $\sigma = 0.1$, the perturbations typically fall within the range $[-0.3, +0.3]$ in normalized units. This range produces perceptible but bounded distortions that realistically reflect sensor noise.

In our implementation, we parameterize the Gaussian noise with a mean $\mu = 0$ and variance $\sigma^2 = 0.01$, resulting in a standard deviation of $\sigma = \sqrt{0.01} \approx 0.1$. The noise is applied independently to each pixel and scaled according to the image intensity range. Specifically, for 8-bit images, noise values are multiplied by 255 before being added to the image. The perturbed pixel values are subsequently clipped to maintain valid intensity bounds. This setup introduces a moderate level of corruption while preserving the overall structure of the image, facilitating a realistic and controlled robustness evaluation.

Figure 3.5: Visualization of Five Gaussian Noise Severities on MNIST Input

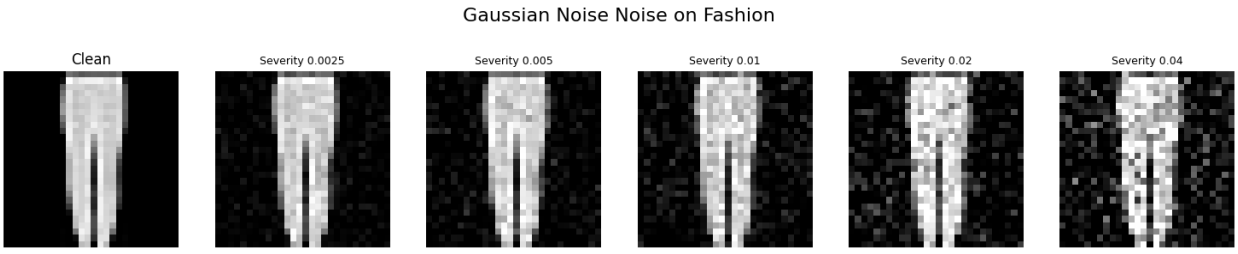


Figure 3.6: Visualization of Five Gaussian Noise Severities on Fashion MNIST Input

- **Poisson Noise**

  Poisson noise introduces noise that depends on the pixel intensity, meaning that brighter pixels experience more noise. This simulates the natural fluctuations in photon detection processes, particularly in low-light imaging. The concept of Poisson noise, as a statistical model for random events, was first proposed by Siméon-Denis Poisson (Poisson (1837)), and has been widely adopted for simulating noise in imaging systems where the signal is composed of discrete photon events (Schottky (1918)).

  Mathematically, for a given clean pixel intensity $x \in [0, 1]$, the corrupted pixel value $\tilde{x}$ is sampled from a Poisson distribution:

$$\tilde{x} \sim \frac{1}{\lambda} \cdot \text{Poisson}(\lambda x)$$

  where λ is a scaling constant that controls the total photon count, and hence the noise level. In our implementation, we follow a normalized convention in which $\lambda = 1$, thereby letting the noise variance be implicitly governed by the intensity x itself. This reflects the key property of Poisson noise: its variance equals its mean. That is,

$$\mathbb{E}[\tilde{x}] = x, \quad \text{Var}[\tilde{x}] = x$$

  To apply this noise, we first normalize the image to the $[0, 1]$ range, simulate Poisson noise using the `skimage.util.random_noise` function with the mode set to `"poisson"`,

and subsequently rescale the result to the original intensity range. This method preserves the statistical characteristics of Poisson-distributed fluctuations while maintaining consistency across all image samples. Notably, no explicit severity parameter is introduced, as the fluctuation level is inherently linked to the input intensity values.

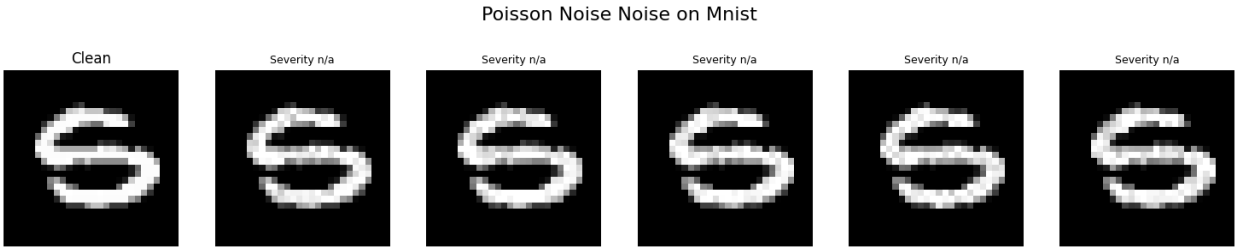


Figure 3.7: Visualization of Five Poisson Noise Severities on MNIST Input

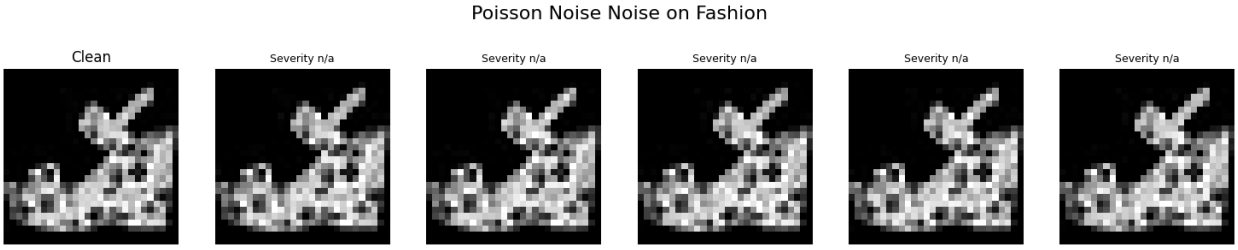


Figure 3.8: Visualization of Five Poisson Noise Severities on Fashion MNIST Input

- **Salt and Pepper Noise**

  Salt and pepper noise is a type of impulse noise that simulates abrupt disruptions in the imaging process, often caused by sensor faults, transmission errors, or faulty memory locations. This corruption randomly alters a subset of image pixels to either the minimum or maximum possible intensity, creating sharp black-and-white specks that are challenging for both human observers and machine learning models to interpret reliably (Hendrycks and Dietterich (2019)).

  Mathematically, given an image $I \in \mathbb{R}^{H \times W \times C}$, where H, W, and C represent the height, width, and number of channels respectively, salt and pepper noise produces a

corrupted image $\tilde{I}$ as follows:

$$\tilde{I}(x,y,c) = \begin{cases} 0, & \text{with probability } \frac{p}{2} \\ 255, & \text{with probability } \frac{p}{2} \\ I(x,y,c), & \text{with probability } 1-p \end{cases}$$

Here, (x, y) denotes the spatial coordinates of a pixel, $c \in \{1, \ldots, C\}$ indicates the channel index, and $p \in [0, 1]$ is the noise level, representing the fraction of pixels affected by the corruption. A value of $\tilde{I}(x, y, c) = 0$ corresponds to pepper noise (black pixel), while $\tilde{I}(x, y, c) = 255$ corresponds to salt noise (white pixel), assuming 8-bit grayscale or color intensity values.

In our implementation, the severity of the corruption is controlled by the *amount* parameter, which we set to $p = 0.05$, thereby corrupting 5% of the image pixels. This parameter typically ranges from 0.01 to 0.2 in practice. The value of p plays a critical role in determining the corruption intensity and, consequently, the difficulty of the downstream learning task. Salt and pepper noise remains a widely studied corruption type in both classical image processing and modern robustness research.

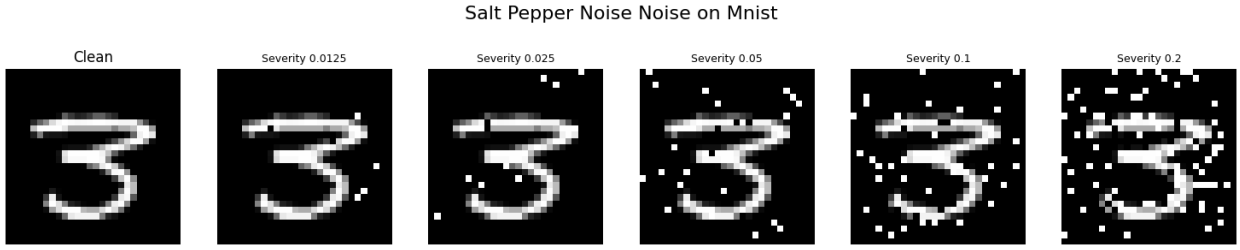


Figure 3.9: Visualization of Five Salt and Pepper Noise Severities on MNIST Input

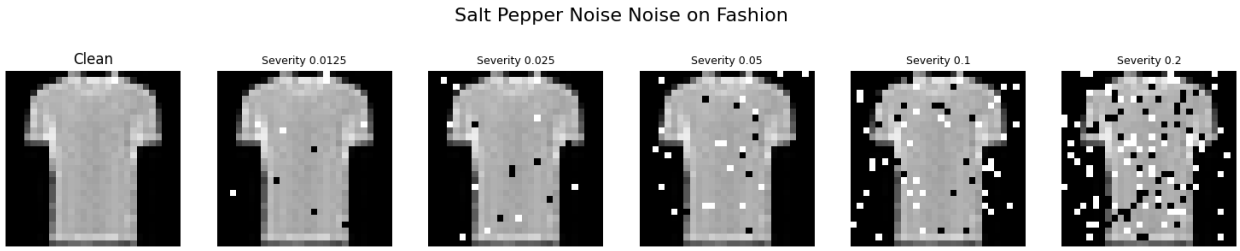


Figure 3.10: Visualization of Five Salt and Pepper Noise Severities on Fashion MNIST Input

- **Speckle Noise**:

  Speckle noise introduces *multiplicative noise*, which simulates grainy interference patterns often encountered in coherent imaging systems such as synthetic aperture radar (SAR) and medical ultrasound. This type of noise serves as a crucial benchmark in evaluating the robustness of image processing and machine learning models (Goodman (1975)).

  Mathematically, speckle noise can be modeled as follows:

$$\tilde{X}(i,j) = X(i,j) + X(i,j) \cdot N(i,j)$$

  where: $\tilde{X}(i,j)$ denotes the corrupted pixel value at location (i,j), $X(i,j)$ is the original clean image pixel value at (i,j), and $N(i,j) \sim \mathcal{N}(0,\sigma^2)$ is the multiplicative noise sampled independently from a Gaussian distribution with zero mean and variance σ^2.

  This formulation reflects the *intensity-dependent* nature of speckle noise, where the noise amplitude scales with the pixel intensity itself.

  In our implementation, we apply speckle noise by first sampling noise values $N(i,j)$ from a normal distribution $\mathcal{N}(0,\sigma^2)$ and then computing the noisy image $\tilde{X}(i,j)$ using the above formula. We set the severity of the corruption to $\sigma = 0.1$, which produces visible but moderate degradation. In practice, the severity can vary between $\sigma = 0.05$ and $\sigma = 0.2$, enabling controlled experimentation with different levels of interference.

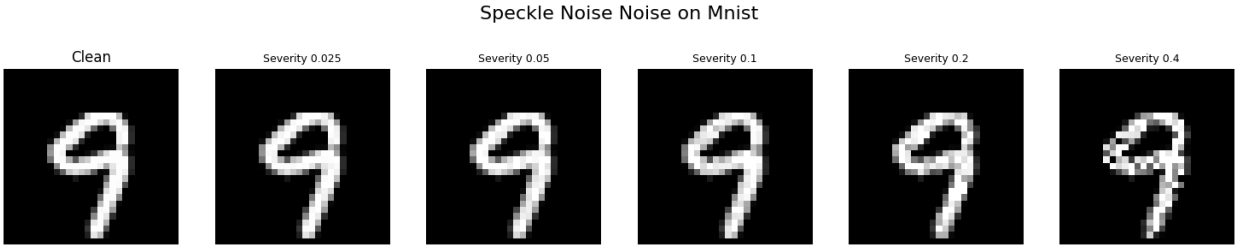


Figure 3.11: Visualization of Five Speckle Noise Severities on MNIST Input

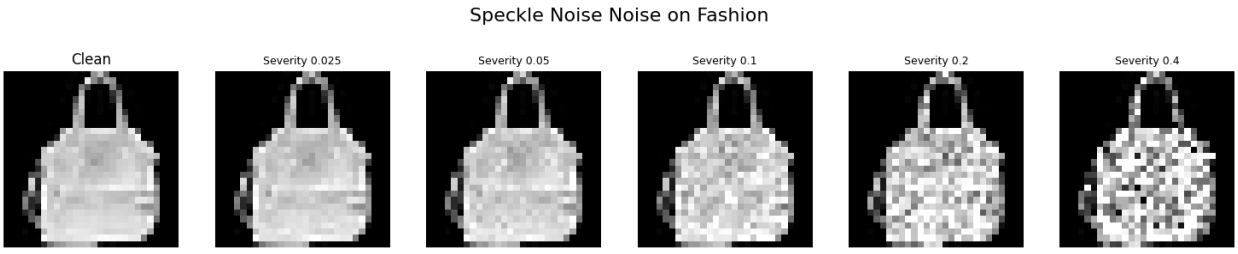


Figure 3.12: Visualization of Five Speckle Noise Severities on Fashion MNIST Input

3.2.2 Transformation Filters

These filters modify the appearance or geometry of an image by applying structured changes. They simulate common transformations that can occur in real-world environments and are useful for testing model robustness.

- **Gaussian Blur**:

  Simulates image defocus or motion blur by convolving the input image with a Gaussian kernel. This operation reduces high-frequency components such as edges and fine textures, thereby introducing a smoothing effect that is commonly observed in real-world data corruptions (Lindeberg (1990), Shapiro and Stockman (2001)).

  The Gaussian blur operation is defined mathematically as:

$$I'(x,y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} G(i,j;\sigma) \cdot I(x+i, y+j)$$

  where $I(x,y)$ is the original image intensity at pixel location (x,y), $I'(x,y)$ is the resulting blurred image, and $G(i,j;\sigma)$ is the Gaussian kernel defined as:

$$G(i,j;\sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{i^2+j^2}{2\sigma^2}\right)$$

  Here, (i,j) are the coordinates relative to the center of the kernel, σ is the standard deviation controlling the spread of the Gaussian distribution, and k determines the kernel radius (i.e., for a $(2k+1) \times (2k+1)$ kernel).

The severity of Gaussian blur is primarily controlled by the kernel size and the standard deviation σ, with larger values leading to stronger smoothing effects. In our implementation, a kernel size of 5×5 was used, corresponding to $k = 2$. This parameter setting introduces a moderate level of blur sufficient to simulate realistic degradation. Generally, the severity can vary from smaller kernels such as 3×3 to larger ones like 11×11, depending on the desired level of corruption.

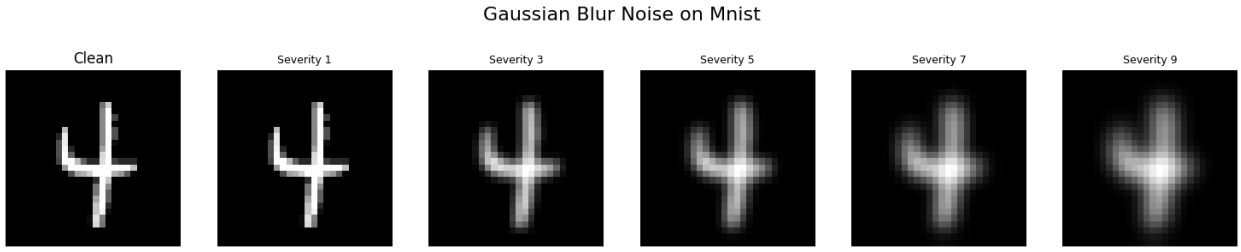


Figure 3.13: Visualization of Five Gaussian Blur Noise Severities on MNIST Input

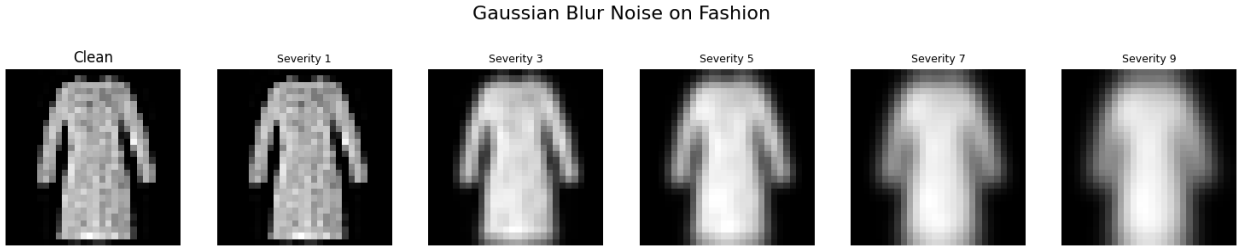


Figure 3.14: Visualization of Five Gaussian Blur Noise Severities on Fashion MNIST Input

- **Motion Blur**:

  This corruption simulates the effect caused by relative motion between the camera and the scene during exposure, resulting in a linear blur along the direction of motion. It is commonly observed in real-world scenarios involving dynamic environments, such as moving vehicles or handheld camera shake.

  Mathematically, the motion blur process can be modeled as a convolution of the original image $I(x, y)$ with a motion blur kernel $K_{\text{motion}}(x, y)$:

$$I_{\text{blurred}}(x, y) = (I * K_{\text{motion}})(x, y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} I(x - i, y - j) \cdot K_{\text{motion}}(i, j) \qquad (3.1)$$

where $I_{\text{blurred}}(x, y)$ is the resulting blurred image, $*$ denotes the 2D convolution operation, and $K_{\text{motion}}(i, j)$ is a normalized filter kernel of size $(2k + 1) \times (2k + 1)$ that defines the direction and magnitude of the blur. The kernel is typically designed to have nonzero values along a straight line that aligns with the direction of motion, and zeros elsewhere.

For a $(2k+1) \times (2k+1)$ kernel simulating motion blur along a 45° diagonal, the kernel $K_{\text{motion}}(i, j)$ can be defined as:

$$K_{\text{motion}}(i, j) = \begin{cases} \frac{1}{2k+1}, & \text{if } i = j \text{ and } -k \leq i, j \leq k \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

This creates a normalized kernel with uniform weights along the main diagonal, ensuring the sum of all elements equals 1.

In our case, with $k = 2$, the kernel becomes:

$$K_{\text{motion}} = \frac{1}{5} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

In our implementation, we use this motion blur kernel of size 5×5, oriented at an angle of 45° to simulate diagonal motion. The chosen kernel size directly affects the severity of the blur: larger kernels produce more pronounced distortion, while smaller ones retain finer image details. By setting the size to 5, we aim for a moderate level of corruption—sufficient to reflect real motion artifacts without overly degrading the image to the point of label ambiguity. More broadly, the severity of motion blur can be tuned by varying the kernel size, typically within the range of 3 to 15.

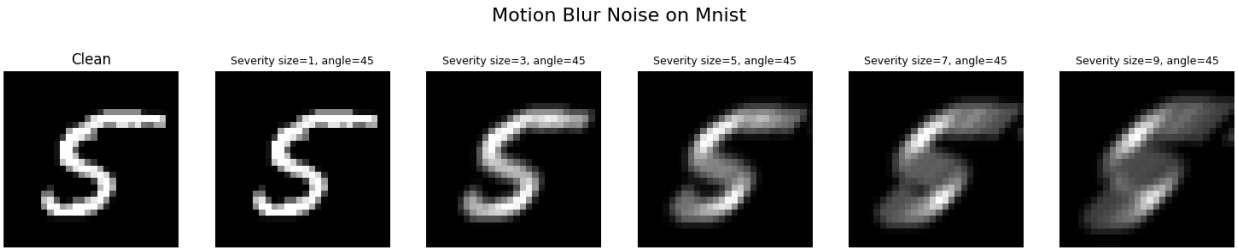


Figure 3.15: Visualization of Five Motion Blur Noise Severities on MNIST Input

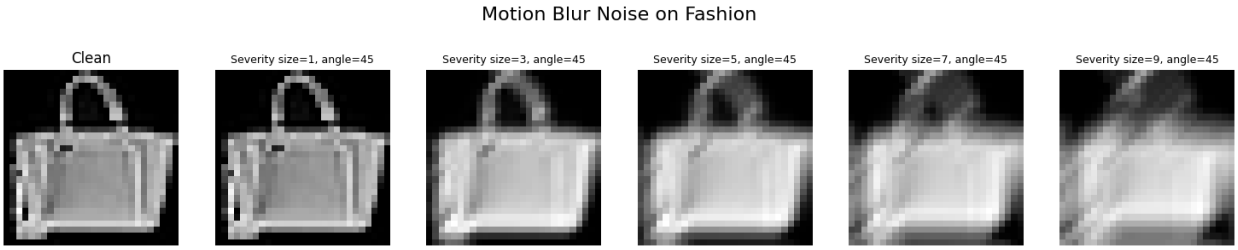


Figure 3.16: Visualization of Five Motion Blur Noise Severities on Fashion MNIST Input

- **Elastic Deformation**:

  Elastic deformation introduces localized geometric distortions to an image by applying spatially varying displacement fields. This process simulates non-rigid transformations such as bending, stretching, or compression, and is particularly relevant for evaluating a model's robustness to subtle shape changes that do not significantly alter the image's overall structure. This technique was originally proposed by Jaderberg et al. (Jaderberg et al. (2015)) in the context of spatial transformer networks.

  Mathematically, the deformation is applied to each pixel of the input image by adding a displacement vector derived from a smoothed random displacement field. Let $\mathbf{I}(\mathbf{x})$ denote the intensity at pixel location $\mathbf{x} = (x, y)$ in the input image. The transformed image $\mathbf{I}'(\mathbf{x})$ is given by:

$$\mathbf{I}'(\mathbf{x}) = \mathbf{I}(\mathbf{x} + \mathcal{G}_\sigma * \mathcal{N}_\alpha(\mathbf{x}))$$

  Here:

– $\mathcal{N}_\alpha(\mathbf{x}) = \alpha \cdot \mathcal{N}(0,1)^2$ is a 2D random displacement field, where each component is independently sampled from a standard normal distribution $\mathcal{N}(0,1)$ and scaled by the deformation intensity parameter α. That is, for each pixel $\mathbf{x}$,

$$\mathcal{N}_\alpha(\mathbf{x}) = \begin{bmatrix} u_x(\mathbf{x}) \\ u_y(\mathbf{x}) \end{bmatrix}, \quad u_x(\mathbf{x}), u_y(\mathbf{x}) \sim \alpha \cdot \mathcal{N}(0,1)$$

– $\mathcal{G}_\sigma$ is a Gaussian kernel with standard deviation σ, defined as:

$$\mathcal{G}_\sigma(i,j) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{i^2+j^2}{2\sigma^2}\right)$$

for $(i,j) \in [-k,k]^2$, where k determines the kernel size (typically chosen such that $k = 3\sigma$).

– The smoothed displacement field $\mathcal{G}_\sigma * \mathcal{N}_\alpha(\mathbf{x})$ is the convolution of the Gaussian kernel with the random displacement field:

$$\mathcal{G}_\sigma * \mathcal{N}_\alpha(\mathbf{x}) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} \mathcal{G}_\sigma(i,j) \cdot \mathcal{N}_\alpha(\mathbf{x} - (i,j))$$

This operation ensures the displacement field varies smoothly across space.

In our implementation, we used fixed values of $\alpha = 34$ and $\sigma = 4$, which were chosen to produce visually noticeable but realistic distortions. These values yield moderately strong deformations while maintaining coherent image structure. However, in practice, elastic deformation can be applied with a range of severity levels by varying these parameters. For instance, $\alpha \in [20, 50]$ and $\sigma \in [3, 6]$ offer a reasonable range for exploring different degrees of shape perturbation.

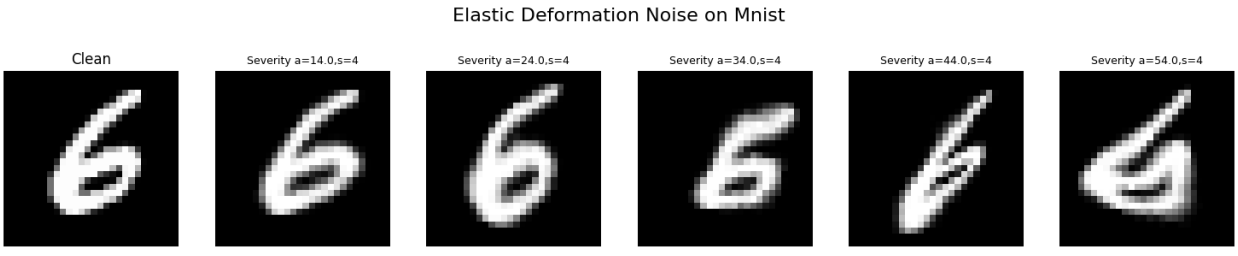


Figure 3.17: Visualization of Five Elastic Deformation Noise Severities on MNIST Input

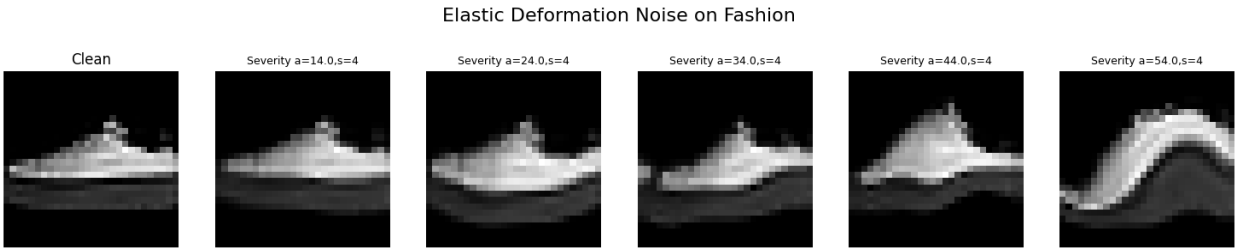


Figure 3.18: Visualization of Five Elastic Deformation Noise Severities on Fashion MNIST Input

- **Brightness/Contrast Adjustment**:

  This corruption modifies the global luminance characteristics of an image by adjusting its brightness and contrast levels, without altering the spatial content or structural features of the scene. It is designed to simulate realistic lighting variations that may arise in real-world conditions, such as underexposure, overexposure, or inconsistent ambient illumination (Peli (1990)).

  Mathematically, the transformation applied to each pixel intensity $I_{i,j}$ in the image can be expressed as:

$$I'_{i,j} = \text{clip}\left((1+\gamma)\cdot I_{i,j} + \beta \cdot 255,\ 0,\ 255\right)$$

  where $I'_{i,j}$ is the adjusted pixel intensity at position (i,j), $\gamma \in \mathbb{R}$ denotes the contrast factor, and $\beta \in \mathbb{R}$ represents the brightness factor. The function $\text{clip}(\cdot,\ 0,\ 255)$ ensures that the output remains within the valid 8-bit image intensity range $[0, 255]$.

  In our implementation, we applied a contrast factor of $\gamma = 0.2$ and a brightness factor of $\beta = 0.2$. This corresponds to amplifying the contrast by 20% and increasing the image brightness by an offset of 0.2×255. This form of corruption is commonly adopted in robustness evaluation frameworks to assess model sensitivity to changes in global illumination and overall intensity dynamics.

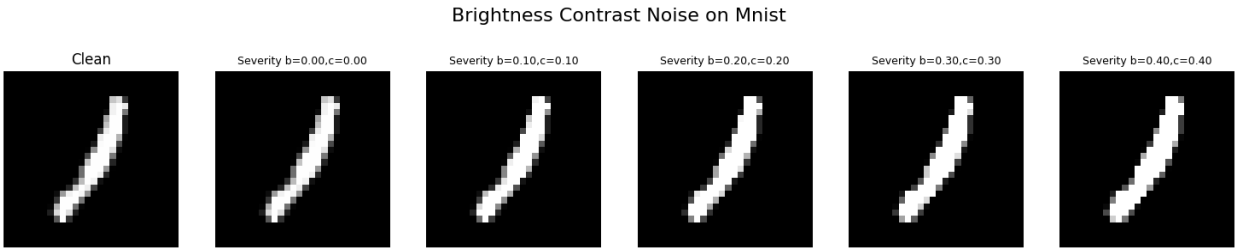


Figure 3.19: Visualization of Five Brightness Contrast Noise Severities on Mnist Input

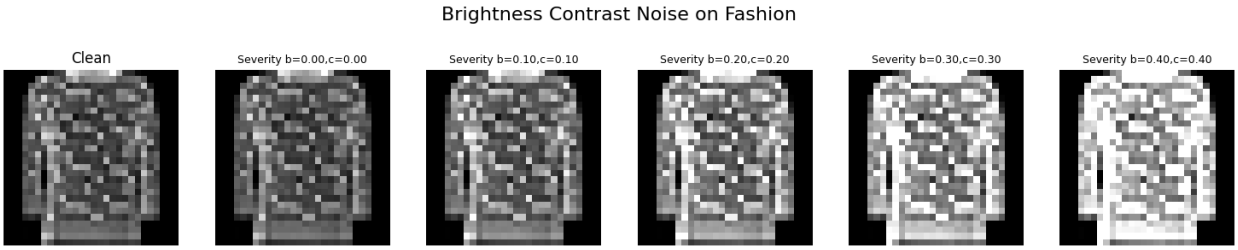


Figure 3.20: Visualization of Five Brightness Contrast Noise Severities on Fashion Input

3.2.3 Compression and Occlusion Methods

These corruptions simulate data loss or visual obstruction, which commonly occur in low-bandwidth scenarios or when parts of the image are blocked.

- **JPEG Compression**:

  JPEG (Joint Photographic Experts Group) compression simulates image degradation through a lossy compression algorithm that significantly reduces file size by discarding perceptually less important information. This process, however, introduces visible artifacts such as blurring, blockiness, and color distortions. The JPEG algorithm, initially proposed by Wallace et al. (Wallace (1992)), relies primarily on the Discrete Cosine Transform (DCT) to compact energy into a few frequency coefficients.

  The core step of JPEG involves dividing an image into 8×8 pixel blocks, each of which undergoes a 2D DCT:

$$F(u,v) = \frac{1}{4}C(u)C(v)\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)\cos\left[\frac{(2x+1)u\pi}{16}\right]\cos\left[\frac{(2y+1)v\pi}{16}\right]$$

where $f(x,y)$ denotes the intensity value at position (x,y) in the spatial domain, and $F(u,v)$ represents the corresponding DCT coefficient at frequency indices (u,v). The normalization terms are defined as $C(u) = \frac{1}{\sqrt{2}}$ if $u = 0$ and $C(u) = 1$ otherwise (similarly for $C(v)$).

Following the DCT, coefficients are quantized using a standard quantization matrix $Q(u,v)$, scaled according to a *quality factor* $q \in [1,100]$. The quantized coefficients $\hat{F}(u,v)$ are computed as:

$$\hat{F}(u,v) = \text{round}\left(\frac{F(u,v)}{Q_q(u,v)}\right)$$

where $Q_q(u,v)$ is the quantization matrix adjusted based on the quality factor q; lower q values increase the entries of Q_q, thus yielding higher compression and more distortion.

In our implementation, we applied JPEG compression using a quality factor of $q = 50$, representing a moderate severity level. This parameter controls the balance between compression strength and visual fidelity: lower values result in more aggressive quantization, leading to more pronounced degradation, while higher values preserve more image detail.

Although we used only a single quality level in our evaluation, JPEG compression severity can typically be varied by selecting q in the range of 10 (representing high degradation) to 90 (low degradation).

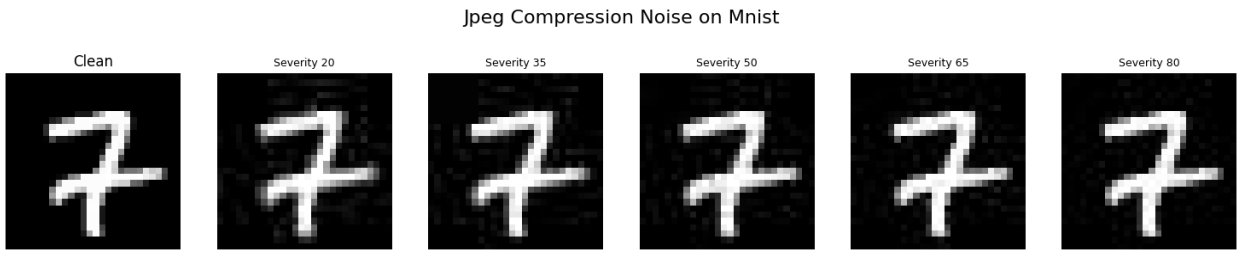


Figure 3.21: Visualization of Five JPEG Compression Noise Severities on MNIST Input

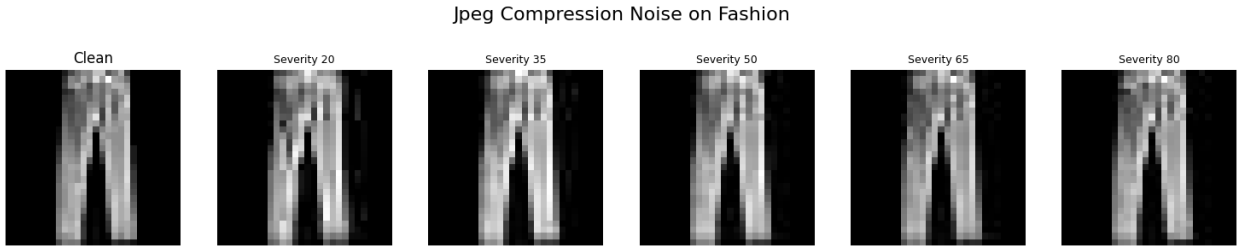


Figure 3.22: Visualization of Five JPEG Compression Noise Severities on Fashion MNIST Input

- **Random Occlusion**:

  This corruption simulates partial visual obstruction by randomly removing rectangular regions from the image, thereby mimicking scenarios where objects are partially hidden due to occluding elements or limited visibility (Hendrycks and Dietterich (2019)).

  Mathematically, let $\mathbf{X} \in \mathbb{R}^{H \times W}$ denote a grayscale input image of height H and width W. The occluded image $\mathbf{X}'$ is generated by applying a binary mask $\mathbf{M} \in \{0,1\}^{H \times W}$ to $\mathbf{X}$, such that:

  $$\mathbf{X}' = \mathbf{X} \odot \mathbf{M}$$

  Here, $\odot$ denotes the element-wise (Hadamard) product. The mask $\mathbf{M}$ is initialized as an all-ones matrix (i.e., no occlusion), and a square patch of zeros of size $s \times s$, where $s \in \{1, \ldots, S_{\max}\}$, is placed at a randomly chosen location (i, j) within the image. In this study, we set $S_{\max} = 8$. The number of occlusion patches per image is limited by $N_{\max} = 1$.

In our implementation, each image is subjected to a single occlusion patch (`max_num_patches = 1`) with patch size $s \times s$ randomly sampled such that $1 \leq s \leq 8$. The top-left coordinate (i, j) of the patch is also randomly selected, ensuring that the patch lies entirely within the image boundaries. The region corresponding to the patch is set to zero (black) to simulate missing or blocked visual information, with the randomness in both patch size and location introducing variability in the occlusion effect across the dataset.

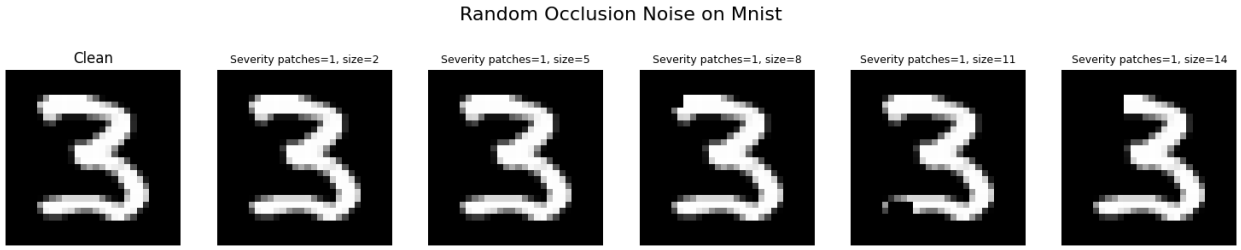


Figure 3.23: Visualization of Five Random Occlusion Noise Severities on MNIST Input

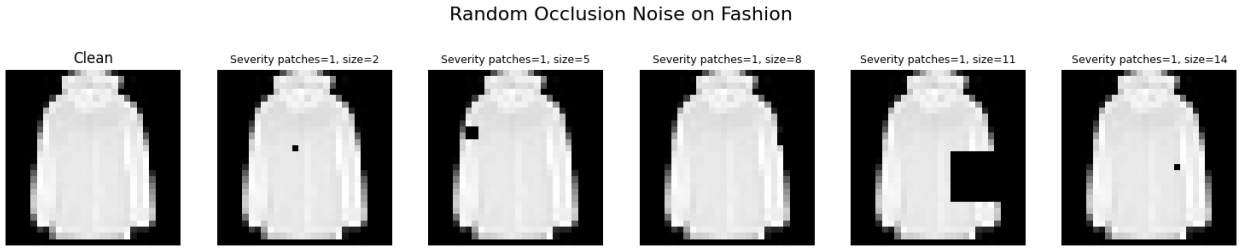


Figure 3.24: Visualization of Five Random Occlusion Noise Severities on Fashion MNIST Input

3.2.4 Mixed Corruption

In addition to applying individual corruption noises to original images and creating new sets of images, we also evaluate model performance using a mixed-noise test set. Each image in this set is randomly corrupted with one of the above types to assess model robustness across multiple degradation conditions. To ensure the preservation of the labels, we carefully control the intensity and ratio of corruptions applied. The goal is to keep the corruptions

subtle enough so that the true class label of the image remains valid. For instance, excessive rotations, such as rotating an image of the digit "6" by 180 degrees, would change the label to "9." Similarly, corrupting an image too much — such as brightening it to make it completely white or darkening it to make it completely black — would render the image unrecognizable, making its label invalid (e.g., a "6" becomes a blank image with no distinguishable features).

Therefore, we ensure that the applied corruptions do not significantly alter the label or make the image unsuitable for model testing. Preserving the label during corruption is crucial, yet it is often overlooked in some studies. If we assume that the corrupted image retains the same label as the original and generate new images based on this assumption, we may end up with a large number of invalid images. These corrupted images may no longer be recognizable or could have labels that no longer align with the content, making them unreliable for testing. Without a manual labeling process, it's vital to ensure that the data generation process preserves both the label and the integrity of the image. To remove the need for manual labeling in testing, we must guarantee that our data generation process produces images that are both label-preserving and human-recognizable.

- **Mixed-Noise**: We randomly select 1,000 samples from each of the corrupted test sets (for both MNIST and Fashion-MNIST), where each sample is randomly corrupted with one of the noise types described above. The corruptions are applied in a controlled manner to avoid invalidating the image labels.

4. *Model Architecture and Training*

4.1 Traditional Machine Learning Models

Traditional machine learning (ML) algorithms remain valuable for image classification due to their computational efficiency, interpretability, and well-understood theoretical foundations (Bishop (2006), Murphy (2012)). Unlike deep learning models, which automatically learn layered (hierarchical) features—representations built by combining simpler patterns into increasingly complex ones—traditional machine learning relies on manually engineered features. This manual feature-engineering approach offers greater transparency, which is crucial in applications like medical imaging and safety-critical systems where decision traceability is essential (Caruana et al. (2015)).

This study evaluates five classical ML methods: Logistic Regression, Support Vector Machine (SVM), k-Nearest Neighbors (K-NN), Random Forest, and Multi-layer Perceptron (MLP). All models were implemented using scikit-learn (Pedregosa et al. (2011)) and trained on MNIST and Fashion-MNIST datasets. Each 28×28 grayscale image was flattened into a 784-dimensional vector, standardized to zero mean and unit variance, and used as input. To ensure reproducibility, random seeds were fixed across data loading, model training, and evaluation. Trained models and scalers were saved for consistent inference on both clean and corrupted test data, establishing a baseline for robustness analysis.

4.1.1 Logistic Regression

Logistic regression is a fundamental linear classifier used for binary and multi-class classification. It models the relationship between input features and class probabilities using a linear decision function followed by a non-linear transformation. In its multinomial form, it estimates class probabilities using the softmax function, making it suitable for multi-class classification tasks such as MNIST (Bishop (2006), Murphy (2012)).

In the context of MNIST, each grayscale image of size 28×28 is flattened into a feature vector $X \in \mathbb{R}^{784}$. The model computes a score for each class $k \in \{1, \ldots, 10\}$ using a linear function:

$$z_k = \beta_k^\top X + b_k$$

where:

- $\beta_k \in \mathbb{R}^{784}$ is the *weight vector* for class k, assigning learned importance to each input pixel,
- $b_k \in \mathbb{R}$ is the *bias term* for class k,
- $z_k \in \mathbb{R}$ is the *logit* or unnormalized score for class k.

The class probabilities are then obtained using the softmax function:

$$P(Y = k \mid X) = \frac{e^{z_k}}{\sum_{j=1}^{10} e^{z_j}}$$

where $e \approx 2.718$ is Euler's number, and the denominator ensures the probabilities over all classes sum to 1. The model predicts the class with the highest probability:

$$\hat{Y} = \arg\max_k \; P(Y = k \mid X)$$

To learn the parameters β_k and b_k, the model minimizes the negative log-likelihood of the true class labels, also known as the cross-entropy loss. For a single training example (X, y), where y is the true class label, the loss is defined as:

$$\mathcal{L}(X, y) = -\log P(Y = y \mid X) = -\log\left(\frac{e^{z_y}}{\sum_{j=1}^{10} e^{z_j}}\right)$$

The total loss is computed by averaging over all training examples. Optimization algorithms such as stochastic gradient descent (SGD) or the limited-memory Broyden Fletcher Goldfarb Shanno (L-BFGS) method are used to minimize this loss and update the parameters during training.

The model addresses the need for reliable, interpretable classification in real-world applications where understanding decision-making is crucial. While it performs well for linearly separable data, its limited capacity for capturing complex non-linear relationships highlights the importance of testing and comparing it against more expressive models on challenging datasets.

In this project, the model is implemented using `scikit-learn`'s `LogisticRegression` class, with the `lbfgs` solver and `multinomial` option enabled to handle multi-class output. As a linear model, logistic regression lacks the expressive power needed to model complex, non-linear patterns. This offers a clear baseline for evaluating robustness improvements provided by more advanced models. Additionally, because of its fast training time and low computational cost, logistic regression remains widely applicable in many real-world domains (Hosmer et al. (2013)), making it important to thoroughly test its robustness under various conditions.

For clarity, we summarize the notation used throughout this section:

- $X \in \mathbb{R}^{784}$: flattened input image (feature vector),
- $\beta_k \in \mathbb{R}^{784}$: weight vector for class k,
- $b_k \in \mathbb{R}$: bias term for class k,
- $z_k = \beta_k^\top X + b_k$: logit (linear score) for class k,
- e: Euler's number, used in the softmax computation,
- $P(Y = k \mid X)$: predicted probability of class k,

- $\hat{Y}$: predicted class label,
- $\mathcal{L}(X, y)$: cross-entropy loss for true label y.

4.1.2 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised classification algorithm designed for separating different classes by finding the optimal boundary, or hyperplane, with the largest possible margin between classes. This margin maximization principle helps the model generalize better and reduces overfitting, especially when data is well-separated (Cortes and Vapnik (1995)).

In the context of MNIST, we employ an SVM with a Radial Basis Function (RBF) kernel to handle the complex, non-linear relationships present in handwritten digit images. The RBF kernel enables the SVM to map input features into a higher-dimensional space, where non-linear separation becomes feasible. The RBF kernel function is defined as:

$$K(x_i, x) = \exp(-\gamma \|x_i - x\|^2)$$

where x_i are the support vectors, x is the input image vector, and γ controls the kernel's width. During prediction, each MNIST image is flattened into a feature vector and transformed using the RBF kernel. The SVM then uses its learned hyperplane to assign the image to the digit class via the decision function:

$$f(x) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b\right)$$

By including SVM in our comparisons, we aim to assess its robustness and classification performance relative to both simpler linear models, like Logistic Regression, and more complex non-linear methods, such as deep learning models.

In this project, we implemented an SVM classifier using scikit-learn with the RBF kernel and gamma set to scale, which automatically adjusts the kernel's behavior based on the data. The model is trained by minimizing hinge loss, which penalizes misclassifications and

points near the decision boundary. The regularization parameter C is used to control the trade-off between margin maximization and classification error.

4.1.3 k-Nearest Neighbors ($K - NN$)

k-Nearest Neighbors (k-NN) is a non-parametric classification method that assigns to each input sample the majority class among its k closest training examples, typically measured by Euclidean distance (Cover and Hart (1967)). Unlike parametric models such as logistic regression, k-NN does not assume a fixed data distribution and instead relies on the entire training set for predictions.

Each image is represented as a flattened vector of pixel values. To classify a new image, k-NN computes the Euclidean distance between the new image vector $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ and every training image vector $x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,n}) \in \mathbb{R}^n$, where n is the number of features (for MNIST, $n = 784$):

$$d(x, x_i) = \sqrt{\sum_{j=1}^{n} (x_j - x_{i,j})^2}$$

Here, $d(x, x_i)$ is the Euclidean distance measuring the similarity between the new image x and the training image x_i. The model then selects the k closest training images, denoted by the set $N_k(x)$, and assigns to the new image the class $\hat{y}$ that appears most frequently among these neighbors:

$$\hat{y} = \arg\max_{c} \sum_{i \in N_k(x)} \mathbb{I}(y_i = c)$$

where $\mathbb{I}(\cdot)$ is the indicator function that equals 1 if the condition is true and 0 otherwise, and y_i is the class label of the i-th neighbor in $N_k(x)$.

k-Nearest Neighbors remains a popular choice in many practical applications due to its simplicity, interpretability, and minimal training cost, as it requires no explicit model fitting (Cover and Hart (1967), Hastie et al. (2009)). Its non-parametric nature allows it to adapt flexibly to complex data distributions without assuming a specific underlying model,

making it particularly useful in scenarios where data structure is unknown or highly irregular (Bishop (2006)). Furthermore, k-NN is widely applicable across diverse domains such as recommendation systems, anomaly detection, and certain medical diagnoses, where quick and understandable decisions are essential (Tan et al. (2005), Zhang and Zha (2017)). Given its balance of low computational overhead during training and reasonable effectiveness, it serves as a valuable baseline for evaluating model robustness. In our project, we employed k-NN to assess generalization under data distribution shifts and noise, thereby highlighting the trade-offs between computational cost and robustness when compared to more complex models like deep neural networks and logistic regression.

4.1.4 Random Forest

Random Forest is an ensemble learning algorithm that builds a collection of decision trees and aggregates their predictions to improve generalization performance (Breiman (2001)). Each tree is trained on a bootstrap sample—random sampling with replacement—from the original training data. At each node within a tree, a random subset of features is selected, and the best feature among them is used to split the data. This dual randomness—both in data sampling and feature selection—reduces variance and mitigates overfitting, addressing limitations of individual decision trees.

In the context of image classification tasks such as MNIST, where input features correspond to pixel intensities, Random Forest treats these inputs as numerical variables. Each tree independently produces a class prediction and is constructed greedily using the following procedure:

- A bootstrap sample of the training data is drawn.
- Starting at the root node, a random subset of features is selected.
- Among these features, the best split is chosen to minimize an impurity measure (e.g., Gini impurity or entropy).

- The data is split accordingly, and the process repeats recursively for child nodes until stopping criteria are met (e.g., maximum depth or minimum samples per leaf).

This process continues until the tree is fully grown. Because each tree sees a slightly different subset of data and features, the ensemble benefits from reduced correlation between trees, improving generalization. Each individual tree uses a splitting criterion such as **Gini impurity**:

$$G = 1 - \sum_{i=1}^{C} p_i^2$$

where:

- G is the Gini impurity,
- C is the total number of classes,
- p_i is the fraction of samples belonging to class i at a given node.

Alternatively, **entropy** can be used to measure information gain:

$$H = -\sum_{i=1}^{C} p_i \log(p_i)$$

where:

- H is the entropy,
- p_i is again the proportion of class i at the node,
- C is the number of classes.

The final prediction from the Random Forest is made by majority vote across the trees. Class probabilities can be computed by averaging the predicted probabilities from all trees.

Random Forest does not use gradient-based optimization or minimize a global loss function. Instead, each decision tree is built using a greedy, local procedure based on impurity reduction. Its ensemble structure, combining many diverse trees, reduces overfitting and leads to strong predictive performance.

It performs well even without extensive preprocessing or hyperparameter tuning, effectively handles non-linear relationships and high-dimensional data, and can estimate prediction uncertainty. However, prediction time can be slower for large ensembles, and interpretability is lower compared to simpler models like logistic regression.

4.1.5 Multi-layer Perceptron (MLP)

A Multi-layer Perceptron (MLP) is a class of feedforward neural networks composed of multiple fully connected layers. Each neuron in one layer is connected to every neuron in the next layer, allowing the network to model complex, non-linear relationships between input and output variables (Rumelhart et al. (1986)). MLPs are widely used in various domains due to their flexibility and ability to approximate a wide range of functions.

The hidden layers form the core computational structure of a MLP, positioned between the input and output layers. Each hidden layer applies a non-linear activation function—commonly the Rectified Linear Unit (ReLU)—to its inputs, enabling the network to learn non-linear decision boundaries:

$$\text{ReLU}(x) = \max(0, x)$$

where $x \in \mathbb{R}$ is the pre-activation input to a neuron. These hidden layers transform the input feature space through learned weights and biases, producing intermediate representations that capture higher-level patterns in the data.

To reduce overfitting, regularization techniques such as dropout are often applied during training. Dropout randomly deactivates a fraction of neurons in a layer, forcing the network to learn more robust and redundant features.

The final output layer typically matches the number of target classes and uses the softmax activation function to produce a probability distribution over possible classes:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \quad i = 1, \ldots, K$$

where $z_i \in \mathbb{R}$ is the logit (unnormalized score) for class i, and K is the total number of classes.

Training is typically performed using the categorical cross-entropy loss function:

$$\mathcal{L}_{\text{CE}}(y, \hat{y}) = -\sum_{i=1}^{K} y_i \log(\hat{y}_i)$$

where y_i is the true label encoded as a one-hot vector, and $\hat{y}_i$ is the predicted probability for class i. Optimization algorithms such as stochastic gradient descent (SGD) or Adam are used to iteratively update the model's parameters and minimize this loss.

MLPs are a foundational class of neural networks, balancing model capacity and computational cost, and are often used as a baseline or component in more complex deep learning architectures.

4.2 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a class of deep learning models specifically designed to capture spatial hierarchies and local patterns within image data (Krizhevsky et al. (2012), LeCun et al. (1998)). The core idea behind CNNs is to automatically learn spatial correlations between neighboring pixels or regions through convolutional layers, which apply localized filters with shared weights across the input. This structure efficiently detects patterns such as edges, textures, and more complex features, making CNNs exceptionally effective for image classification, object detection, and other vision tasks.

CNNs are widely popular in both academic research and industry due to their ability to model complex visual patterns while maintaining computational efficiency through parameter sharing and sparse connectivity. Their success in applications such as image recognition, medical image analysis, and autonomous driving has established them as a standard choice in computer vision.

We selected CNNs in this study because they excel at identifying spatial patterns and hierarchical features in image data, making them highly relevant for evaluating robustness against image perturbations. To explore different architectural choices and performance levels, we employed two well-established CNN architectures: Lenet-5, a pioneering model

for handwritten digit recognition, and ResNet-18, a deeper and more modern architecture known for its residual learning capability.

4.2.1 Lenet-5

Lenet-5 is a convolutional neural network (CNN) architecture introduced by LeCun et al. (LeCun et al. (1998)) for image classification tasks. It exemplifies how deep learning methods can automatically learn hierarchical spatial features from images, thereby reducing reliance on manual feature engineering.

The architecture comprises a sequence of convolutional, pooling, and fully connected layers that progressively transform the input image through parameterized operations. Each component and its associated parameters are described below.

The network accepts as input a single-channel grayscale image of size 32×32, represented as a tensor $x \in \mathbb{R}^{32\times32}$. The input layer normalizes and prepares the raw pixel data for subsequent processing. *This layer functions as the initial data interface, providing normalized input to the network.*

The first convolutional layer (C1) applies six learnable filters $W^{(k)} \in \mathbb{R}^{5\times5}$, for $k = 1, \ldots, 6$, to the input image according to:

$$h_{ij}^{(k)} = \tanh\left((W^{(k)} * x)_{ij} + b^{(k)}\right),$$

where $W^{(k)}$ denotes the k-th convolution kernel, $b^{(k)}$ is the bias term, $*$ represents the 2D convolution operation, and tanh is the activation function. *This layer is responsible for extracting low-level local features such as edges and textures from the input image.*

Subsequently, the first subsampling layer (S2) performs average pooling with a 2×2 kernel and stride 2:

$$s_{ij}^{(k)} = \frac{1}{4}\sum_{m=0}^{1}\sum_{n=0}^{1} h_{2i+m,2j+n}^{(k)}.$$

This operation reduces the spatial dimensions of the feature maps, resulting in spatial downsampling. *The primary role of this layer is to reduce feature map resolution while enhancing translational invariance and lowering computational complexity.*

The second convolutional layer (C3) applies 16 filters $W^{(k)} \in \mathbb{R}^{5\times5}$ over selected combinations of feature maps from S2:

$$h_{ij}^{(k)} = \tanh\left((W^{(k)} * s^{(\cdot)})_{ij} + b^{(k)}\right).$$

Here, $s^{(\cdot)}$ indicates a subset of S2 feature maps connected to each filter. *This layer learns more complex and abstract representations by integrating features across multiple channels.*

Following this, the second subsampling layer (S4) performs average pooling analogous to S2, further decreasing the spatial resolution. *This stage consolidates features and reduces dimensionality, contributing to greater robustness to spatial variations.*

The output of S4 is flattened and fed into a fully connected layer (F5), wherein each neuron performs a weighted sum followed by a tanh activation:

$$a_i = \tanh(W_i^\top z + b_i),$$

with z representing the flattened input vector, W_i the weight vector for neuron i, and b_i its bias. This layer typically contains 120 neurons, succeeded by another fully connected layer with 84 neurons. *The fully connected layers synthesize the extracted features to form higher-level abstractions relevant to classification.*

The output layer (F6) consists of 10 neurons corresponding to the classification categories. The softmax function converts the pre-activation outputs (logits) into class probabilities:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}, \quad i = 1, \ldots, C,$$

where $C = 10$ denotes the number of classes. *This layer provides a probabilistic interpretation of the network's predictions over the target classes.*

The network is trained by minimizing the categorical cross-entropy loss:

$$\mathcal{L}_{\text{CE}}(y, \hat{y}) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i),$$

where y_i is the ground truth label encoded as a one-hot vector, and $\hat{y}_i$ the predicted class probability. *The loss function quantifies the discrepancy between predicted probabilities and true labels, guiding parameter updates.*

During training, parameters $\{W, b\}$ are optimized using stochastic gradient descent or its variants (e.g., Adam) to minimize the loss over the training dataset.

Lenet-5 remains a seminal CNN architecture due to its conceptual clarity and efficacy in image classification, despite the subsequent development of deeper and more complex networks.

4.2.2 ResNet-18

ResNet-18 is a deep convolutional neural network introduced by He et al. (He et al. (2016)), designed to address the vanishing gradient problem in deep architectures through the use of residual connections. These allow the network to learn modifications (residuals) to identity mappings, facilitating the training of very deep networks and improving both convergence and accuracy.

Input Layer

The network accepts an input image $\mathbf{x} \in \mathbb{R}^{224\times224\times3}$, representing an RGB image. Inputs are typically normalized by subtracting the dataset mean and dividing by the standard deviation to ensure consistent scale and distribution across the dataset. This preprocessing step is essential for **stabilizing training** and ensuring **faster convergence**, as unnormalized inputs can lead to unstable gradients and slow learning.

Initial Convolutional Layer

A convolutional layer with 64 filters of size 7×7 and stride 2 is applied. This layer has a large receptive field, allowing it to capture **low-level visual patterns**, such as edges, blobs, and textures, across a wide spatial area. It also downsamples the input, reducing computational cost early in the network. This is followed by batch normalization and ReLU activation.

Batch normalization stabilizes training by normalizing the layer's input distribution:

$$\hat{x} = \frac{x - \mu_{\text{batch}}}{\sqrt{\sigma^2_{\text{batch}} + \epsilon}} \times \gamma + \beta,$$

where $\mu_{\text{batch}}, \sigma^2_{\text{batch}}$ are the batch mean and variance, and γ, β are learnable parameters. This operation improves **gradient flow**, **reduces internal covariate shift**, and supports

higher learning rates.

The ReLU activation introduces non-linearity by zeroing negative values:

$$\text{ReLU}(x) = \max(0, x),$$

which allows the network to learn **complex, non-linear functions** that linear layers cannot model alone.

Max Pooling Layer

A 3×3 max pooling layer with stride 2 follows, which further reduces the spatial dimensions while retaining the most dominant features in local regions. This operation introduces **translation invariance** and helps the model focus on **salient structures** in the image, like edges or corners, without increasing parameter count.

Residual Blocks

The core of ResNet-18 is composed of four stages of residual blocks, each consisting of two 3×3 convolutional layers followed by batch normalization and ReLU activation. A residual block learns a transformation function $\mathcal{F}(\mathbf{x})$, which is then added to the original input $\mathbf{x}$:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x},$$

where $\mathbf{y}$ is the output and $\{W_i\}$ are the weights. This skip connection allows the network to learn the **residual**, or difference from identity, which simplifies learning and improves gradient flow during backpropagation. If dimensions differ due to downsampling, a 1×1 convolution with stride 2 is applied to the shortcut path to align them.

Each stage increases the number of channels (64, 128, 256, 512), while the spatial resolution is halved via stride-2 convolution in the first block of each stage. These stages form a **hierarchical feature extraction** pipeline:

- Stage 1 (64 channels): Learns **fine-grained local patterns**, such as textures and small shapes.

- Stage 2 (128 channels): Detects **repeated patterns or part-level features**, like corners or blobs.
- Stage 3 (256 channels): Captures **combinations of parts** to form mid-level structures (e.g., faces, limbs).
- Stage 4 (512 channels): Extracts **semantic-level, abstract features** relevant to object categories.

This hierarchical representation enables the network to understand both local and global aspects of the image progressively.

Global Average Pooling

After the final stage, global average pooling is applied to reduce each $H' \times W'$ feature map F_k into a single scalar value:

$$g_k = \frac{1}{H'W'} \sum_{i=1}^{H'} \sum_{j=1}^{W'} F_k(i,j),$$

producing a feature vector that summarizes the **presence of high-level features** across the spatial domain. This drastically reduces the number of parameters and prevents overfitting, while preserving the **semantic essence** of each feature map.

Fully Connected Layer and Softmax

The pooled features are fed into a fully connected (dense) layer that outputs class logits $\mathbf{z} \in \mathbb{R}^C$, where C is the number of classes. The softmax function then converts these logits into class probabilities:

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}, \quad i = 1, \ldots, C,$$

which represent the model's confidence in each class. This final classification step leverages the high-level representations learned throughout the network.

All convolutional weights are initialized using the normal initialization:

$$W \sim \mathcal{N}\left(0, \sqrt{\frac{2}{n_l}}\right),$$

where n_l is the number of input units to the layer. This initialization ensures that the variance of activations remains consistent across layers, promoting **stable and efficient gradient propagation**.

Adam optimizer is usually used to minimize the categorical cross-entropy loss:

$$\mathcal{L}_{\text{CE}}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i),$$

where $\mathbf{y}$ is the one-hot encoded ground truth label and $\hat{\mathbf{y}}$ is the predicted class distribution. This loss quantifies the discrepancy between predicted and actual labels, guiding the optimization process. Each component of ResNet-18 plays a crucial role: early layers extract fine details, intermediate layers build on them to detect more complex structures, and deeper layers identify high-level semantic features. Residual connections ensure effective training of this deep network by maintaining gradient flow, while global average pooling and softmax enable compact and interpretable final predictions. This architecture achieves a balance between **depth**, **expressiveness**, and **efficiency**, making it a powerful backbone for a wide range of vision tasks.

Unlike early neural network models such as Lenet-5, ResNet-18 is a considerably deeper and more sophisticated architecture, capable of learning rich hierarchical representations. It consistently achieves high accuracy across diverse domains, including medical imaging, autonomous driving, industrial inspection, and natural image classification. Its success stems from both its architectural depth and the use of residual connections, which address vanishing gradients and enable effective training of deeper networks.

Due to its strong performance and wide deployment in real-world, safety-critical applications, ResNet-18 is an important subject for robustness evaluation. Assessing its behavior under input perturbations, such as noise, occlusions, or adversarial attacks, is essential for understanding its reliability beyond ideal conditions. This motivates its selection as a testbed

in this thesis, to analyze how even high-performing models may degrade under challenging or non-standard inputs, and to explore methods for enhancing their resilience.

4.3 Training and Implementation Setup

Table 4.1 presents a detailed comparison of the models used in this study, including their architectural structures, training configurations, and preprocessing requirements. The table highlights key aspects such as the number of layers, along with the specific training settings (e.g., batch size) and preprocessing steps (e.g., data normalization, resizing) applied to each model and training dataset. Given the structure and dimensionality of the MNIST and Fashion-MNIST datasets, these tailored preprocessing and training strategies were essential to ensure a fair and consistent evaluation across models with varying complexities and assumptions.

Table 4.1: Comparison of traditional ML and CNN models used in this study.

Model	**Type**	**Layers / Params**	**Epochs / Batch**	**Preprocessing**
Logistic Regression	Traditional	1 layer / ∼7k	–	Normalization
SVM (Linear)	Traditional	1 layer / ∼15k	–	Normalization
K-NN (k=5)	Traditional	Non-parametric	–	Normalization
Random Forest	Traditional	100 trees	–	None
MLP	Traditional	2 hidden layers / ∼530k	20	Normalization
Lenet-5	CNN	2 layers / ∼61k	10 / 128	Normalization
ResNet-18	CNN	18 layers / ∼11M	20 / 32	Normalization + Resize

4.3.1 Traditional ML Models

For traditional machine learning models, each image from the MNIST and Fashion-MNIST datasets was flattened into a 784-dimensional vector and standardized to have zero mean and unit variance. Given the structure and dimensionality of the datasets, these preprocessing

were crucial for ensuring a fair and consistent evaluation across traditional models. Logistic Regression was trained using multinomial loss with ℓ_2 regularization via `scikit-learn`'s `LogisticRegression`, modeling all 10 output classes in a single optimization framework. For Support Vector Machines (SVM), we employed an RBF kernel to capture nonlinear patterns, using `SVC` from `scikit-learn` with default hyperparameters ($C = 1.0$). Both models serve as interpretable baselines with minimal architectural complexity.

The k-Nearest Neighbors (K-NN) model was configured with $k = 5$, balancing locality and robustness to noise, using Euclidean distance for similarity. Inputs were normalized and standardized, ensuring pixel values did not disproportionately affect distance metrics. Random Forest classifiers were built using 100 decision trees, trained on standardized features, and configured to run in parallel for efficiency. For the Multi-layer Perceptron (MLP), we used a fully connected architecture with two hidden layers of 512 and 256 neurons, respectively, both employing ReLU activation and a 20% dropout rate to prevent overfitting. Training was conducted for 20 epochs using the Adam optimizer and categorical cross-entropy loss. Model weights and preprocessing scalers were saved and reused across corruption tests to ensure consistency.

4.3.2 CNN Models

For the CNN models, input preprocessing varied based on architecture requirements. Lenet-5, originally designed for digit recognition, required images to be padded from 28×28 to 32×32 pixels and reshaped to $(32, 32, 1)$. These grayscale images were normalized to $[0, 1]$. The model followed its canonical architecture: two convolutional layers with 6 and 16 filters (5×5 kernels, `tanh` activations), each followed by average pooling layers (2×2, stride 2), feeding into two fully connected layers (120 and 84 neurons) and a 10-way softmax output. Lenet-5 was trained using Adam optimizer and categorical cross-entropy loss for 10 epochs with a batch size of 128.

In contrast, ResNet-18 required higher-dimensional input. To meet its input specifications, each MNIST and Fashion-MNIST image was resized to 224×224 pixels and converted

to RGB format using grayscale replication. The model was implemented using TensorFlow and trained with the initialization for the convolutional layers. Training was conducted for up to 20 epochs using the Adam optimizer (learning rate 10^{-3}), a batch size of 32, and early stopping with a patience of three epochs.[1]

This setup ensured that the training process was computationally efficient and regularized.

[1]This means that if the model's performance (such as validation loss or accuracy) does not improve for three consecutive epochs, the training will be stopped early to prevent overfitting and save computation time.

5. *Experimental Design and Analysis*

5.1 Research Question 1

How do CNNs and traditional ML models differ in handling different natural corruptions?

5.1.1 Research Question Significance

In this study, we aim to systematically evaluate model robustness when exposed to real-world disturbances, such as noise, blurriness, or distortions. This is particularly important in practical applications where inputs are often degraded—like camera feeds affected by weather, motion blur, or low-quality images from embedded vision systems. In these cases, maintaining reliable performance is more crucial than achieving high accuracy on clean data alone.

In automotive applications, RGB cameras are commonly used due to their low cost and scalability. However, they face challenges in maintaining clear and accurate image capture across varying lighting conditions, such as transitioning from bright daylight to tunnel shadows. Another critical issue is handling motion blur caused by the relative movement between the vehicle and surrounding objects (Sayed and Brostow (2021) .

Ideally, models should maintain high accuracy across both clean and noisy conditions to ensure reliability. If not, we may prefer a model with a better average performance or one that excels in handling specific noise types relevant to the environment. For instance, in automotive scenarios, a model that maintains 90% accuracy under both clean and blurry conditions is preferable to one with 96% accuracy on clean data but only 86% when blurry,

as blur is a more frequent and critical challenge. Understanding which model performs best under specific types of natural corruptions is essential for designing robust systems. Knowing which model is more reliable in a particular noisy situation can guide the development of effective ensemble models or support dynamic model selection to increase robustness. Moreover, if a traditional ML model proves to be more resilient to certain corruptions compared to CNNs, we could leverage this insight to reduce computational cost and improve interpretability by opting for simpler models when appropriate. This not only enhances system efficiency but also allows for faster and more interpretable solutions in real-world applications.

Despite the growing interest in robust machine learning, existing research primarily focuses on deep learning models, such as CNNs, while often neglecting traditional ML approaches. Moreover, robustness studies often rely on simple metrics, such as accuracy and F1-score on clean datasets, without providing a comprehensive evaluation across different types of noise and robustness metrics. This limited focus fails to capture the stability and consistency of predictions under real-world disturbances, leaving a gap in understanding how traditional models compare to more complex CNN architectures when handling input degradations.

To address this gap, we design a systematic evaluation experiment that directly compares the robustness of CNNs and traditional ML models. Our approach goes beyond traditional performance metrics by assessing prediction stability across a wide range of natural corruptions. By doing so, we explore the robustness-efficiency trade-off, investigating whether traditional models can offer comparable robustness to CNNs despite their lower complexity.

Our evaluation involves extensive experiments where we uniformly apply diverse natural corruptions to both model types. We use comprehensive metrics to assess not only accuracy but also performance under varying corruption levels. This approach allows us to rank models based on their resilience, guiding the selection of architectures that ensure reliable performance in real-world applications.

5.1.2 Methodology for RQ1

To quantitatively assess the robustness of convolutional neural networks (CNNs) versus traditional machine learning (ML) models under natural corruptions, we construct a controlled evaluation framework consisting of 7 classifiers trained and tested on 12 image distributions per dataset 3.2. This section directly addressing the practical significance of robust classification.

Dataset and Pre-proceccing

The MNIST and Fashion MNIST datasets are utilized, each containing 60,000 training and 10,000 test images. The official `train` and `test` splits, used for model fitting, and evaluation. No additional hold-out or validation subsets are created to maintain a controlled comparison. All images are single-channel greyscale with a resolution of 28×28 pixels.

Let $\mathcal{D}_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{60000}$ and $\mathcal{D}_{\text{test}} = \{(x_j, y_j)\}_{j=1}^{10000}$ denote the training and test sets for the MNIST and Fashion-MNIST datasets, where $x_i \in \mathbb{R}^{28\times28}$ and $y_i \in \{0, 1, \ldots, 9\}$. Each pixel value x_{ij} is normalized by dividing by 255, such that $x'_{ij} \in [0, 1]$. No additional preprocessing such as data augmentation or rebalancing is applied to ensure comparability across models. Detailed descriptions of the model architectures and training procedures are provided in Sections 4 and 4.3.

Model Training

We train 7 models—5 traditional ML and 2 CNNs—using fixed training data and default hyperparameters recommended in their original references or software packages (scikit-learn developers (2024), TensorFlow Authors (2024) to isolate architectural effects.

Let $\mathcal{M} = \{M_1, M_2, \ldots, M_7\}$ be the set of models, where:

- M_1–$M_5 \in \mathcal{M}_{\text{traditional}}$: Logistic Regression, Support Vector Machine, K-Nearest Neighbors, Random Forest, and Multi-Layer Perceptron (MLP)
- M_6–$M_7 \in \mathcal{M}_{\text{CNN}}$: Lenet-5 and ResNet-18

Each model M_k is trained using only $\mathcal{D}_{\text{train}}$ and initialized with a fixed random seed $s = 42$. Model-specific parameters θ_k are optimized using the default loss functions and solvers provided in their respective open-source implementations (`scikit-learn` for M_1–M_5 and `TensorFlow/Keras` for M_6–M_7). No manual hyperparameter tuning or validation set is used. The test set is strictly held out during training.

Corruption Benchmark

To evaluate robustness, we generate corrupted versions of the `test` split, while keeping the training distribution unchanged. We apply eleven distinct corruption types (detailed in Section 3.2), conservatively chosen severity level that preserves the original image ground-truth label. This process results in 11 corrupted test sets for MNIST and 11 for Fashion-MNIST, each containing 10,000 images, in addition to the original clean test set—yielding a total of 12 evaluation sets per dataset and 120,000 test images in total.

Let $\mathcal{C} = \{c_1, c_2, \ldots, c_{11}\}$ denote the set of 11 natural corruption functions, where each $c_k : \mathbb{R}^{28\times28} \to \mathbb{R}^{28\times28}$ applies a unique distortion (3.2). For each test image $x_j \in \mathcal{D}_{\text{test}}$, a set of corrupted versions is generated:

$$x_j^{(k)} = c_k(x_j), \quad \forall k \in \{1, \ldots, 11\}$$

The evaluation dataset includes:

- $\mathcal{D}_{\text{test}}^{(0)}$: original clean test set
- $\mathcal{D}_{\text{test}}^{(k)} = \{(x_j^{(k)}, y_j)\}_{j=1}^{10000}$: the corrupted versions for each $k \in \{1, \ldots, 11\}$

This yields 12 evaluation subsets per dataset.

Evaluation Protocol and Metrics

Each saved model is evaluated on the clean test set as well as on all 11 corrupted variants of the test datasets.

For each model M_k and each version $\mathcal{D}_{\text{test}}^{(m)}$, predictions are computed as:

$$\hat{y}_{j,k}^{(m)} = M_k(x_j^{(m)}; \theta_k) \quad \text{for } j = 1, \ldots, 10000,\ m = 0, \ldots, 12$$

For each dataset, we calculate performance and stability metrics. We use classical metrics like accuracy, precision, recall, and F1-score to measure classification quality. To assess stability under corruption, we include the average flip rate, which shows how often predictions change due to corruptions, and label variation, which counts the number of unique predicted labels per input across different corruptions.

5.1.3 Evaluation Metrics

Accuracy

Accuracy measures the proportion of test samples for which the predicted class exactly matches the true label, reflecting the overall reliability of a classifier on both clean and corrupted inputs. In multi-class classification problems like MNIST or Fashion MNIST, accuracy counts how often the predicted label is correct, regardless of the class. It is formally defined as

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} = \frac{\sum_{i=1}^{C} TP_i}{N},$$

where C is the number of classes, TP_i is the number of correctly predicted samples for class i, and N is the total number of samples.

Accuracy is widely used as a baseline metric due to its simplicity and interpretability (LeCun et al. (1998, 2010b).

Precision

Precision measures the correctness of positive predictions for each class individually, answering: "Of all samples predicted as class i, how many truly belong to class i?" In multi-class

settings, precision is computed per class treating that class as positive and all others as negative, then aggregated (weighted by class frequency). Its formula is

$$\text{Precision}_{\text{weighted}} = \sum_{i=1}^{C} \left(\frac{n_i}{n} \cdot \frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i} \right)$$

where TP is the count of correctly predicted samples of class i, and FP is the count of samples incorrectly predicted as class i. n_i is the number of true instances (support) for class i and n is the total number of instances. C is the total number of classes (10).

Precision is particularly important when false positives are costly or misleading, such as distinguishing visually similar classes. We include weighted precision to evaluate the quality of positive class predictions across all classes.

Recall

Recall, or sensitivity, measures the ability to correctly identify all true instances of each class, answering: "Of all samples truly belonging to class i, how many did the model correctly predict?" It is defined as

$$\text{Recall}_{\text{weighted}} = \sum_{i=1}^{C} \left(\frac{n_i}{n} \cdot \frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i} \right)$$

where FN is the count of samples of class i missed by the model. Also, n_i is the number of true instances for class i and n is the total number of instances. C is the total number of classes (10).

Like precision, recall is computed per class and aggregated with weighting. Recall is critical when missing true class instances is costly, for example, ensuring all "Sneaker" images are recognized in Fashion MNIST. We use recall to assess model sensitivity across classes, especially under corrupted inputs.

F1-score

The F1-score combines precision and recall into a single balanced metric, calculated as the harmonic mean:

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

This metric penalizes large imbalances between precision and recall, providing a robust measure of classification effectiveness, especially for imbalanced classes. We selected the weighted F1-score to summarize overall class-wise performance, balancing false positives and false negatives.

Average Flip Rate

Average Flip Rate measures how often the model's predicted labels change when the input is corrupted compared to clean inputs, (Hendrycks and Dietterich (2019). It is calculated as the mean proportion of samples whose predicted labels differ between clean and corrupted versions. Formally, for a dataset of N samples, it is defined as

$$\text{Average Flip Rate} = \frac{1}{K}\sum_{k=1}^{K}\left(\frac{1}{N}\sum_{j=1}^{N}\mathbb{I}(\hat{y}_j^{clean} \neq \hat{y}_j^{corrupted_k})\right),$$

where $\hat{y}_j^{clean}$ and $\hat{y}_j^{corrupted}$ denote the predicted labels for the j-th sample on clean and corrupted inputs respectively, K is number of corruption types (maximum 11), and $\mathbb{I}(\cdot)$ is the indicator function that equals 1 if the condition is true and 0 otherwise.

This metric quantifies prediction stability by capturing how sensitive the model is to input perturbations: a lower Average Flip Rate indicates more consistent predictions and greater robustness to noise or corruption. While the exact term Average Flip Rate is not standardized, similar concepts measuring prediction consistency or flip points have been explored in robustness and interpretability literature (Hendrycks and Dietterich (2019), Yousefzadeh and O'Leary (2020). It complements traditional accuracy-based robustness metrics by focusing on prediction consistency rather than correctness alone.

Label Variation

Label Variation quantifies the average number of unique predicted labels assigned to each input sample across all corrupted variants of the dataset. Formally, given K corrupted versions and N samples, it is defined as

$$\text{Label Variation} = \frac{1}{N}\sum_{j=1}^{N}\left|\bigcup_{k=1}^{K}\{\hat{y}_j^{corrupted_k}\}\right|,$$

where $\hat{y}_j^{corrupted_k}$ is the predicted label for the j-th sample under the k-th corruption, and $|\cdot|$ denotes the number of unique predicted labels for that sample.

This metric measures prediction consistency across corruptions: a lower Label Variation indicates that the model tends to assign the same label to an input despite different corruptions, reflecting greater stability and robustness. The range of Label Variation is from 1 (perfect stability, same predicted label across all corruptions) up to the total number of classes M (maximum instability, with predictions varying across all possible classes). Values closer to 1 are desirable.

While Label Variation is a novel metric, it captures an important aspect of prediction stability by measuring the diversity of predicted labels under corruptions. This complements more common robustness metrics like Average Flip Rate and aligns with recent research emphasizing prediction consistency and annotation variability (Hendrycks and Dietterich (2019), Pham et al. (2023), Yousefzadeh and O'Leary (2020).

5.1.4 RQ1 Results

5.1.4.1 MNIST Results

As you can see in Table 5.1, the comparison of different classification models in terms of accuracy on the MNIST test dataset shows several insights into their robustness. As expected, deep learning models dominate the leaderboard, with ResNet-18 consistently delivering top-tier accuracy across almost all datasets. Impressively, ResNet-18 achieves near-perfect accuracy on the clean data (99.36%) and maintains remarkable resilience under challenging distortions like Elastic Deformation (92.69%) and Gaussian Blur (98.84%). This confirms its strong generalization ability even in the presence of complex image transformations.

Interestingly, Lenet-5 emerges as a surprisingly robust contender, outperforming even

ResNet-18 in certain scenarios such as Gaussian Noise (96.23%) and Motion Blur (98.02%). This suggests that simpler convolutional architectures still hold significant value, especially when dealing with specific noise types or moderate distortions. Furthermore, Lenet-5 leads in the Mixed distortion setting with an accuracy of 92.62%, indicating its versatility and adaptability to combined real-world corruptions. This is a nice, unexpected, and interesting result, likely because Lenet-5's relatively shallow architecture focuses on more generalizable and stable features, making it less sensitive to complex noise patterns that can disrupt deeper, more intricate models.

Random Forest and MLP models show moderate resilience overall, with Random Forest excelling in handling Salt Pepper Noise (96.78%), which is quite intriguing. This suggests that ensemble methods like Random Forest can effectively aggregate multiple decision boundaries to filter out sparse, high-intensity noise such as salt and pepper artifacts. The robustness of Random Forest in this scenario may stem from its inherent ability to model non-linearities and isolate noisy features through voting among trees, a strength not shared by simpler models like Logistic Regression or SVM.

On the other hand, classical machine learning models such as Logistic Regression, SVM, and K-NN experience dramatic drops in accuracy under corruptions, often falling below 15% in cases like Brightness Contrast and Gaussian Noise.

Overall, while deep learning models typically excel on clean data and many types of distortions, our results show that greater architectural complexity does not always guarantee improved robustness. This phenomenon is likely related to the inherent simplicity of the dataset. In relatively simple datasets like MNIST, highly complex models such as ResNet-18 tend to rely on subtle and fine-grained image patterns to achieve maximum accuracy. However, this reliance can make them more sensitive to distortions and noise, thereby reducing their robustness in certain real-world noisy conditions.

Therefore, achieving near-perfect accuracy with a highly complex model does not necessarily guarantee practical robustness, especially when dealing with imperfect or corrupted data. This underscores the importance of finding the right balance between model complex-

ity, generalization, and robustness, rather than focusing solely on maximizing accuracy on clean datasets.

Table 5.1: Accuracy of Different Models Across Datasets (Max in Bold)

Dataset	Logistic Regression	SVM	K-NN	Random Forest	MLP	Lenet-5	ResNet-18
Clean	0.9216	0.9661	0.9443	0.9704	0.9780	0.9843	**0.9936**
Brightness Contrast	0.1135	0.1028	0.1010	0.2531	0.1032	**0.6787**	0.2009
Elastic Deformation	0.4746	0.4676	0.5097	0.7331	0.5760	0.8115	**0.9269**
Gaussian Blur	0.8852	0.8813	0.9224	0.6097	0.9232	0.9772	**0.9884**
Gaussian Noise	0.1726	0.1028	0.1370	0.7115	0.1128	**0.9623**	0.6657
JPEG Compression	0.6950	0.2786	0.9382	0.9040	0.3990	0.9840	**0.9933**
Motion Blur	0.8707	0.8633	0.9224	0.7332	0.9110	**0.9802**	0.9769
Poisson Noise	0.9183	0.9658	0.9444	0.9688	0.9776	0.9840	**0.9936**
Random Occlusion	0.8713	0.9394	0.9319	0.9376	0.9395	0.9457	**0.9596**
Speckle Noise	0.9182	0.9662	0.9444	0.9690	0.9773	0.9846	**0.9935**
Salt Pepper Noise	0.2156	0.1080	0.1310	**0.9678**	0.1806	0.9542	0.8312
Mixed	0.6118	0.5612	0.6484	0.7789	0.6084	**0.9262**	0.8512

Following the accuracy analysis, the precision results shown in Table 5.2 reveal similar patterns across different models and datasets. As expected, ResNet-18 consistently achieves high precision on clean data (99.36%) and challenging distortions like Elastic Deformation (92.86%) and Gaussian Blur (98.85%).

Interestingly, Lenet-5 shows notable precision improvements in noise conditions where it previously demonstrated accuracy advantages, such as Gaussian Noise (96.67%) and Brightness Contrast (88.18%). Additionally, Random Forest maintains its strong performance with Salt Pepper Noise (96.78%), reinforcing its robustness in handling this specific type of distortion.

Overall, the precision scores in most cases follow the same trend as accuracy scores, indicating that the models' ability to correctly identify positive instances is closely related to their general performance on clean and noisy data for the MNIST dataset, which has balanced data across each class.

Table 5.2: Precision of Different Models Across Datasets (Max in Bold)

Dataset	**Logistic Regression**	**SVM**	**K-NN**	**Random Forest**	**MLP**	**Lenet-5**	**ResNet-18**
Clean	0.9215	0.9663	0.9444	0.9704	0.9780	0.9844	**0.9936**
Brightness Contrast	0.0129	0.0106	0.0102	0.4312	0.0107	**0.8818**	0.5143
Elastic Deformation	0.4969	0.6394	0.5604	0.7554	0.6201	0.8194	**0.9286**
Gaussian Blur	0.8884	0.9112	0.9255	0.8450	0.9332	0.9777	**0.9885**
Gaussian Noise	0.1766	0.0106	0.2782	0.8140	0.1323	**0.9667**	0.8545
JPEG Compression	0.7671	0.7977	0.9394	0.9172	0.5817	0.9841	**0.9933**
Motion Blur	0.8761	0.9042	0.9251	0.8637	0.9215	**0.9804**	0.9773
Poisson Noise	0.9181	0.9660	0.9446	0.9688	0.9776	0.9841	**0.9936**
Random Occlusion	0.8720	0.9401	0.9323	0.9383	0.9403	0.9469	**0.9606**
Speckle Noise	0.9180	0.9663	0.9445	0.9690	0.9773	0.9847	**0.9935**
Salt Pepper Noise	0.2307	0.7443	0.1720	**0.9678**	0.2191	0.9578	0.8991
Mixed	0.6992	0.8447	0.7153	0.8590	0.7152	**0.9349**	0.9006

As shown in Table 5.3, ResNet-18 achieves the highest recall on clean data (99.36%), confirming its strong capability to correctly identify positive instances in uncorrupted settings. However, its performance drops significantly in noisy conditions, particularly in the Brightness Contrast (20.09%) and Gaussian Noise (66.57%) scenarios.

Interestingly, Lenet-5 demonstrates superior recall in these challenging cases, with 67.87% for Brightness Contrast and 96.23% for Gaussian Noise. This suggests that simpler architectures may retain more stable feature representations under noise, while more complex models like ResNet-18 may become sensitive to subtle distortions.

For structural distortions such as Elastic Deformation (92.69%) and Motion Blur (97.69%), ResNet-18 outperforms other models, leveraging its deeper architecture to capture complex spatial relationships. However, in the Mixed distortion scenario, Lenet-5 achieves higher recall (92.62%), indicating that less complex models can offer more consistent performance when facing combined noise types.

Notably, Random Forest shows excellent recall under Salt Pepper Noise (96.78%), reflecting the robustness of ensemble methods to sparse, high-intensity noise. In contrast,

traditional models like Logistic Regression and SVM generally show low recall under most distortions, highlighting their limitations in handling noisy data.

Table 5.3: Recall of Different Models Across Datasets (Max in Bold)

Dataset	**Logistic Regression**	**SVM**	**K-NN**	**Random Forest**	**MLP**	**Lenet-5**	**ResNet-18**
Clean	0.9216	0.9661	0.9443	0.9704	0.9780	0.9843	**0.9936**
Brightness Contrast	0.1135	0.1028	0.1010	0.2531	0.1032	**0.6787**	0.2009
Elastic Deformation	0.4746	0.4676	0.5097	0.7331	0.5760	0.8115	**0.9269**
Gaussian Blur	0.8852	0.8813	0.9224	0.6097	0.9232	0.9772	**0.9884**
Gaussian Noise	0.1726	0.1028	0.1370	0.7115	0.1128	**0.9623**	0.6657
JPEG Compression	0.6950	0.2786	0.9382	0.9040	0.3990	0.9840	**0.9933**
Motion Blur	0.8707	0.8633	0.9224	0.7332	0.9110	**0.9802**	0.9769
Poisson Noise	0.9183	0.9658	0.9444	0.9688	0.9776	0.9840	**0.9936**
Random Occlusion	0.8713	0.9394	0.9319	0.9376	0.9395	0.9457	**0.9596**
Speckle Noise	0.9182	0.9662	0.9444	0.9690	0.9773	0.9846	**0.9935**
Salt Pepper Noise	0.2156	0.1080	0.1310	**0.9678**	0.1806	0.9542	0.8312
Mixed	0.6118	0.5612	0.6484	0.7789	0.6084	**0.9262**	0.8512

The F1 scores in Table5.4 across different datasets highlight the models' ability to maintain a balance between precision and recall, especially under varying data distortions. Generally, deep learning models such as Lenet-5 and ResNet-18 demonstrate superior performance with consistently higher F1 scores compared to classical machine learning models like Logistic Regression, SVM, and K-NN.

For clean data, all models perform well, with ResNet-18 achieving the highest F1 score. However, in challenging conditions, such as Brightness Contrast and Gaussian Noise, traditional models experience a sharp drop in F1, while Lenet-5 and ResNet-18 maintain comparatively stable performance.

In datasets involving structured distortions like Elastic Deformation and Motion Blur, models like Random Forest and MLP show moderate robustness, but deep learning models still outperform. The significant variation in F1 scores among models under noise and distortion reflects the difference in their ability to generalize and adapt to challenging data

conditions.

Table 5.4: F1 Scores of Different Models Across Datasets (Max in Bold)

Dataset	**Logistic Regression**	**SVM**	**K-NN**	**Random Forest**	**MLP**	**Lenet-5**	**ResNet-18**
Clean	0.9215	0.9661	0.9441	0.9704	0.9780	0.9843	**0.9936**
Brightness Contrast	0.0231	0.0192	0.0185	0.2002	0.0193	**0.6690**	0.0807
Elastic Deformation	0.4715	0.4982	0.5079	0.7333	0.5804	0.8123	**0.9268**
Gaussian Blur	0.8856	0.8843	0.9222	0.6196	0.9228	0.9773	**0.9884**
Gaussian Noise	0.1053	0.0192	0.1120	0.6800	0.0660	**0.9628**	0.6846
JPEG Compression	0.6800	0.2954	0.9382	0.9053	0.4143	0.9840	**0.9933**
Motion Blur	0.8713	0.8686	0.9221	0.7517	0.9104	**0.9802**	0.9768
Poisson Noise	0.9181	0.9658	0.9442	0.9688	0.9776	0.9840	**0.9936**
Random Occlusion	0.8712	0.9393	0.9316	0.9374	0.9395	0.9458	**0.9595**
Speckle Noise	0.9180	0.9662	0.9442	0.9690	0.9773	0.9846	**0.9935**
Salt Pepper Noise	0.1861	0.0299	0.1193	**0.9678**	0.1720	0.9546	0.8427
Mixed	0.6261	0.6269	0.6619	0.7940	0.6336	**0.9276**	0.8608

Table 5.5 shows the average flip rate and label variation for each model, indicating prediction stability under input perturbations. Lenet-5 achieves the lowest flip rate (0.1409) and label variation (2.37), reflecting its robustness and consistent predictions, especially compared to more complex models like ResNet-18 (0.2133, 2.74).

Traditional models such as Logistic Regression (0.4094), SVM (0.4603), and K-NN (0.3805) exhibit higher flip rates, indicating sensitivity to minor input changes. Similarly, MLP shows a relatively high flip rate (0.4287), while Random Forest demonstrates moderate stability (0.2707, 2.86), likely due to its ensemble structure.

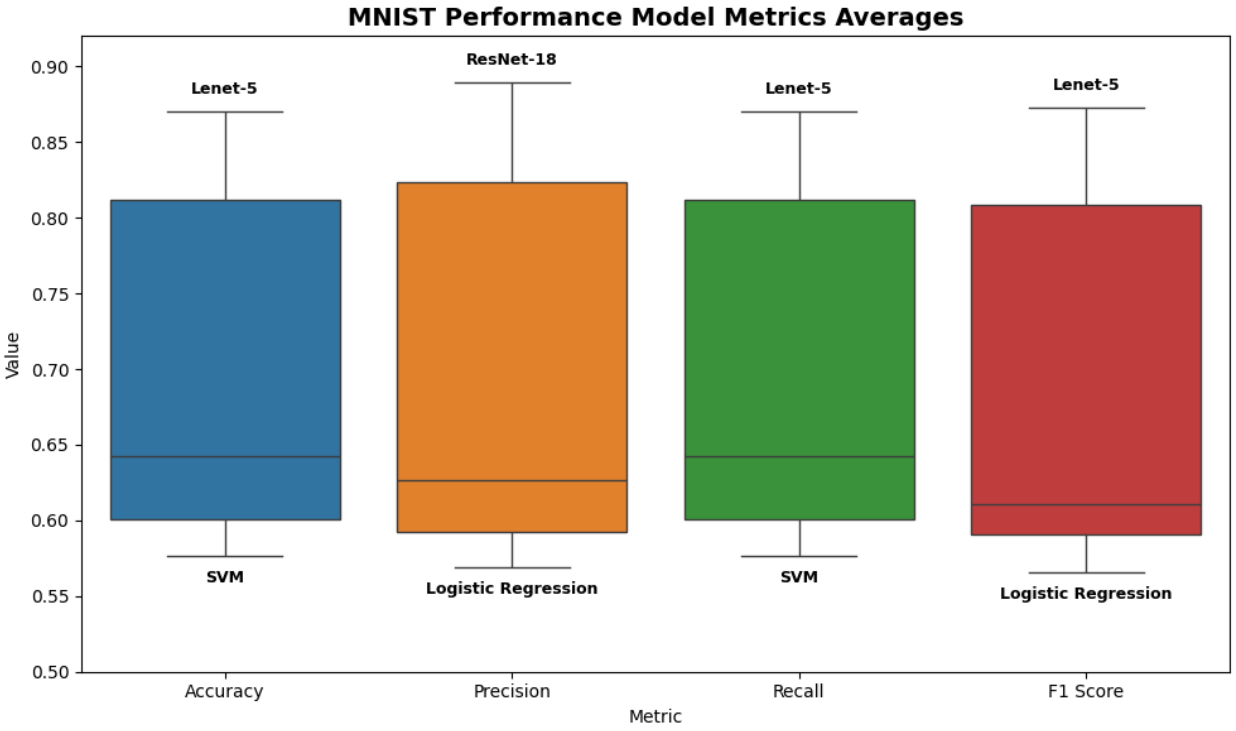


Figure 5.1: Comparison of average Model Performance Metrics on the MNIST Datasets

Table 5.5: Flip Rate and Label Variation Summary for MNIST

Model	Average Flip Rate	Label Variation
Logistic Regression	0.4094	3.64
SVM	0.4603	2.6
K-NN	0.3805	4.05
Random Forest	0.2707	2.86
MLP	0.4287	3.79
Lenet-5	**0.1409**	**2.37**
ResNet-18	0.2133	2.74

The comprehensive evaluation of multiple classification models on the MNIST dataset reveals key insights into their accuracy, robustness, and stability across various noise and distortion conditions. As anticipated, deep learning models—particularly ResNet-18—consistently

achieve the highest accuracy and precision on clean data and many distortion types, such as Elastic Deformation and Gaussian Blur, demonstrating superior generalization and feature extraction capabilities from complex patterns.

However, an intriguing and somewhat unexpected finding is the robust performance of the simpler Lenet-5 architecture under several noise conditions, including Gaussian Noise, Brightness Contrast, and Mixed distortions. Lenet-5 frequently outperforms ResNet-18 in these cases, suggesting that less complex models can retain more stable, generalized features and are less prone to overfitting subtle data perturbations. This is further supported by Lenet-5's lowest flip rate and label variation, indicating greater prediction stability under input perturbations.

Classical machine learning models like Logistic Regression, SVM, and K-NN show considerable vulnerability to corruptions, often dropping below 20% accuracy in difficult noise scenarios. Random Forest stands out among them by effectively handling Salt Pepper Noise, likely due to its ensemble nature that aggregates decisions from multiple trees, offering resilience to sparse, high-intensity noise.

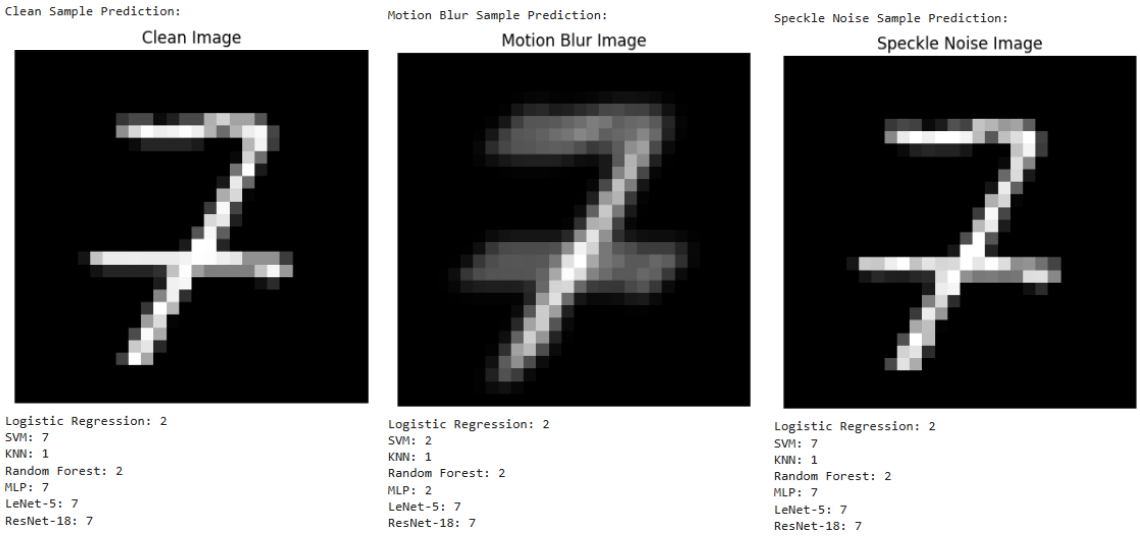
From the boxplot analysis 5.1, it is clear that the range between the minimum and maximum average performance metrics across all datasets varies from approximately 65% to 96%. This wide variation highlights the critical importance of selecting an appropriate model for each specific task, as it can significantly influence the final outcomes.

The results also showed that for MNIST, a relatively simple image dataset, increased architectural complexity does not always translate to better robustness. Instead, a balance between model complexity and generalization is critical.

In practical terms, ResNet-18 is preferred when achieving the highest possible accuracy on clean or mildly distorted data is the goal. Conversely, Lenet-5 is better suited for real-world scenarios where mixed or severe noise corruptions are present, offering more dependable performance and stability.

Overall, the MNIST experiments highlight the importance of selecting models not solely based on peak accuracy but also considering robustness and stability metrics, especially for

deployment in environments where data quality cannot be guaranteed.



(a) Clean sample (b) Motion blur sample (c) Speckle noise sample

Figure 5.2: Samples of MNIST images: (a) clean, (b) with motion blur, and (c) with speckle noise.

Figure 5.2 demonstrates how various machine learning models respond to a specific MNIST digit under three conditions: a clean image, a motion-blurred version, and a speckle noise–corrupted version. The same digit is shown in each condition to highlight how model predictions vary when different types of corruption are introduced. Predictions are compared across all models traditional models and deep learning models. This visualization supports the broader thesis analysis by showcasing model behavior under two distinct corruptions, motion blur and speckle noise, drawn from the larger set of distortions studied.

5.1.4.2 FASHION MNIST Results

For each dataset and corruption type, the table 5.6 displays the accuracy scores achieved by each model. The highest accuracy value for each row is highlighted in bold to indicate the model that performs best under that specific condition.

Notably, ResNet-18 demonstrates superior performance by achieving the highest accuracy in 6 out of the 12 scenarios (including the Clean dataset). This highlights ResNet-18's

robustness, especially in complex data conditions. In contrast, Lenet-5 outperforms the other models in 5 scenarios, particularly in cases involving Gaussian Blur, Gaussian Noise, and Mixed corruptions. The Random Forest model only excels in one scenario, specifically Salt Pepper Noise.

This comparison underscores the resilience of deep learning architectures (ResNet-18 and Lenet-5) compared to traditional machine learning models when dealing with data corruption.

Table 5.6: Accuracy of Different Models Across Datasets (Max in Bold)

Dataset / Corruption	Logistic Regression	SVM	K-NN	Random Forest	MLP	Lenet-5	ResNet-18
Clean	0.8346	0.8836	0.8533	0.8760	0.8870	0.8856	**0.9292**
Brightness Contrast	0.2460	0.1000	0.0989	0.1207	0.1934	**0.5783**	0.5588
Elastic Deformation	0.4828	0.4805	0.5190	0.6503	0.5439	0.6144	**0.6884**
Gaussian Blur	0.7961	0.8145	0.7771	0.6782	0.8300	**0.8725**	0.7150
Gaussian Noise	0.3036	0.1337	0.3094	0.7658	0.4238	**0.8655**	0.5154
JPEG Compression	0.6633	0.6452	0.6984	0.8605	0.7941	0.8838	**0.8967**
Motion Blur	0.7821	0.7954	0.7592	0.6702	0.8131	**0.8602**	0.6801
Poisson Noise	0.8288	0.8827	0.8518	0.8736	0.8873	0.8851	**0.8915**
Random Occlusion	0.7999	0.8655	0.8417	0.8407	0.8543	0.8309	**0.8980**
Speckle Noise	0.8255	0.8830	0.8510	0.8766	0.8851	0.8815	**0.8918**
Salt Pepper Noise	0.3570	0.3255	0.5367	**0.8694**	0.5759	0.8480	0.7066
Mixed	0.6158	0.5922	0.6239	0.7225	0.6765	**0.8118**	0.7410

The second table 5.7, generally follows a similar trend as the first table (Accuracy Scores), with **ResNet-18** achieving the highest F1 score in 6 out of 12 scenarios and **Lenet-5** in 5 scenarios. However, some differences are observed.

For **Brightness Contrast**, although Lenet-5 remains the best-performing model, the gap between Lenet-5 (0.5526) and ResNet-18 (0.4913) is larger compared to the accuracy scores. This indicates that Lenet-5 maintains a relatively better balance between precision and recall under brightness variations.

For **Gaussian Noise**, Lenet-5 significantly outperforms ResNet-18 in both accuracy and

F1 scores, with the gap slightly widening in the F1 score. This suggests that Lenet-5's architecture is more resilient to random noise distortions.

In the case of **Salt Pepper Noise**, although Random Forest achieves the highest F1 score, the difference from Lenet-5 is smaller than in the accuracy comparison. This shows that Lenet-5's performance remains competitive in terms of precision and recall despite Random Forest's superior accuracy.

These differences highlight that F1 scores reveal variations in how models handle corruptions, particularly in scenarios with noise and brightness distortions.

Table 5.7: F1 Scores of Different Models Across Datasets (Max in Bold)

Dataset / Corruption	Logistic Regression	SVM	K-NN	Random Forest	MLP	Lenet-5	ResNet-18
Clean	0.8335	0.8829	0.8534	0.8745	0.8866	0.8866	**0.9288**
Brightness Contrast	0.1311	0.0182	0.0219	0.0512	0.0798	**0.5526**	0.4913
Elastic Deformation	0.4743	0.4812	0.5067	0.6484	0.5296	0.6147	**0.6816**
Gaussian Blur	0.7856	0.8072	0.7729	0.6618	0.8231	**0.8733**	0.7013
Gaussian Noise	0.2823	0.0818	0.3262	0.7707	0.4164	**0.8673**	0.5267
JPEG Compression	0.6458	0.6500	0.7016	0.8598	0.7847	0.8849	**0.8973**
Motion Blur	0.7732	0.7874	0.7542	0.6505	0.8045	**0.8610**	0.6676
Poisson Noise	0.8280	0.8821	0.8518	0.8721	0.8869	0.8861	**0.8922**
Random Occlusion	0.7985	0.8647	0.8416	0.8379	0.8530	0.8311	**0.8971**
Speckle Noise	0.8247	0.8824	0.8509	0.8753	0.8848	0.8825	**0.8924**
Salt Pepper Noise	0.3626	0.3658	0.5511	**0.8682**	0.5850	0.8497	0.7117
Mixed	0.6145	0.6308	0.6388	0.7333	0.6802	**0.8144**	0.7444

The precision scores in Table 5.8 generally align with previous results but reveal some nuanced differences. Notably, Lenet-5 shows a much higher precision than ResNet-18 under **Brightness Contrast**, indicating fewer false positives in this scenario. For **Gaussian Noise**, the precision gap between these two models narrows, suggesting ResNet-18 better manages false positives than reflected in accuracy or F1. In **Salt Pepper Noise**, Random Forest maintains the highest precision, with Lenet-5 closely behind, showing both models effectively handle positive predictions despite noise.

Table 5.8: Precision of Different Models Across Datasets (Max in Bold)

Dataset / Corruption	Logistic Regression	SVM	K-NN	Random Forest	MLP	Lenet-5	ResNet-18
Clean	0.8329	0.8829	0.8565	0.8749	0.8878	0.8888	**0.9288**
Brightness Contrast	0.1651	0.0100	0.1121	0.0368	0.0534	**0.7465**	0.7092
Elastic Deformation	0.4785	0.5844	0.5176	0.6803	0.5464	0.6322	**0.7045**
Gaussian Blur	0.7851	0.8171	0.7849	0.7301	0.8315	**0.8774**	0.7550
Gaussian Noise	0.4254	0.7338	0.6049	0.7978	0.5870	**0.8719**	0.7977
JPEG Compression	0.6837	0.7935	0.7563	0.8608	0.7946	0.8873	**0.8989**
Motion Blur	0.7728	0.8030	0.7686	0.7242	0.8150	**0.8673**	0.7144
Poisson Noise	0.8274	0.8820	0.8549	0.8723	0.8885	0.8882	**0.9006**
Random Occlusion	0.7973	0.8645	0.8449	0.8398	0.8546	0.8328	**0.8990**
Speckle Noise	0.8241	0.8822	0.8539	0.8758	0.8862	0.8846	**0.9001**
Salt Pepper Noise	0.3902	0.7549	0.6251	**0.8683**	0.6438	0.8538	0.8126
Mixed	0.6325	0.7652	0.6885	0.7632	0.7271	**0.8227**	0.7657

Recall scores in Table 5.9 show ResNet-18 leading in 6 cases and Lenet-5 in 5. Lenet-5 excels in **Brightness Contrast** and **Gaussian Noise**, while ResNet-18 performs best on **Elastic Deformation** and other noise types. Random Forest stands out in **Salt Pepper Noise**.

Table 5.9: Recall Scores of Different Models Across Datasets (Max in Bold)

Dataset / Corruption	Logistic Regression	SVM	K-NN	Random Forest	MLP	Lenet-5	ResNet-18
Clean	0.8346	0.8836	0.8533	0.8760	0.8870	0.8856	**0.9292**
Brightness Contrast	0.2460	0.1000	0.0989	0.1207	0.1934	**0.5783**	0.5588
Elastic Deformation	0.4828	0.4805	0.5190	0.6503	0.5439	0.6144	**0.6884**
Gaussian Blur	0.7961	0.8145	0.7771	0.6782	0.8300	**0.8725**	0.7150
Gaussian Noise	0.3036	0.1337	0.3094	0.7658	0.4238	**0.8655**	0.5154
JPEG Compression	0.6633	0.6452	0.6984	0.8605	0.7941	0.8838	**0.8967**
Motion Blur	0.7821	0.7954	0.7592	0.6702	0.8131	**0.8602**	0.6801
Poisson Noise	0.8288	0.8827	0.8518	0.8736	**0.8873**	0.8851	0.8915
Random Occlusion	0.7999	0.8655	0.8417	0.8407	0.8543	0.8309	**0.8980**
Speckle Noise	0.8255	0.8830	0.8510	0.8766	0.8851	0.8815	**0.8918**
Salt Pepper Noise	0.3570	0.3255	0.5367	**0.8694**	0.5759	0.8480	0.7066
Mixed	0.6158	0.5922	0.6239	0.7225	0.6765	**0.8118**	0.7410

Table 5.10 summarizes the *flip rate*—the frequency of prediction changes under input corruptions—and *label variation*—the diversity of predicted classes—for each model. Lenet-5 shows the lowest flip rate (0.1917) and label variation (2.66), indicating it maintains more stable and consistent predictions despite corruptions. In contrast, traditional models like Logistic Regression and SVM have higher flip rates and label variation, reflecting greater sensitivity and less robustness. The results highlight Lenet-5's resilience and ability to confidently handle corrupted inputs, likely due to its convolutional architecture effectively capturing invariant features.

The boxplot in Figure 5.3 reveals the distribution of model performance across various corruptions on the Fashion MNIST dataset. Among the models evaluated, ResNet-18 consistently achieves the highest median performance with low variability, underscoring its robustness and stability in handling corrupted data. Lenet-5 performs competitively, ranking just behind ResNet-18, though with slightly greater variance. In contrast, traditional models such as Logistic Regression and SVM exhibit lower median scores coupled with wider variability, indicating a higher sensitivity to input corruptions.

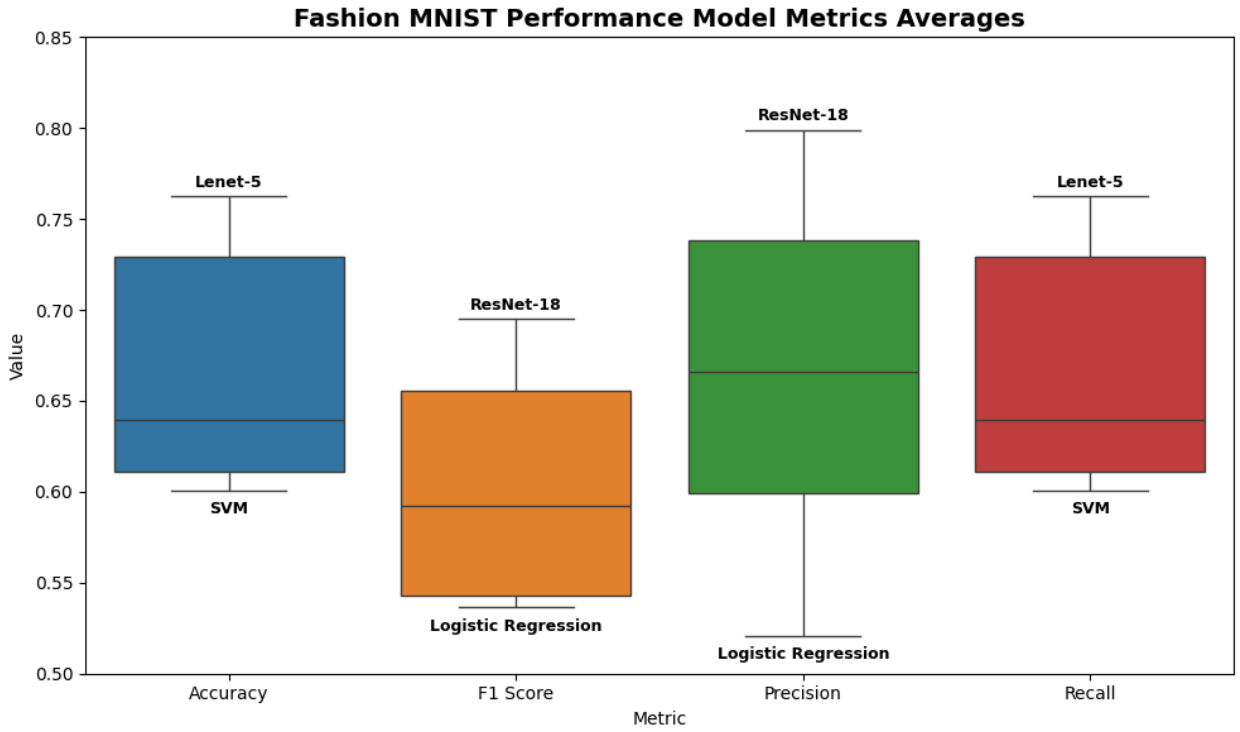


Figure 5.3: Comparison of average Model Performance Metrics on the Fashion MNIST Datasets

Table 5.10: Flip Rate and Label Variation Summary for Fashion MNIST

Model	Flip Rate	Label Variation
Logistic Regression	0.3911	4.01
SVM	0.4054	2.78
K-NN	0.362	3.84
Random Forest	0.2829	3.16
MLP	0.3315	3.35
Lenet-5	**0.1917**	**2.66**
ResNet-18	0.2928	3.22

When considering multiple evaluation metrics—accuracy, F1-score, precision, recall, and robustness indicators including flip rate and label variation—ResNet-18 leads in half of the tested scenarios, demonstrating strong generalizability across a broad range of corruptions. Lenet-5 dominates in nearly as many cases, particularly excelling in noise-heavy settings such as Gaussian Noise and Brightness Contrast, likely due to its convolutional architecture's aptitude for capturing local spatial distortions.

Overall, ResNet-18 stands out as the top-performing model, consistently delivering high scores and maintaining stability under challenging corruptions. Lenet-5 also proves resilient, achieving the lowest flip rate (0.1917) and label variation (2.66), which reflect its ability to make consistent predictions despite data degradation. The boxplot analysis further confirms the advantage of deep learning models, which demonstrate higher median performances and tighter interquartile ranges compared to traditional methods.

Interestingly, ensemble-based Random Forest outperforms deep models under Salt and Pepper Noise, showing approximately 15% higher accuracy and F1-score than ResNet-18. This suggests that certain traditional models may better handle specific types of impulse noise. Additionally, SVM and Logistic Regression deliver surprisingly strong precision in Gaussian Blur and Gaussian Noise conditions, occasionally rivaling or exceeding the performance of deep networks on these metrics.

(a) Clean sample (b) Motion blur sample (c) Speckle noise sample

Figure 5.4: Samples of Fashion MNIST images: (a) clean, (b) with motion blur, and (c) with speckle noise.

Figure 5.4 shows how different models classify a Fashion MNIST shirt (class 6) under three conditions: clean, motion blur, and speckle noise. The same item is used across all cases to highlight how corruption affects predictions across traditional and deep learning models. This example also reinforces patterns observed in the broader quantitative results. For instance, ResNet-18, which exhibited reduced accuracy under motion blur in the full evaluation, incorrectly classifies the blurred shirt image here. Similarly, SVM, K-NN, and Random Forest fail to identify the corrupted shirt correctly. This case visualizes the models' vulnerabilities and complements the overall analysis presented.

These findings highlight that while deep learning architectures generally offer superior robustness and stability, traditional machine learning models remain relevant, providing interpretable and computationally efficient alternatives in targeted scenarios. This nuanced understanding encourages a complementary approach to model selection based on the nature of data corruptions encountered.

5.1.5 RQ1 Conclusion

This study reveals clear differences between CNNs and traditional ML models in handling various natural corruptions on MNIST and Fashion MNIST datasets. Deep CNNs such as ResNet-18 generally achieve the highest accuracy on clean and structured distortions, while shallower CNNs like Lenet-5 demonstrate greater robustness and prediction stability under heavy noise conditions.

Among traditional models, Random Forest performs well on specific noise types such as Salt Pepper Noise but overall struggles with complex corruptions. Logistic Regression, SVM, and K-NN exhibit significant accuracy degradation under noise.

In summary, CNNs provide stronger overall performance and robustness, while traditional models maintain niche strengths depending on corruption type, highlighting the importance of model selection based on specific noise and distortion characteristics.

5.2 Research Question 2

How does uncertainty estimation relate to model robustness in CNNs and traditional ML models under natural image corruptions?

5.2.1 Research Question Significance

Knowing how CNNs and traditional machine learning models differ in confidence calibration and uncertainty measurement is very important, especially when these models are exposed to real-world problems like noisy or corrupted data. In many applications-such as self-driving cars or medical diagnosis-it's not enough for a model to simply make predictions; we also need to know how much we can trust those predictions. If a model is too confident about a wrong answer, it could lead to dangerous mistakes, like a car misreading a traffic sign or a medical system missing a disease.

If we ignore how well a model's confidence matches reality, we risk using systems that make errors without warning. For example, a CNN might give a very high confidence score to a misclassified, blurry image, while a traditional model might be more cautious and signal uncertainty. This difference matters because it can help us decide when to trust the model and when to ask for human help. Without understanding these differences, we could end up with unreliable systems that fail when faced with unexpected or poor-quality data.

Uncertainty metrics can be calculated in real time, as the model is making predictions. If we find that these metrics are good indicators of when the model is likely to be wrong-especially under different types of data corruption-we can use them to make smarter decisions in practice. For example, if a model shows low confidence on a corrupted image, an autonomous vehicle could slow down or ask for human input, reducing the risk of an accident.

By studying how confidence and uncertainty behave in both CNNs and traditional models under challenging conditions, we can choose or design models that are not just accurate, but also safe and reliable. This research question is significant because it helps bridge this gap.

5.2.2 Methodology for RQ2

To assess the differences in uncertainty between CNNs and traditional ML models, we followed a similar methodology to that used in Research Question 1. Using the previously trained and saved models, we evaluated each on the clean test set and the eleven previously generated corrupted datasets. For each evaluation, we computed two uncertainty metrics: the maximum predicted probability (Hendrycks and Gimpel (2017)) and the Gini index (Gini (1912)). We then averaged each metric across all corruption types for each model to enable a consistent comparison of uncertainty behavior under varying conditions.

5.2.3 Evaluation Metrics

Max Probability (Max-P)

Max-P Uncertainty measures the model's confidence in its predictions by evaluating the maximum predicted class probability for each input (Hendrycks and Gimpel (2017)). It is calculated as the mean of the maximum predicted probabilities across all samples.

Let:

- N be the number of test samples per corruption set ($N = 10{,}000$),
- C be the total number of classes ($C = 10$),
- K be the number of test sets ($K = 12$),
- $p_{j,k,c}^{(m)}$ denote the predicted probability assigned to class c by model M_k for the j-th sample in the m-th test set,

The Maximum Predicted Probability (Max-P) for a single sample is defined as:

$$\text{Max-P}_{j,k}^{(m)} = \max_{c \in \{1,\ldots,C\}} p_{j,k,c}^{(m)}$$

To compute the average Max Probability across all test samples and all test sets:

$$\text{Average Max-P} = \frac{1}{N} \sum_{j=1}^{N} Max - P_{j,k}^{(m)},$$

This metric quantifies the model's overall certainty in its predictions. A lower Average

Max-P indicates that the model is less confident and more uncertain across the dataset, while a higher value reflects more confident predictions. The minimum possible value is $1/k$ (when predictions are maximally uncertain and uniform across classes), and the maximum is 1 (when the model is completely certain in its predictions). Max-P is widely used in uncertainty estimation for its simplicity and interpretability.

Gini Index

The Gini Index, originally introduced by Corrado Gini in 1912 (Gini (1912)), quantifies the impurity or uncertainty in a set of predicted class probabilities for each input sample. In the context of machine learning, especially in classification tasks, it is widely used due to its computational efficiency and interpretability as a measure of how mixed or uncertain the model's predictions are.

Let:

- N be the number of test samples per corruption set ($N = 10{,}000$),
- C be the total number of classes ($C = 10$),
- K be the number of test sets ($K = 12$),
- $p_{j,k,c}^{(m)}$ denote the predicted probability assigned to class c by model M_k for the j-th sample in the m-th test set,

Formally, the Gini Index for a single sample is defined as:

$$\text{Gini}_{j,k}^{(m)} = 1 - \sum_{c=1}^{C} \left(p_{j,k,c}^{(m)} \right)^2$$

To compute the Average Gini Index across all test samples:

$$\text{Average Gini} = \frac{1}{N} \sum_{j=1}^{N} \text{Gini}_{j,k}^{(m)}$$

This metric captures the degree of uncertainty in the model's predictions: a lower Average Gini indicates more confident, pure predictions (with probability mass concentrated on a single class), while a higher value reflects greater uncertainty and class mixing. The minimum possible value is 0 (when predictions are completely certain), and the maximum is $1 - 1/C$

(when predictions are maximally uncertain and uniform across classes). The Gini Index is widely used in machine learning for its computational efficiency and interpretability as a measure of prediction uncertainty.

5.2.4 RQ2 Results

5.2.4.1 MNIST Results

Table 5.11: Gini Index Across Models and Corruptions (Min in bold)

Dataset	Logistic Regression	SVM	K-NN	Random Forest	MLP	Lenet-5	ResNet-18
Clean	0.1007	0.0548	0.0666	0.2631	0.0101	0.0138	**0.0075**
Brightness Contrast	**0.0**	0.792	0.72	0.8212	**0.0**	0.1016	0.224
Elastic Deformation	0.1664	0.4662	0.2552	0.6443	0.1056	0.1131	**0.0654**
Gaussian Blur	0.1236	0.1334	0.088	0.6718	0.0173	0.0213	**0.0161**
Gaussian Noise	0.0237	0.792	0.6808	0.7781	0.1793	**0.0348**	0.1934
JPEG Compression	0.131	0.6787	0.1305	0.708	0.1633	0.0149	**0.0093**
Motion Blur	0.1231	0.1523	0.0919	0.6453	0.0191	**0.0199**	0.0284
Poisson Noise	0.1071	0.0563	0.0672	0.2894	0.0109	0.0142	**0.0080**
Random Occlusion	0.1265	0.0797	0.0765	0.3361	**0.0220**	0.0359	0.0286
Speckle Noise	0.1057	0.0557	0.0671	0.2841	0.0105	0.0141	**0.0075**
Salt Pepper Noise	0.0431	0.7894	0.5571	0.4852	0.0976	**0.0396**	0.1155
Mixed	0.0945	0.4006	0.2714	0.5660	0.0632	**0.0416**	0.0703

Table 5.11 presents Gini Index values for models trained on MNIST clean data and tested on corrupted datasets, where lower values indicate greater prediction confidence. ResNet achieves the lowest Gini Index across most corruptions and clean data, demonstrating superior robustness compared to other architectures. This performance is attributed to its deep residual architecture, which enables learning abstract, hierarchical features that generalize effectively to unseen distortions.

MLP maintains relatively low uncertainty despite its simpler structure, suggesting that neural networks inherently generalize better to corrupted inputs even without explicit corruption-

augmented training. In contrast, classical models such as SVM and K-NN exhibit high Gini Index values under corruption, reflecting increased prediction uncertainty due to their reliance on handcrafted features and rigid decision boundaries.

Random Forest shows significant degradation under corrupted test conditions, likely stemming from overfitting to noise-free feature correlations during training. LeNet demonstrates intermediate robustness, outperforming classical models but underperforming relative to ResNet, highlighting the benefits of convolutional architectures while underscoring the importance of network depth for feature abstraction. These results collectively emphasize how architectural choices in deep learning models directly influence robustness to input perturbations.

Table 5.12: Max_p for each model and dataset/corruption (Max in bold)

Dataset / Corruption	Logistic Regression	SVM	K-NN	Random Forest	MLP	Lenet-5	ResNet-18
Clean	0.9327	0.9634	0.9510	0.8266	0.9931	0.9908	**0.9949**
Brightness Contrast	**1.0000**	0.3110	0.4000	0.2985	**1.0000**	0.9288	0.8429
Elastic Deformation	0.8825	0.6303	0.8049	0.5034	0.9247	0.9217	**0.9551**
Gaussian Blur	0.9170	0.8976	0.9353	0.4819	0.9880	0.9857	**0.9893**
Gaussian Noise	**0.9834**	0.3110	0.4129	0.3590	0.8504	0.9766	0.8625
JPEG Compression	0.9078	0.4198	0.9057	0.4543	0.8749	0.9901	**0.9938**
Motion Blur	0.9163	0.8818	0.9324	0.5054	0.9867	**0.9868**	0.9811
Poisson Noise	0.9284	0.9623	0.9504	0.8096	0.9924	0.9905	**0.9947**
Random Occlusion	0.9137	**0.9459**	0.9433	0.7702	0.9846	0.9752	0.9802
Speckle Noise	0.9293	0.9628	0.9509	0.8132	0.9928	0.9906	**0.9950**
Salt Pepper Noise	**0.9699**	0.3137	0.5427	0.6777	0.9285	0.9732	0.9191
Mixed	0.9353	0.6631	0.7795	0.5680	0.9520	**0.9715**	0.9510

Table 5.12 reports the **average maximum predicted probability** (*Max p*) across various corruptions for models trained only on clean data. Consistent with our research question on how uncertainty estimation relates to robustness in CNNs and traditional machine learning models, several key observations emerged.

On clean data, all models demonstrate high average Max p values, with ResNet-18

(0.9949) and MLP (0.9931) showing the strongest confidence. This indicates well-calibrated certainty under ideal conditions. Under challenging corruptions such as Brightness Contrast, classical models like SVM (0.3110) and Random Forest (0.2985) exhibit a sharp decline in confidence, reflecting greater uncertainty. In contrast, Logistic Regression and MLP maintain very high confidence (1.0000), which may suggest overconfidence despite the presence of corrupted inputs.

Deep learning models, including MLP, Lenet-5, and ResNet-18, consistently sustain higher average Max p values across corruptions such as Gaussian Blur, JPEG Compression, and Salt Pepper Noise. This highlights their ability to maintain confident predictions even when confronted with unseen distorted data. Conversely, the confidence of classical models degrades significantly with noise and corruption, particularly for SVM and K-NN under Salt Pepper Noise, suggesting that their uncertainty estimates may be more reflective of true prediction difficulty.

These findings address the significance of the research question by illustrating that, while traditional models tend to express reduced confidence on corrupted data, deep learning models often maintain high confidence, which may not always correspond to true robustness.

5.2.4.2 FASHION MNIST Results

Table 5.13: Gini Index Across Models and Corruptions (Min in bold)

Dataset / Corruption	Logistic Regression	SVM	K-NN	Random Forest	MLP	Lenet-5	ResNet-18
Clean	0.1809	0.157	0.1482	0.2844	0.0959	0.1515	**0.0605**
Brightness Contrast	**0.0927**	0.3625	0.7994	0.7611	0.1942	0.3283	0.3089
Elastic Deformation	**0.2076**	0.3936	0.3676	0.5938	0.2253	0.3000	0.2120
Gaussian Blur	0.2124	0.2062	0.2042	0.4704	**0.1631**	0.1669	0.2280
Gaussian Noise	**0.0858**	0.4014	0.5506	0.7559	0.3094	0.1741	0.2433
JPEG Compression	0.1317	0.2619	0.2627	0.5263	0.1746	0.1548	**0.0900**
Motion Blur	0.2179	0.2237	0.2193	0.4810	**0.1697**	0.1737	0.2668
Poisson Noise	0.1824	0.1606	0.1487	0.3122	0.0974	0.1532	**0.0823**
Random Occlusion	0.1879	0.1820	0.1556	0.3794	0.1082	0.1864	**0.0840**
Speckle Noise	0.1822	0.1608	0.1486	0.3145	0.0971	0.1543	**0.0824**
Salt Pepper Noise	**0.1017**	0.4084	0.3288	0.4775	0.1845	0.1864	0.1737
Mixed	**0.1604**	0.2769	0.3189	0.5066	0.1743	0.1985	0.1774

As shown in Table 5.13, which presents Gini Index values across various corruptions for models trained on clean Fashion-MNIST data, ResNet-18 consistently demonstrates the lowest Gini Index values across most corruptions. This includes the clean set (0.0605), Poisson Noise (0.0823), and JPEG Compression (0.0900), indicating strong uncertainty calibration and architectural robustness. Notably, Logistic Regression unexpectedly shows the lowest Gini Index under Gaussian Noise (0.0858) and Salt Pepper Noise (0.1017), suggesting more cautious prediction behavior than typically associated with linear models.

The K-NN algorithm exhibits extremely high uncertainty under Brightness Contrast (0.7994), likely stemming from its sensitivity to pixel-level variations. Random Forest classifiers also demonstrate elevated Gini values across multiple corruptions, indicative of prediction instability. While MLP and Lenet-5 maintain moderate calibration, their performance is consistently surpassed by ResNet-18 in most corruption scenarios.

Table 5.14: Max_p for each model and dataset/corruption (Max in bold)

Dataset / Corruption	Logistic Regression	SVM	K-NN	Random Forest	MLP	Lenet-5	ResNet-18
Clean	0.8719	0.8928	0.8870	0.7951	0.9321	0.8944	**0.9583**
Brightness Contrast	**0.9344**	0.7930	0.2015	0.3337	0.8676	0.7648	0.7743
Elastic Deformation	**0.8509**	0.7256	0.7084	0.5294	0.8355	0.7835	0.8490
Gaussian Blur	0.8481	0.8576	0.8446	0.6434	0.8812	**0.8829**	0.8368
Gaussian Noise	**0.9401**	0.7564	0.5559	0.3745	0.7662	0.8783	0.8285
JPEG Compression	0.9075	0.8243	0.7930	0.6191	0.8680	0.8920	**0.9382**
Motion Blur	0.8441	0.8457	0.8335	0.6316	0.8771	**0.8783**	0.8063
Poisson Noise	0.8705	0.8904	0.8867	0.7744	0.9309	0.8931	**0.9429**
Random Occlusion	0.8665	0.8748	0.8814	0.7206	0.9231	0.8681	**0.9413**
Speckle Noise	0.8711	0.8902	0.8867	0.7723	0.9313	0.8922	**0.9426**
Salt Pepper Noise	**0.9286**	0.7332	0.7455	0.6610	0.8650	0.8690	0.8778
Mixed	**0.8859**	0.8178	0.7337	0.6067	0.8724	0.8594	0.8734

As shown in Table 5.14, ResNet-18 achieves the highest confidence on clean data (0.9583) and several corruptions like Poisson Noise (0.9429), Speckle Noise (0.9426), and Random Occlusion (0.9413), aligning with its low Gini values and suggesting consistent, calibrated confidence.

Interestingly, Logistic Regression shows unusually high confidence under extreme corruptions such as Gaussian Noise (0.9401) and Brightness Contrast (0.9344), despite having higher Gini indices—indicating overconfidence in uncertain conditions. This overconfidence is risky, especially when predictions are wrong but the model remains highly confident .

K-NN, on the other hand, shows low $Max - P$ under Brightness Contrast (0.2015), which matches its high Gini (0.7994), reflecting a correctly cautious attitude in uncertain scenarios.

Overall, ResNet-18 combines high $Max - P$ with low Gini across many corruptions, reflecting both confident and trustworthy predictions, while some traditional models (like Logistic Regression) appear overconfident in noisy conditions.

5.2.5 RQ2 Conclusion

Correlation Study

To investigate how uncertainty estimation relates to model robustness in CNNs and traditional machine learning models under natural image corruptions, we performed a correlation analysis between uncertainty measures and model performance.

Specifically, for each model and dataset, we collected 12 paired observations of the uncertainty metric (either Gini index or Max-P) and the corresponding F1 score. These 12 values correspond to one clean dataset and 11 corrupted versions, capturing a range of input perturbations that impact model robustness.

We employed Spearman's rank correlation coefficient for this analysis due to its ability to measure monotonic relationships without assuming linearity or normality in the data. This is important as the relationship between uncertainty estimation and performance may be non-linear or affected by outliers.

A strong negative Spearman correlation between the Gini index and F1 score indicates that lower uncertainty (lower Gini) corresponds to higher F1 scores, reflecting better robustness. Similarly, a strong positive correlation between Max-P and F1 score indicates that higher maximum softmax probabilities align with better model performance.

This analysis provides insights into how well different uncertainty estimators can serve as proxies for robustness across both CNN and traditional ML models under natural corruptions.

Results of correlation study

The Spearman correlation results presented in Table 5.15 and Table 5.17 reveal a consistent, strong negative relationship between the Gini index and F1 score across multiple models on both MNIST and Fashion-MNIST datasets. Specifically, the Support Vector Machine (SVM) model exhibits the highest negative correlations of -0.992 on MNIST and -0.933 on Fashion-MNIST, demonstrating that a **lower Gini index corresponds to a higher**

F1 score, indicating better model confidence and accuracy under corrupted inputs. Other models, including K-NN, Random Forest, Lenet-5, and ResNet-18, also show strong negative correlations with magnitudes above 0.8, emphasizing the reliability of the Gini index as a proxy for prediction uncertainty and robustness. In contrast, Logistic Regression consistently displays weak and statistically insignificant correlations, highlighting the metric's limited usefulness for simpler linear classifiers.

Similarly, the correlations between maximum softmax probability (Max-P) and F1 score, detailed in Table 5.16 for MNIST and Table 5.18 for Fashion-MNIST, demonstrate a robust positive relationship across most models. SVM again leads with very strong positive correlations (0.993 for MNIST and 0.923 for Fashion-MNIST), indicating that a **higher Max-P aligns with a higher F1 score**, reflecting greater model certainty and better performance under corruptions. K-NN, Lenet-5, ResNet-18, Random Forest, and MLP similarly maintain high positive correlations (above 0.8), confirming Max-P as an effective confidence measure for assessing model performance. Logistic Regression is the only exception, exhibiting moderate negative or weak correlations, further reinforcing its reduced suitability for uncertainty estimation via Max-P in these settings.

Together, these results from both uncertainty metrics across two benchmark datasets underscore their strong and consistent predictive power regarding F1 score variations under corrupted data conditions, especially for complex, non-linear, and deep learning models. They provide practical, unsupervised tools for monitoring model reliability, facilitating real-time detection of performance degradation without requiring labeled data.

Table 5.15: Spearman Correlation between Gini Index and F1 Score for MNIST

Model	**Spearman ρ (Gini vs F1)**	**p-value**
Logistic Regression	0.371	0.236
SVM	-0.993	1.32×10^{-10}
K-NN	-0.928	1.33×10^{-5}
Random Forest	-0.902	6.00×10^{-5}
MLP	-0.490	0.106
Lenet-5	-0.977	4.64×10^{-8}
ResNet-18	-0.984	7.53×10^{-9}

Table 5.16: Spearman Correlation between Max-P and F1 Score for MNIST

Model	**Spearman ρ (Max-P vs F1)**	**p-value**
Logistic Regression	-0.364	0.245
SVM	0.993	1.32×10^{-10}
K-NN	0.928	1.33×10^{-5}
Random Forest	0.909	4.19×10^{-5}
MLP	0.490	0.106
Lenet-5	0.977	4.64×10^{-8}
ResNet-18	0.977	4.64×10^{-8}

Table 5.17: Spearman Correlation between Gini Index and F1 Score for Fashion-MNIST

Model	**Spearman ρ (Gini vs F1)**	**p-value**
Logistic Regression	0.545	0.0666
SVM	-0.944	3.93×10^{-6}
K-NN	-0.993	1.3×10^{-10}
Random Forest	-0.748	0.00512
MLP	-0.951	2.04×10^{-6}
Lenet-5	-0.984	7.46×10^{-9}
ResNet-18	-0.909	4.19×10^{-5}

Table 5.18: Spearman Correlation between Max-P and F1 Score for Fashion MNIST

Model	**Spearman ρ**	**p-value**
Logistic Regression	-0.545	0.0666
SVM	0.923	1.86e-5
K-NN	0.991	3.99e-10
Random Forest	0.804	0.0016
MLP	0.930	1.17e-5
Lenet-5	0.991	3.99e-10
ResNet-18	0.909	4.19e-5

5.3 Research Question 3

What are the trade-offs between computational complexity and robustness across different models?

5.3.1 Research Question Significance

Considering model complexity is crucial in real-world machine learning practice because computational resources can be costly and not all organizations have access to high-performance infrastructure (Dean and Ghemawat (2012)). Selecting an efficient model that matches the specific needs of an application ensures that resources are used wisely, minimizing both financial and operational burdens. For example, training deep learning models for large-scale image processing tasks, such as those used in healthcare diagnostics or autonomous vehicles, can require significant investment in hardware and energy, making it impractical for many settings (Esteva et al. (2017)). Beyond cost, model complexity also affects interpretability, deployment feasibility, and maintenance requirements. Not every use case demands the most complex model; in many scenarios, simpler models can achieve sufficient accuracy while being faster, easier to interpret, and more adaptable to changing conditions (Caruana and Niculescu-Mizil (2006)). Therefore, carefully considering model complexity when choosing an architecture is essential to balance performance, efficiency, and practical deployment constraints in diverse real-world environments.

5.3.2 Methodology for RQ3

To address the question of model complexity in relation to practical deployment, a systematic methodology was employed to ensure a fair and meaningful comparison across all evaluated models. All models were trained using the same hardware environment—a **NVIDIA T4 GPU server provided via Google Colab Pro**—within the same session to eliminate variability arising from hardware differences or fluctuating system loads. This controlled computational setup is crucial for producing reliable and directly comparable results concerning the training efficiency of each model. In addition to measuring training time, the number of trainable parameters was also calculated for every model, offering a quantitative assessment of model size and intrinsic complexity. Parameter count serves as a widely accepted metric to evaluate the resource demands and scalability of machine learning models

in real-world applications (Goodfellow et al. (2016)).

By integrating both training time and parameter count, the analysis captures not only the computational cost associated with model development but also the implications for memory usage and potential inference speed. This dual perspective is particularly relevant when considering deployment scenarios where resources may be limited, such as edge devices or real-time applications. Furthermore, maintaining a consistent experimental setup ensures that the observed differences in complexity are attributable to the models themselves rather than external factors. This methodological rigor supports the broader goal of identifying models that offer an optimal balance between performance, robustness, and practical feasibility, ultimately guiding practitioners in making informed decisions about model selection for diverse real-world contexts.

5.3.3 RQ3 Results

The results presented in Table 5.19 provide several valuable insights into the relationship between model complexity, as measured by training time and number of parameters, and the practical considerations for deploying machine learning models on the MNIST and Fashion-MNIST datasets. First, it is evident that traditional models such as Logistic Regression and Random Forest are extremely lightweight in terms of both parameter count and training time, making them highly accessible for environments with limited computational resources. For instance, Logistic Regression required only 0.04 minutes to train on MNIST and maintained a parameter count of just 7,850, while Random Forest demonstrated similarly low complexity with only 100 parameters and a training time under three minutes. These characteristics make such models attractive for rapid prototyping or deployment on edge devices, although this often comes at the cost of reduced performance and robustness when evaluated under more challenging or corrupted input conditions (see RQ1 5.1.4).

In contrast, deep learning models, particularly ResNet-18, exhibit a dramatic increase in both training time and parameter count, with over 11 million parameters and more than 37 minutes of training required for both datasets. While this increased complexity often

Table 5.19: Training Time and Model Complexity of Each Model on MNIST and Fashion-MNIST

Model	MNIST		Fashion-MNIST	
	Training Time (min)	Parameters	Training Time (min)	Parameters
Logistic Regression	0.04	7,850	3.44	7,850
SVM	39.42	15,549	27.63	21,586
K-NN	≈0	60,000	≈0	60,000
Random Forest	2.44	100	0.28	100
MLP	0.48	535,818	28.21	535,818
Lenet-5	20.19	61,706	21.01	61,706
ResNet-18	37.69	11,196,042	37.69	11,196,042

translates to higher accuracy and improved robustness to natural corruptions, it also imposes significant demands on hardware, energy consumption, and deployment infrastructure. Lenet-5 represents an intermediate point, offering a balance between complexity and performance (see 5.1.4 for the performance results), with a moderate parameter count and training time that are substantially lower than ResNet-18 but higher than traditional models.

The observed differences in training time between datasets for certain models, such as MLP and SVM, also highlight the impact of dataset characteristics on computational efficiency. For example, SVM training is notably slower on MNIST compared to Fashion-MNIST, likely due to differences in data distribution and feature complexity.

Overall, this table was instrumental in answering the research question by quantifying the trade-offs between model complexity and resource requirements. It demonstrates that while deep neural networks can offer superior performance and robustness, their practical deployment may be constrained by resource limitations. Conversely, simpler models, despite their lower computational demands, may not always meet the performance or robustness needs of certain applications. These insights underscore the importance of aligning model choice with the specific computational, operational, and performance requirements of the

intended application, reinforcing that model complexity is a critical factor in real-world machine learning practice.

5.3.4 RQ3 Conclusion

Figures 5.5 and 5.6 present the trade-off between average F1-score drop and training time for each model, evaluated on the MNIST and Fashion-MNIST datasets. The average F1-score drop is calculated by first measuring the difference between the F1-score on the clean dataset and the F1-score on each of the 11 corrupted versions for a given model, and then averaging these differences across all corruptions for that model.

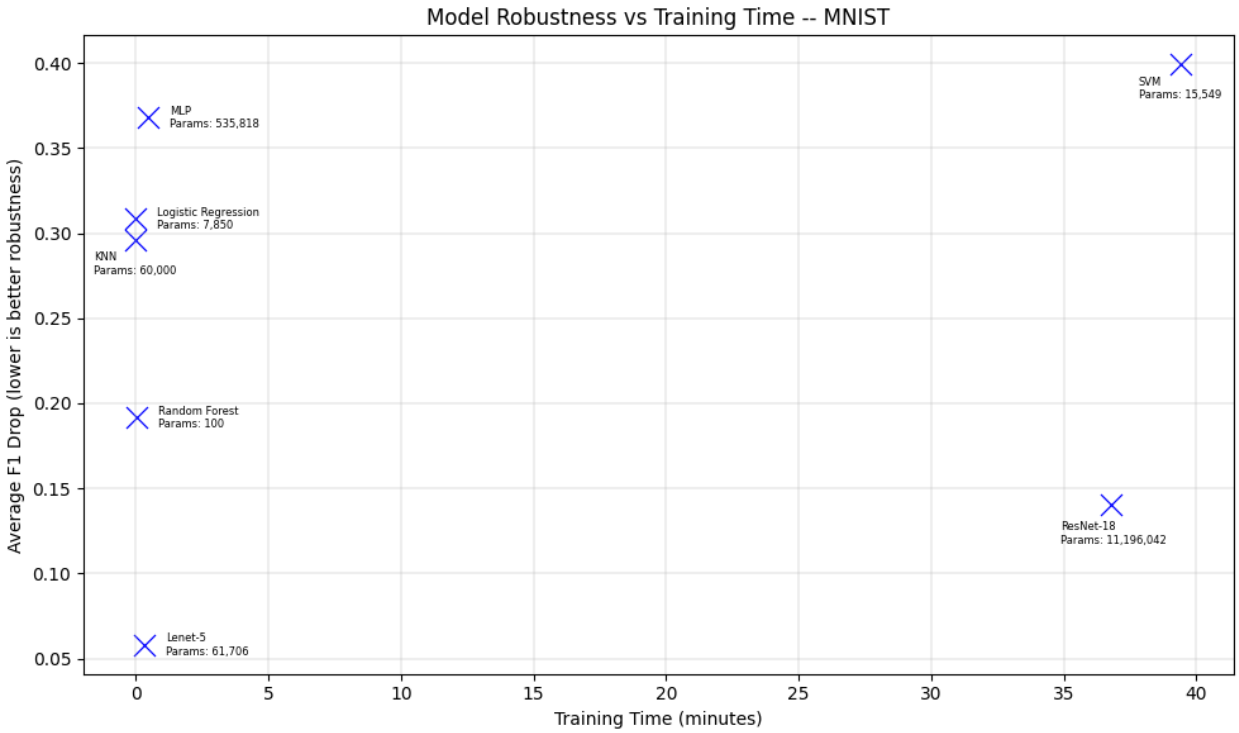


Figure 5.5: F1 score drops versus training time for the models on the MNIST dataset.

In both plots, ResNet-18 appears at the far right due to its high training time, which aligns with its architectural complexity. However, it does not achieve the lowest average F1-score drop. Notably, Lenet-5 consistently appears at the lowest point on both plots, indicating the smallest performance degradation under corruption. With a training time

of approximately three minutes, Lenet-5 offers an effective balance between robustness and computational efficiency.

Figure 5.6: F1 score drops versus training time for the models on the Fashion MNIST dataset.

For MNIST, ResNet-18 ranks second in terms of robustness, which is expected given its deep architecture. In contrast, on the Fashion-MNIST dataset, Random Forest demonstrates the second-best robustness, outperforming other traditional models. Additionally, the MLP performs better than K-NN, Logistic Regression, and SVM on Fashion-MNIST in terms of average F1-score drop. On MNIST, K-NN ranks just below Lenet-5, followed by Logistic Regression and then MLP.

These findings highlight the importance of dataset-specific model selection. While more complex models like ResNet-18 require significantly more resources, their robustness under corruption does not necessarily exceed that of simpler architectures like Lenet-5. Furthermore, strong performance on clean data does not guarantee robustness in real-world scenarios involving noise and corruptions.

The results demonstrate that increased computational complexity does not always lead

to improved robustness. ResNet-18, despite its long training time, does not significantly outperform simpler models in terms of F1-score drop. Lenet-5, a much less complex CNN, achieves the highest robustness with minimal training time. Some traditional models—such as Random Forest and MLP—also show competitive robustness depending on the dataset. Therefore, selecting a model involves weighing the cost of computational resources against the desired level of robustness under corrupted conditions.

6. *Conclusions and Future Work*

This thesis is motivated by the need for reliable machine learning (ML) systems in safety-critical domains such as medical diagnosis, insurance risk assessment, and autonomous driving. In such applications, robustness evaluation during pre-deployment is essential to ensure stable, calibrated, and trustworthy model behavior, helping prevent costly failures. This work addresses this need by introducing a statistically grounded, multi-dimensional robustness evaluation framework that captures not only predictive performance, but also model stability, uncertainty calibration, and computational efficiency. Our framework systematically evaluates seven diverse model families—ranging from classical machine learning algorithms to modern convolutional neural networks (CNNs)—on two widely used image classification benchmarks: MNIST and the more challenging Fashion-MNIST. Each model is tested under 11 corruption types, allowing consistent and controlled comparisons across varying degradation scenarios. By incorporating a diverse set of evaluation metrics—including uncertainty (Gini Index), confidence (Maximum Predicted Probability), prediction stability (Flip Rate, Label Variation), and computational efficiency—we deliver a comprehensive, multi-dimensional analysis of model robustness.

Building on this foundation, the primary contribution of this work is the development of a modular and publicly available robustness evaluation framework that can be easily adapted to any image classification dataset. This framework enables practitioners to simulate real-world corruptions and comprehensively assess models across critical dimensions—performance, uncertainty, stability, and efficiency—before deployment. By facilitating robust model selection and validation in pre-production stages, it supports more informed and reliable decision-

making in high-stakes applications.

In addition to introducing this evaluation framework, we conduct a detailed empirical analysis to uncover deeper insights into model behavior. Through correlation studies and comparative evaluations, we examine the relationships between robustness, uncertainty, and efficiency across model types. These findings offer a more principled understanding of performance trade-offs and provide practical guidance for selecting models suitable for deployment in real-world, safety-critical environments.

MNIST: A Simpler Yet Revealing Benchmark

Across the MNIST dataset, convolutional neural networks (CNNs) consistently outperformed classical models on both clean and corrupted inputs, highlighting their superior capacity for feature extraction and generalization. Among all models, **Lenet-5** emerged as the most consistent overall in terms of average accuracy and F1-score. While **ResNet-18** often led in individual corruptions, **Lenet-5** was more frequently ranked within the top two across all 11 corruption types. Notably, even when not the best performer, **Lenet-5**'s performance was consistently close to the top model, demonstrating minimal performance drop across corruptions such as brightness/contrast, Gaussian noise, motion blur, and mixed noise.

Interestingly, **ResNet-18**, despite its architectural depth, underperformed on certain perturbations. It did not rank first or even second on corruptions like salt-and-pepper noise, and experienced substantial drops in performance on Gaussian noise and brightness contrast. This suggests that deeper CNNs may be more sensitive to high-frequency or intensity-altering corruptions unless explicitly trained with robustness in mind.

Random Forest demonstrated surprising robustness, particularly on salt-and-pepper noise, where it ranked first. This is likely due to its ensemble structure, which confers resilience to localized and sparse noise patterns. It also secured second place on Gaussian noise and brightness contrast, highlighting its adaptability despite being a non-deep model.

MLP (Multi-Layer Perceptron) consistently appeared within the top three, ranking third in five out of the 11 corruptions with competitive performance metrics. Although it is simpler

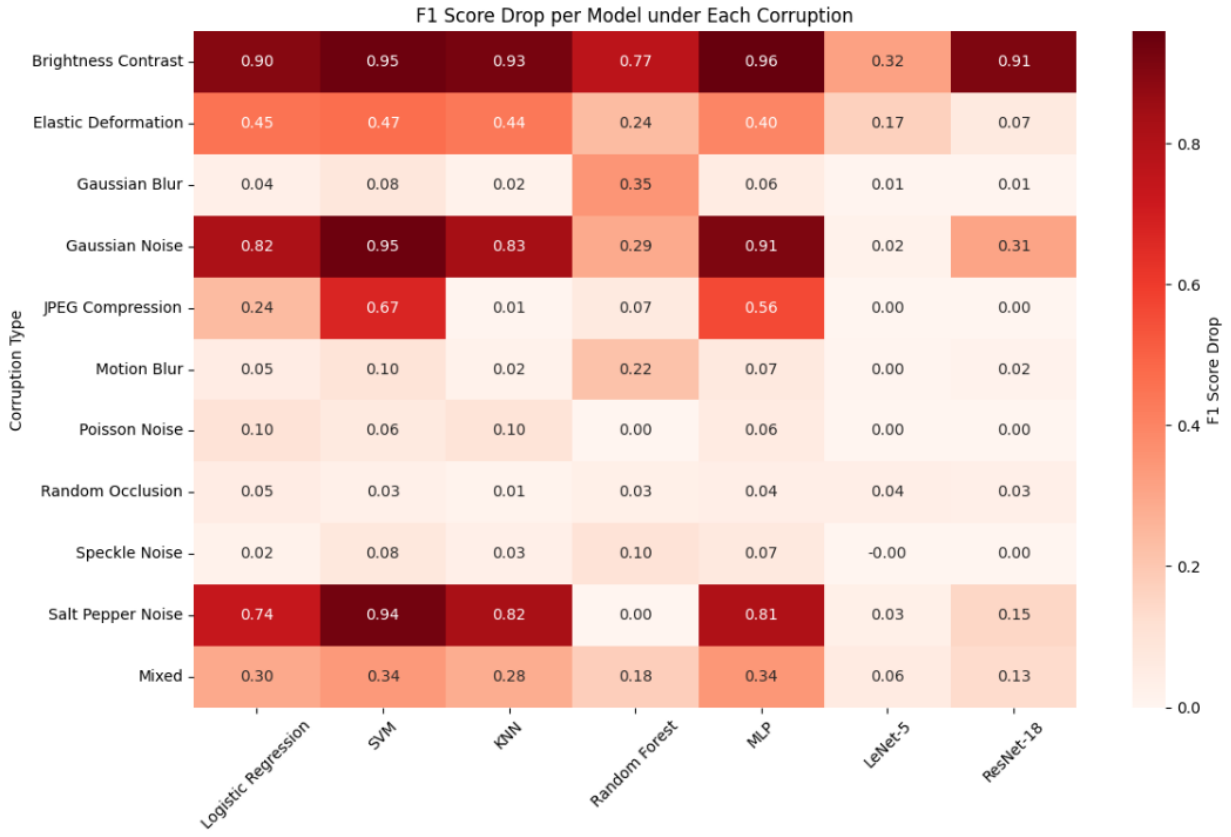


Figure 6.1: MNIST - Heatmap of F1 Score Drop Across Models and Corruptions

and less specialized than CNNs, it performed well likely because its fully connected layers can still capture important patterns—especially when the distortions do not heavily disrupt the overall shape of the digits.

In contrast, **Logistic Regression**, **SVM**, and **K-NN** suffered significant performance degradation under most corruptions. These models showed particularly poor results on Gaussian noise and brightness contrast, indicating their limited reliability and lack of effective feature learning under noisy or distorted conditions.

A visual inspection of the F1 drop heatmap 6.1 further supports these findings. The gradient of colors across rows reveals that no model is immune to corruption-induced degradation.

Model Stability under Corruption: Flip Rate and Label Variation

Lenet-5 led in prediction stability, achieving the lowest Flip Rate and Label Variation across all corruptions. This confirms its robustness not only in accuracy but also in maintaining consistent predictions under degraded inputs.

ResNet-18 and Random Forest followed, with Random Forest showing slightly better label consistency, although ResNet-18 held a lower Flip Rate. Notably, SVM exhibited strong label consistency (second-best Label Variation), despite a high Flip Rate—suggesting its predicted class boundaries remained stable even if individual predictions fluctuated.

Logistic Regression remained the least stable model, with both the highest Label Variation and one of the highest Flip Rates, further highlighting its lack of resilience in corrupted settings.

Fashion-MNIST: A More Challenging Benchmark

Fashion-MNIST introduced greater visual complexity compared to MNIST, making it a more demanding benchmark for evaluating model performance. As with MNIST, convolutional neural networks (CNNs) outperformed classical models on both clean and corrupted inputs. **Lenet-5** and **ResNet-18** each achieved top performance on 5 out of the 11 corruption types. However, when considering average performance across all corruptions—based on both accuracy and F1-score—**Lenet-5** emerged as the most consistently high-performing model overall. It maintained strong results across most distortions, with the only noticeable weakness being on random occlusion, where it still performed competitively.

Following **Lenet-5**, **ResNet-18** demonstrated high performance across many corruptions but showed significant performance drops on Gaussian blur, Gaussian noise, and motion blur. This highlights that **ResNet-18**'s effectiveness can vary depending on the nature of the corruption, making its reliability more context-dependent compared to **Lenet-5**.

Among classical models, **Random Forest** again delivered notable results, leading on salt-and-pepper noise and ranking second on Gaussian noise and elastic deformation. Inter-

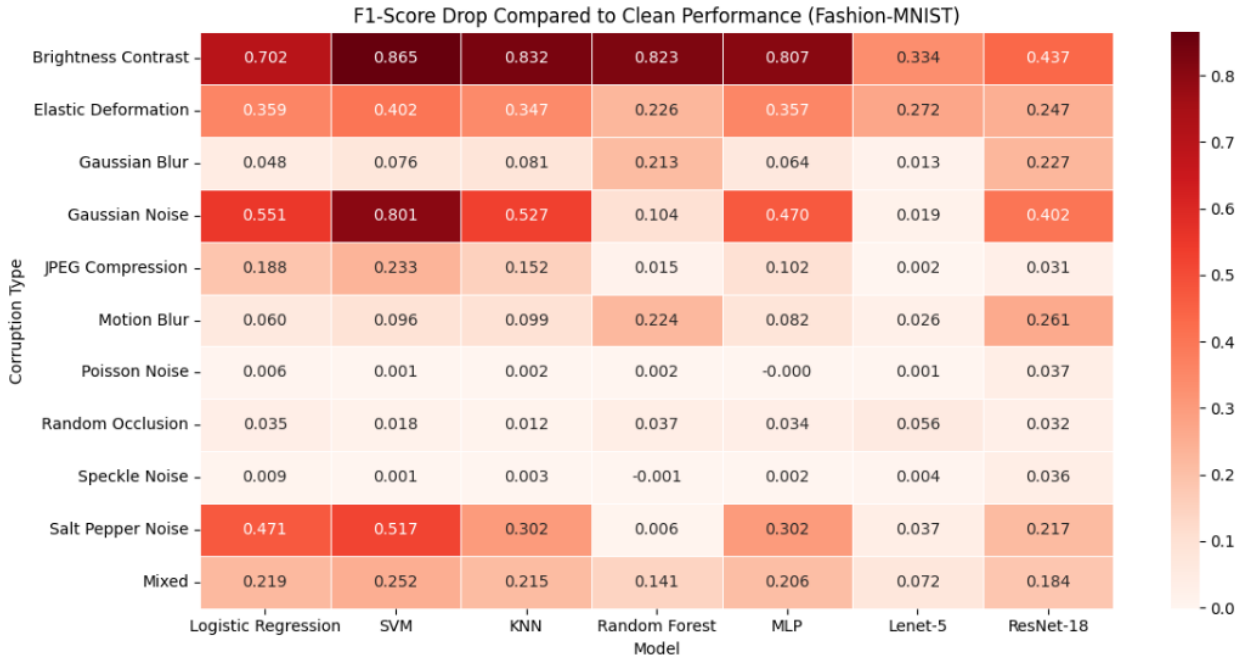


Figure 6.2: Fashion MNIST - Heatmap of F1 Score Drop Across Models and Corruptions

estingly, unlike in MNIST where it performed better on brightness contrast, its strengths in Fashion-MNIST shifted to more spatially complex distortions. This suggests some sensitivity to dataset characteristics.

MLP (Multi-Layer Perceptron) continued to show competitive results, placing in the top three for 5 out of 11 corruption types. Compared to MNIST, **MLP** appeared more frequently in second place rankings than third, indicating slightly stronger relative performance on this more complex dataset.

A surprising shift in Fashion-MNIST was the stronger showing from **SVM**, which appeared in the top three on three challenging corruptions: random occlusion, motion blur, and Gaussian blur. This suggests that under certain structured distortions, margin-based classifiers like **SVM** can still hold their ground.

In contrast, **Logistic Regression** and **K-NN** consistently underperformed, especially on corruptions such as Gaussian noise and brightness contrast, reinforcing their limited ability to generalize under complex perturbations.

A visual inspection of the F1 drop heatmap 6.2 further supports these findings.

Model Stability under Corruption: Flip Rate and Label Variation

The Flip Rate and Label Variation metrics further reveal how stable each model's predictions remain under corruption.

Lenet-5 again stood out as the most stable model, achieving the lowest Flip Rate (0.1917) and the lowest Label Variation (2.66). This indicates strong consistency in its outputs, even as input quality degrades.

Random Forest and ResNet-18 followed in prediction stability, with Random Forest ranking second in Label Variation (3.16), ahead of ResNet-18 (3.22). Interestingly, Random Forest, not ResNet-18, held the second position in output consistency across corruptions. Moreover, resNet-18 showed a significant drop in stability compared to Lenet-5, with noticeably higher Flip Rates and more variable label outputs.

SVM, despite its success in certain corruptions, showed mixed stability: while it had the second-lowest Label Variation (2.78), it ranked last in Flip Rate (0.4054), indicating frequent changes in predicted labels across corrupted inputs.

Logistic Regression remained the least stable model overall, with the highest Label Variation (4.01) and one of the worst Flip Rates (0.3911), underscoring its poor resilience in noisy or distorted environments.

Uncertainty, Confidence, and Robustness

A central contribution of this thesis is the systematic incorporation of uncertainty and confidence measures as complementary indicators of model robustness. By leveraging the Gini Index (as a proxy for predictive uncertainty) and Maximum Predicted Probability (Max-P, as a measure of model confidence), we gain unsupervised, real-time insights into how models behave under unseen input corruptions—critical for safety-critical applications where labels may not be available post-deployment.

On both MNIST and Fashion-MNIST, ResNet-18 and the MLP consistently lead the combined Gini/Max-P rankings, indicating they not only achieve high average confidence

but also maintain calibrated uncertainty distributions across diverse corruptions. LeNet-5 follows closely, reinforcing the value of compact architectures in balancing confidence and uncertainty. Classical methods like Logistic Regression tend to be overconfident, displaying high confidence scores despite poor generalization performance across corruptions. K-NN exhibits a similar trend, though to a lesser extent, suggesting moderate misalignment between its confidence and actual robustness.

Intriguingly, the correlation patterns between these metrics and downstream F1 performance shift markedly between datasets. On MNIST, deep models (ResNet-18, LeNet-5) exhibit the strongest positive correlation between low uncertainty (high Max-P) and high F1, whereas on Fashion-MNIST, simpler models like K-NN and MLP demonstrate unexpectedly strong correlations. This dataset-dependent reordering underscores that uncertainty and confidence signals are not universally transferable—they must be interpreted in the context of data complexity and feature distributions.

Taken together, these findings highlight two key takeaways: 1. **Real-Time Monitoring without Labels:** Gini Index and Max-P can function as unsupervised robustness monitors, flagging instances of distributional shift or extreme input degradation purely from model outputs, which is invaluable for live systems lacking immediate ground truth. 2. **Dataset Sensitivity of Uncertainty Signals:** The utility and interpretability of uncertainty and confidence diagnostics are inherently tied to dataset characteristics; practitioners should calibrate and validate these metrics on representative benchmarks before trusting them in production.

By formalizing and empirically validating these uncertainty–confidence robustness indicators, this work provides a practical toolkit for anticipating and mitigating failure modes in deployed ML systems, thereby advancing both the theory and practice of robust model evaluation.

Computational Efficiency and Trade-offs

While ResNet-18 delivered top-tier performance, it came with the highest computational cost. Surprisingly, **Lenet-5** achieved the lowest average F1-score drop while maintaining

minimal training time. This trade-off between robustness and complexity suggests that model depth is not synonymous with reliability.

Traditional models like **Random Forest** and **MLP** performed well on Fashion-MNIST despite their lower complexity, reinforcing the idea that robustness must be evaluated in conjunction with dataset characteristics and real-world constraints.

Future Work

As machine learning systems are increasingly deployed in real-world environments, future work must extend robustness evaluation beyond conventional models and datasets. This includes investigating performance on larger, more diverse, and less curated datasets that better reflect deployment conditions. Additionally, there is growing interest in optimizing both robustness and computational efficiency simultaneously, especially for applications in resource-constrained environments.

A particularly promising direction involves evaluating the robustness of **vision-language models (VLMs)** and **agentic multimodal systems**. Recent models such as CLIP, BLIP, Flamingo, and GPT-4o exemplify a shift toward agents that can perceive, reason, and act across multiple modalities. These systems are already being adopted across industries. In **healthcare**, they are used to interpret medical images and generate diagnostic reports (Tiu et al. (2022), Wang et al. (2023)). In **autonomous vehicles**, vision-language models support perception, planning, and language-guided navigation (Kim et al. (2023), Shen et al. (2023)). **Assistive technologies** benefit from real-time scene understanding and multimodal interaction for users with visual or cognitive impairments (Alayrac et al. (2022), Yang et al. (2022)). In **e-commerce and robotics**, these systems enable visual search, product recommendation, and task planning from natural language instructions (Li et al. (2022), Radford et al. (2021)). Such applications demonstrate the versatility of multimodal agents, but also highlight the need to rigorously evaluate their robustness in dynamic, unstructured environments.

While VLMs offer impressive zero-shot and generalization capabilities, their behavior under *input corruptions*, *distribution shifts*, and *uncertainty* remains underexplored. Robustness in this context is critical not just for performance, but for safety, trustworthiness, and usability in real-world scenarios. Therefore, a key area of future research is to benchmark the robustness of multimodal models against traditional CNNs and classical ML models across corruption benchmarks and uncertainty metrics.

This research direction will help uncover new trade-offs, inform deployment decisions, and contribute to the development of **resilient, reliable, and generalizable AI systems**.

Bibliography

J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Laptev, J. Sivic, and A. Zisserman. Flamingo: a visual language model for few-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 23716–23736, 2022.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168. ACM, 2006.

R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, and N. Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1721–1730, 2015.

D. Ciregan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3642–3649. IEEE, 2012.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 113–123, 2019.

J. Dean and S. Ghemawat. Large scale distributed deep networks. *Advances in Neural Information Processing Systems*, 25:1223–1231, 2012.

A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542 (7639):115–118, 2017.

S. G. Finlayson, J. D. Bowers, J. Ito, J. L. Zittrain, A. L. Beam, and I. S. Kohane. Adversarial attacks on medical machine learning. *Science*, 363(6433):1287–1289, 2019. doi: 10.1126/science.aaw4399.

Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning (ICML)*, pages 1050–1059. PMLR, 2016.

C. F. Gauss. *Theoria motus corporum coelestium in sectionibus conicis solem ambientium.* 1809.

J. Gilmer, N. Ford, N. Carlini, E. D. Cubuk, and I. Goodfellow. Adversarial examples are not bugs, they are features. *arXiv preprint arXiv:1905.02175*, 2019.

C. Gini. *Variabilità e mutabilità: contributo allo studio delle distribuzioni e delle relazioni statistiche.* Tipografia di P. Cuppini, Bologna, Italy, 1912. In Italian.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning.* MIT Press, 2016.

J. W. Goodman. Statistical properties of laser speckle patterns. In *Laser Speckle and Related Phenomena*. Springer, 1975.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

D. Hendrycks and T. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations (ICLR)*, 2019.

D. Hendrycks and K. Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations (ICLR)*, 2017.

D. Hendrycks, K. Lee, M. Mazeika, and D. Song. Pretrained transformers improve out-of-distribution robustness. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020a.

D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. In *International Conference on Learning Representations (ICLR)*, 2020b.

D. Hendrycks, K. Zhao, S. Basart, J. Steinhardt, and D. Song. Natural adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15262–15271, 2021. doi: 10.1109/CVPR46437.2021.01500.

D. W. Hosmer, S. Lemeshow, and R. X. Sturdivant. *Applied Logistic Regression*. Wiley, 3 edition, 2013. ISBN 978-0-470-58247-3.

M. Jaderberg, K. Simonyan, and A. Zisserman. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015.

H. Kim, S. Gupta, J. Lee, Y. Kwon, J. Choi, J. Oh, K. Yoon, K. Yoon, and Y. Joo. Language-conditioned imitation learning for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 23504–23513, 2023.

A. Krizhevsky. Learning multiple layers of features from tiny images. Technical Report Tech. Rep., University of Toronto, 2009.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25, pages 1097–1105, 2012.

B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Y. LeCun, C. Cortes, and C. Burges. Mnist handwritten digit database. AT&T Labs [Online], 2010a. URL `http://yann.lecun.com/exdb/mnist`.

Y. LeCun, C. Cortes, and C. J. Burges. The mnist database of handwritten digits. 2010b. http://yann.lecun.com/exdb/mnist/.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. doi: 10.1038/nature14539.

J. Li, D. Su, C. Shen, W. Zhang, Y. Pang, J. Cao, L. Cheng, W. Xiong, J. Feng, S. Yan, et al. Blip: Bootstrapping language-image pre-training for unified vision-language un-

derstanding and generation. In *International Conference on Machine Learning (ICML)*, pages 12888–12900. PMLR, 2022.

T. Lindeberg. Scale-space for discrete signals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(3):234–254, 1990.

A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*, 2018.

A. Malinin and M. Gales. Uncertainty estimation and knowledge distillation with confidence-aware training. In *International Conference on Machine Learning (ICML)*, pages 6704–6714. PMLR, 2020.

C. Michaelis, M. Möller, and T. Ropinski. Benchmarking robustness in object detection: Autonomous driving when winter is coming. In *Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 0–0, 2019. doi: 10.1109/ICCVW.2019.00009.

S.-M. Moosavi-Dezfooli, O. Fawzi, A. Fawzi, and P. Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 86–95, 2017.

N. Mu, J. Gilmer, K. R. Mopuri, S. S. Schoenholz, J. Sohl-Dickstein, E. D. Cubuk, and B. Zoph. Mnist-c: A robustness benchmark for computer vision. *arXiv preprint arXiv:1906.02337*, 2019.

K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, Cambridge, MA, 2012.

Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek. Can you trust your model's uncertainty? evaluating predictive

uncertainty under dataset shift. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

E. Peli. Contrast in complex images. *Journal of the Optical Society of America*, 7(10): 2032–2040, 1990.

D. Pham et al. Solving label variation in scientific information extraction via multi-task learning. *arXiv preprint arXiv:2312.15751*, 2023. URL `https://arxiv.org/abs/2312.15751`.

S.-D. Poisson. *Recherches sur la probabilité des jugements en matière criminelle et en matière civile*. Bachelier, Paris, 1837.

A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

H. R. Sayed and G. J. Brostow. Blur robust optical flow using motion channel. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16406–16415. IEEE, 2021.

W. Schottky. Über spontane stromschwankungen in verschiedenen elektrizitätsleitern. *Annalen der Physik*, 1918.

scikit-learn developers. scikit-learn: Machine learning in python. `https://github.com/scikit-learn/scikit-learn`, 2024. Accessed: 2025-05-18.

L. G. Shapiro and G. C. Stockman. *Computer Vision*. Prentice Hall, Upper Saddle River, NJ, 2001.

Y. Shen, Y. Wang, Y. Chen, Z. Li, Q. Lu, L. Wang, and Z. Huang. Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface. *arXiv preprint arXiv:2303.17580*, 2023.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Pearson Addison Wesley, 2005.

TensorFlow Authors. Tensorflow: An open source machine learning framework for everyone. `https://github.com/tensorflow/tensorflow`, 2024. Accessed: 2025-05-18.

E. Tiu, R. Arnaout, and R. Arnaout. Expert-level detection of pathologies from unannotated chest x-ray images via self-supervised learning. *Nature Biomedical Engineering*, 6:148–158, 2022.

D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry. Robustness may be at odds with accuracy. In *International Conference on Learning Representations (ICLR)*, 2018.

D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry. Robustness may be at odds with accuracy. *International Conference on Learning Representations (ICLR)*, 2019.

G. K. Wallace. The jpeg still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):18–34, February 1992. doi: 10.1109/30.125072.

L. Wan, M. D. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus. Regularization of neural networks using dropconnect. In *International conference on machine learning*, pages 1058–1066. PMLR, 2013.

Q. Wang, S. Gu, Q. Guo, B. Lei, W. Wang, and D. Shen. Chatcad: Interactive computer-aided diagnosis on medical image using large language models. *arXiv preprint arXiv:2301.07652*, 2023.

H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

C. Xie, Y. Wu, L. van der Maaten, A. Yuille, and K. He. Feature denoising for improving adversarial robustness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 501–509, 2019.

J. Yang, B. Tan, C. Zhang, and C. Lu. Empowering visually impaired people with visual question answering: Challenges and opportunities. *IEEE Signal Processing Magazine*, 39 (5):93–103, 2022.

J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3320–3328, 2014.

R. Yousefzadeh and D. P. O'Leary. Deep learning interpretation: Flip points and homotopy methods. In *Proceedings of The First Mathematical and Scientific Machine Learning Conference*, volume 107 of *Proceedings of Machine Learning Research*, pages 1–26, 2020. URL `http://proceedings.mlr.press/v107/yousefzadeh20a.html`.

C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations (ICLR)*, 2017.

H. Zhang and H. Zha. Knn model-based approach in classification. *International Journal of Computer and Information Engineering*, 11(1):22–26, 2017.

H. Zhang, Y. Yu, V. Jojic, B. Xiao, Y. Wang, and M. I. Jordan. Interpreting adversarial robustness: A case study on robust and non-robust models. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

H. Zhang, C. Wu, Z. Zhang, Y. Zhu, H. Lin, and J. Sun. Resnest: Split-attention networks. *arXiv preprint arXiv:2004.08955*, 2020.