Spike Time Sensitivity in Spiking Neural Networks

Investigating the Effect of Sample Difficulty in Time-to-First-Spike Coded Spiking Neural Networks

Eren Aydoslu

Delft University of Technology

Spike Time Sensitivity in Spiking Neural Networks

Investigating the Effect of Sample Difficulty in Time-to-First-Spike Coded Spiking Neural Networks

by

Eren Aydoslu

Student Name	Student Number
Eren Aydoslu	4997778

Faculty: Electrical Engineering, Mathematics and Computer Science, Delft

Cover: Cover art generated by Dream AI
Style: TU Delft Report Style, with modifications by Daan Zwaneveld

Preface

This thesis grew out of my genuine interest in the overlap between neuroscience and machine learning. I've always been fascinated by how the brain processes information, and Spiking Neural Networks offered an ideal framework for pursuing this curiosity. Throughout my research, I discovered unexpected connections with stochastic processes and cognitive neuroscience, which enriched the project as I've always liked working across disciplines. This thesis represents the results of that spirit: it synthesizes ideas from deep learning, stochastic dynamics, and human cognition to shed new light on how sample difficulty affects spike-time sensitivity in SNNs.

I'd like to thank my supervisors, Dr. J.C. van Gemert, Dr. N. Tömen, Dr. O. Booij, and A. Micheli, for their support and guidance. This project deepened my understanding of deep learning more than I expected and boosted my appreciation for all the researchers who dedicate their careers to tackling these complex topics.

And finally, I would like to thank my family and my friends for always being there and for their continuous support throughout the whole process.

Eren Aydoslu
Delft, June 2025

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
ANN(s)	Artificial Neural Network(s)
CNN(s)	Convolutional Neural Network(s)
DDM	Drift Diffusion Model
GRW	Gaussian Random Walk
MLP	Multi Layer Perceptron
NN(s)	Neural Network(s)
SNN(s)	Spiking Neural Network(s)
RW	Random Walk
TTFS	Time to First Spike

1 Introduction

Artificial intelligence has gone through incredible jumps in performance in recent years, pushing the boundaries of what's possible in tasks like image recognition, natural language processing, and even creative generation. Despite these advancements, the human brain remains an unmatched marvel in its efficiency, speed, and accuracy, especially in sensory perception and decision-making tasks [23, 44, 75]. This biological benchmark continues to inspire alternative models of computation that seek to emulate the brain's mechanisms more closely than traditional artificial neural networks (ANNs).

Among the most promising of these bio-inspired approaches are *Spiking Neural Networks* (SNNs), which introduce temporal dynamics into neural computation [47]. Unlike ANNs that process data in continuous activations, SNNs transmit information through discrete events known as *spikes*. This inherently temporal encoding offers a closer approximation to neural behavior observed in the brain and opens up the potential for highly efficient, low-power computation, especially when paired with event-based neuromorphic hardware [19, 35, 48].

Within the family of SNNs, one particularly compelling coding scheme is *Time-to-First-Spike* (TTFS). In this encoding, information is conveyed by the latency of the first spike relative to stimulus onset. TTFS coding has shown promise in accelerating inference and improving energy efficiency [15, 50]. However, one underexplored aspect of TTFS SNNs is how the difficulty of input samples affects their spiking behavior. In biological systems, easier stimuli often result in faster reactions, a trait that enhances survival and decision-making efficiency [23]. Whether TTFS SNNs exhibit similar sensitivity to input difficulty remains an open question.

This thesis aims to investigate this relationship in depth. Specifically, we explore how varying sample difficulty influences inference latency in TTFS-coded SNNs. We hypothesize that easier samples will trigger earlier spikes, mirroring human reaction patterns [60, 61]. Furthermore, we examine whether introducing noise during training, analogous to ℓ_2 regularization, affects this dynamic by smoothing learned representations and potentially altering time-to-spike distributions.

To guide this investigation, our primary research questions are:

- Does spike latency increase with intrinsic sample difficulty in TTFS-SNNs?
- Can noise training (without augmentation) during training reduce inference latency for difficult samples?

The rest of this thesis is structured in two parts. The first is a comprehensive background review that lays the groundwork for understanding the key concepts explored, including SNN architectures, TTFS coding, neural noise, and biological analogues of decision-making latency. The second part is an academic paper that encapsulates the core experiments, results, and contributions of this work.

2
Artificial Neural Networks

Before delving into Spiking Neural Networks (SNNs), it is essential to establish a clear understanding of their non-spiking counterparts - Artificial Neural Networks (ANNs). This chapter, therefore, reviews the core principles that underpin modern ANNs, focusing on concepts that will later be revisited in the spiking domain. We concentrate on fully connected and convolutional architectures, as these are directly transformed to SNNs in the remainder of this thesis.

2.1. Perceptron and Deep Learning

2.1.1. The Perceptron

The simplest neuron model is the *perceptron*, proposed by Rosenblatt in 1958 [65]. Given an input vector $\mathbf{x} \in \mathbb{R}^d$, weight vector $\mathbf{w} \in \mathbb{R}^d$, bias $b \in \mathbb{R}$, and non-linearity $\sigma(\cdot)$ (e.g. $\tanh$ or ReLU), the unit outputs

$$y = \sigma(w^\top \mathbf{x} + b). \tag{2.1}$$

Equation 2.1 is the fundamental building block of both MLPs and the fully connected layers frequently used at the tail of CNNs.

2.1.2. Universal Approximation and Depth

A single perceptron is a linear classifier; stacking layers of perceptrons with non-linear activations yields the MLP (section 2.2) and opens the door for the universal approximation property [34]. Depth allows hierarchical feature composition, but also introduces a non-convex optimization landscape that must be navigated during training.

2.1.3. Back-Propagation and Gradient Descent

Modern artificial neural networks (ANNs) are trained by minimizing a differentiable loss function $\mathcal{L}(\theta)$, where θ represents the set of all trainable parameters. For classification tasks, a widely adopted loss function is the cross-entropy loss:

$$\mathcal{L}_{\mathrm{CE}} = -\sum t_c \log p_c, \tag{2.2}$$

with t_c denoting the one-hot encoded target vector and p_c representing the predicted probability obtained via a soft-max layer.

Efficient gradient computation with respect to parameters is achieved through the back-propagation algorithm, a cornerstone of ANN training. Back-propagation employs the chain rule from calculus to systematically propagate error signals backwards through the network layers, starting from the output and moving towards the input layers [68]. Specifically, after the forward pass computes predictions, the backward pass sequentially computes the partial derivatives of the loss function with respect to each parameter by recursively applying the chain rule. Formally, the parameter gradients are obtained as:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \text{backprop}(\mathcal{L}, \theta). \tag{2.3}$$

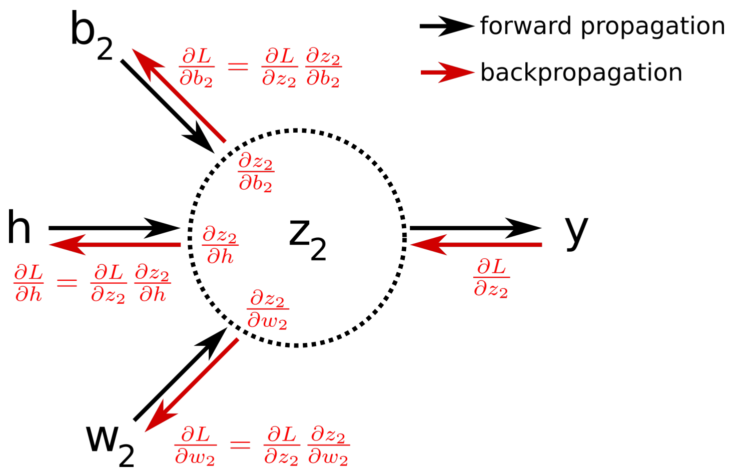

forward propagation
backpropagation
b2
$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial z_2}\frac{\partial z_2}{\partial b_2}$
$\frac{\partial z_2}{\partial b_2}$
h
$\frac{\partial z_2}{\partial h}$
z2
y
$\frac{\partial L}{\partial z_2}$
$\frac{\partial L}{\partial h} = \frac{\partial L}{\partial z_2}\frac{\partial z_2}{\partial h}$
$\frac{\partial z_2}{\partial w_2}$
w2
$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial z_2}\frac{\partial z_2}{\partial w_2}$

Figure 2.1: Illustration of forward and backward propagation for a single neuron. Black arrows indicate the forward pass, where input h and parameters w_2, b_2 are combined. Red arrows indicate the backward pass, where the gradient of the loss $\partial L/\partial z_2$ is propagated back to compute the parameter gradients $\partial L/\partial w_2$ and $\partial L/\partial b_2$, as well as the input gradient $\partial L/\partial h$.

Once gradients are computed, parameters are iteratively updated using (stochastic) gradient descent. Gradient descent is an optimization algorithm foundational to training neural networks. Its primary goal is to minimize the loss function by iteratively adjusting the model parameters in the direction of the steepest descent. The intuition behind gradient descent is analogous to descending a mountain: at each step, the algorithm assesses the slope of the terrain (computed as the gradient of the loss function) and moves downhill, aiming to reach the lowest possible point, or the global minimum, of the loss landscape.

In mathematical terms, gradient descent updates the model parameters as follows:

$$\theta \leftarrow \theta - \eta \frac{\partial \mathcal{L}}{\partial \theta} \tag{2.4}$$

where η is the learning rate, a hyperparameter controlling the step size of each update. A small learning rate ensures stable convergence but may result in slow training, while a large learning rate speeds up training but risks overshooting the minimum. Adaptive variants such as Adam and RMSProp address this challenge by dynamically adjusting step sizes during training, significantly enhancing the robustness and efficiency of the optimization process [40].

2.2. Multi-Layer Perceptrons

An MLP comprises an input layer, $L-2$ hidden layers, and an output layer. Each hidden layer performs a linear transformation followed by an activation:

$$\mathbf{h}^{(\ell)} = \sigma\left(\mathbf{W}^{(\ell)}\mathbf{h}^{(\ell-1)} + \mathbf{b}^{(\ell)}\right), \qquad \ell = 1, \ldots, L-2, \tag{2.5}$$

where $x \in \mathbb{R}^d$ is the input, $\mathbf{W}(\ell) \in \mathbb{R}^{m_\ell \times m_{\ell-1}}$ and $\mathbf{b}(\ell) \in \mathbb{R}^{m_\ell}$ are the layer's weights and biases, and σ is a pointwise nonlinearity (e.g., ReLU or tanh). The final layer's activations $\mathbf{h}^{(L-1)}$ are interpreted as class logits for classification tasks.

Although conceptually simple, MLPs can approximate any continuous function on a compact domain when sufficiently wide or deep, known as the universal approximation theorem [34]. However, they ignore spatial structure (e.g. in images) and suffer $\mathcal{O}(d^2)$ parameter growth, motivating convolutional designs for high-dimensional inputs.

2.3. Convolutional Neural Networks

In a convolutional layer, the kernel (often referred to as a filter) serves as a localized feature detector that systematically surveys the input tensor [43]. Each kernel comprises a small set of learnable weights that, once

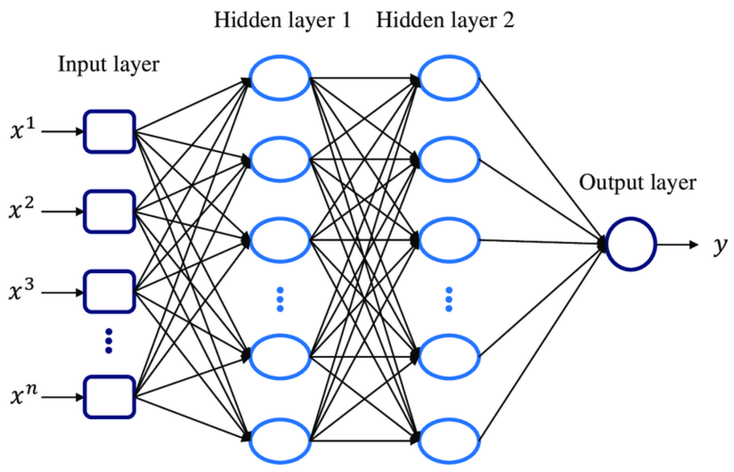


Figure 2.2: MLP with two hidden layers: each neuron in a given layer connects to every neuron in the preceding and subsequent layers.

trained, respond maximally to a specific elementary pattern, such as an edge at a particular orientation or a localized texture [91]. As the kernel is convolved across the entire spatial extent of the input, it generates a feature map whose entries quantify the presence of that pattern at each location. This mechanism confers two important properties: first, the model learns to recognize features irrespective of their position, since the same kernel is reused everywhere [90]; second, by employing multiple distinct kernels in parallel and by stacking convolutional layers, one obtains a hierarchy of representations that progresses from simple, low-level cues in the earliest layers to increasingly abstract, high level constructs in deeper layers.

Mathematically, the learnable kernel is defined as $\mathbf{K} \in \mathbb{R}^{k \times k \times C_{\text{in}} \times C_{\text{out}}}$, and is slid across the input feature map to produce an output feature map:

$$Y_{i,j,c} = \sum_{u=1}^{k} \sum_{v=1}^{k} \sum_{c'=1}^{C_{in}} K_{u,v,c',c} \, X_{i+u,j+v,c'} \tag{2.6}$$

Since the same kernel slides over every spatial location, the layer is intrinsically translation-equivariant [43]. By stacking multiple convolutions (often interleaved with non-linearities and pooling), the network's receptive field grows, enabling it to capture increasingly large (see Figure 2.3) and abstract features [46, 92].

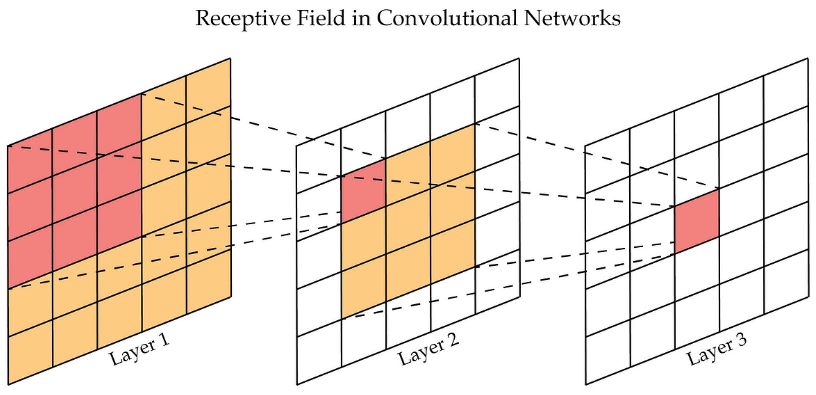


Figure 2.3: The receptive field of a 3×3 convolution increases with depth: each successive layer "sees" a larger region of the original input.

While perceptrons constitute simple nonlinear units (see Equation 2.1), stacking them produces powerful MLPs that capture global data interactions, albeit at the cost of ignoring spatial structure and scaling poorly in high dimensions. In contrast, convolutional kernels detect local patterns by sliding a small weight matrix across inputs, yielding translation invariance through weight sharing. As layers are stacked, their receptive fields expand recursively, enabling hierarchical feature extraction from raw data [27, 45, 70, 73]. These layer types, fully connected and convolutional, underlie virtually all modern deep vision architectures and will be translated into spiking equivalents in the forthcoming chapters.

3 Spiking Neural Networks

Spiking Neural Networks (SNNs) are the third generation of neural network models, in which information is communicated via discrete spikes rather than continuous activations. This temporal coding of information enables SNNs to exploit the time domain, offering potential gains in computational efficiency and biological plausibility [47]. In this chapter, we review the foundations of SNNs, including the neuromorphic computing, neuron and encoding models, and training methods.

3.1. Leaky Integrate-and-Fire Neuron Model

Biological neurons are extraordinarily complex structures, they typically have thousands of dendritic branches, complicated ion-channel dynamics, nonlinear membrane properties, and diverse forms of synaptic plasticity, making detailed biophysical models like Hodgkin-Huxley computationally intensive and mathematically intricate [32]. While such realism is valuable for neuroscience, it poses severe challenges for large-scale simulations or for practical applications requiring real-time or energy-efficient computation. It is therefore common in neuromorphic modelling and spiking neural networks to replace detailed neuron descriptions with much simpler abstractions that capture the essence of neuronal spiking behavior.

The Leaky Integrate-and-Fire (LIF) model exemplifies this approach, distilling core neuron dynamics into a basic temporal integrate-and-decay process with a threshold and reset [9]. Despite its simplicity, the LIF neuron retains essential temporal filtering and thresholding behavior, allowing it to emulate key computational features of more complex neurons while remaining efficient enough for use in large networks or hardware implementations [87]. The membrane potential update equation for LIF in discrete time is given by:

$$V_i[t+1] = \beta\, V_i[t] + \sum_j w_{ij} S_j[t] - V_{th}\, S_i[t] \tag{3.1}$$

where,

- $V_i[t]$ is the membrane potential of neuron i at timestep t,
- $\beta \in (0, 1)$ is the decay factor,
- w_{ij} denotes the synaptic weight from presynaptic neuron j to neuron i,
- $S_j[t] \in \{0, 1\}$ indicates whether neuron j emitted a spike at t,
- V_{th} is the firing threshold, and
- if $V_i[t] \geq V_{th}$ then $S_i[t] = 1$ and $V_i[t]$ is reset by substracting V_{th}.

3.2. Spike Encoding

Biological neurons communicate via discrete action potentials (spikes), whereas most sensory data (e.g., images, audio, sensor readings) are represented as continuous-valued signals. To leverage the temporal dynamics and event-driven efficiency of SNNs, continuous inputs must be transformed into spike trains that the network can process [47].

3.2.1. Rate Encoding

In rate encoding, the intensity of an input feature is mapped to the average firing rate of a spike train over a given time window. For example, one may generate a Poisson spike train whose rate parameter is proportional to the pixel intensity. While rate codes are robust to temporal noise and simple to implement in hardware, they require many spikes (and thus many timesteps) to convey information accurately, limiting latency and energy efficiency [74, 81].

3.2.2. Latency Encoding

Latency (or time-to-first-spike) encoding converts input magnitude into the timing of a single spike: larger values spike earlier, and smaller values spike later. Formally, for an input $I \in [0, 1]$, the spike time is given by

$$t_{\text{spike}} = \begin{cases} \text{round}\left((1 - I) \cdot T_{\text{max}}\right), & I > \epsilon, \\ \text{no spike}, & I \leq \epsilon, \end{cases} \tag{3.2}$$

where T_{max} is the simulation duration and ϵ a threshold below which inputs are ignored [15].

Because of its high efficiency, requiring only one spike per input channel, TTFS encoding is particularly attractive for low-power neuromorphic hardware. In this thesis, we focus on latency encoding to explore the time-to-first-spike dynamics of SNNs.

3.2.3. Biological Plausibility

Rate codes align with slow, averaged neural responses in some sensory systems but fail to capture rapid processing observed in vision and audition [74]. In contrast, latency codes (also called rank-order or first-spike codes) closely mirror observations in the retina, auditory pathways, and tactile systems, where the timing of the first spike carries the majority of the informational content [81].

3.3. Training Spiking Neural Networks

Training SNNs is challenging due to the non-differentiable nature of spikes. Although there are many ways to overcome this problem, we focus on two key techniques that address this issue: surrogate gradient methods and Backpropagation Through Time (BPTT).

3.3.1. Surrogate Gradients

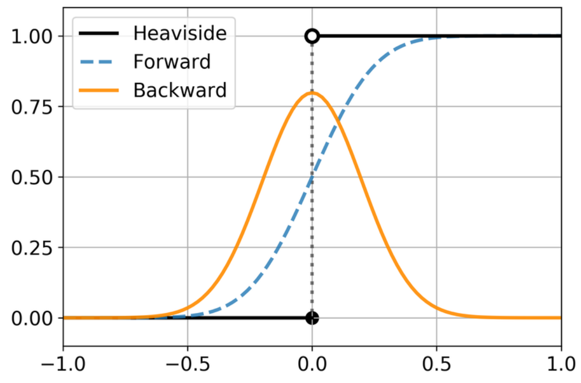


Figure 3.1: Comparison of the ideal Heaviside step activation (solid black) with its differentiable surrogate gradients: a sigmoid-based forward surrogate in dashed blue and the backward surrogate gradient in solid orange.

Spiking generation is modeled by the Heaviside step function in the forward pass:

$$S = \begin{cases} 1, & V \geq V_{th}, \\ 0, & V < V_{th}, \end{cases} \tag{3.3}$$

whose exact derivative is zero almost everywhere and undefined when $V = V_{th}$. Surrogate gradients replace this with a smooth approximation during backpropagation (see Figure 3.1). A common choice is the shifted arc-tangent:

$$S \approx \frac{1}{\pi} \arctan\left(\frac{\pi}{2}\alpha V\right) \tag{3.4}$$

$$\frac{\partial S}{\partial V} = \frac{1}{\pi} \frac{1}{1 + \left(\frac{\pi}{2}\alpha V\right)^2} \tag{3.5}$$

where α controls smoothness [15, 55]. This enables gradient-based optimization despite the inherent discontinuity.

3.3.2. Backpropagation Through Time

Since SNNs evolve over multiple timesteps, training requires propagating gradients through both spatial connections and temporal dynamics. BPTT unfolds the network in time, treating each timestep as a layer in a deep computational graph. Gradients are then computed across this graph, enabling end-to-end learning of synaptic weights while preserving temporal dependencies [85].

3.4. Loss Function

For latency-encoded output layers, one can define a temporal MSE loss over normalized spike times [15]. Given normalized spike times $t_i \in [0, 1]$ and target times y_i (0 for correct class, 1 for incorrect), the loss is:

$$L = \frac{1}{N} \sum_{i=1}^{N} (t_i - y_i)^2. \tag{3.6}$$

However, the mapping from the continuous membrane potential V to the discrete spike time index t_{spike} (see Equation 3.2) is inherently non-differentiable with respect to V, since t_{spike} is defined as the first timestep at which V crosses threshold, resulting in a piecewise constant (indexed) function of V. As a result, the formal derivative $\partial t_{spike}/\partial V$ is undefined almost everywhere. To permit gradient-based learning despite this, it is common to impose the heuristic

$$\frac{\partial t_{spike}}{\partial V} = -1, \tag{3.7}$$

thereby encoding the intuitive relationship that a larger membrane potential causes a proportionately earlier firing time [15].

4 Sample Difficulty in Machine Learning and Deep Learning

This chapter outlines key theoretical frameworks and metrics for understanding and quantifying sample difficulty, setting the stage for empirical investigations of how such difficulty affects latency in Time-to-First-Spike coded Spiking Neural Networks. Finally, we investigate the question:

How can we measure sample difficulty?

4.1. Theoretical Perspectives on Sample Difficulty

In theoretical terms, the "difficulty" of a sample often relates to how inherently hard it is for any model to predict that sample's label correctly. One classical view comes from statistical learning theory: if a data point lies in an ambiguous region of feature space, where multiple classes have similar probability, that point has a high Bayes error and is intrinsically difficult [79]. In other words, if the true conditional probability $P(y|x)$ for a sample x is near 0.5 in a binary classification or uniformly spread across classes in multi-class, no classifier can be confident, i.e., the sample is inherently hard to classify. By contrast, a sample in a "pure" region (far from class boundaries, with $P(y|x) \approx 1$ for the correct class) is intrinsically easy.

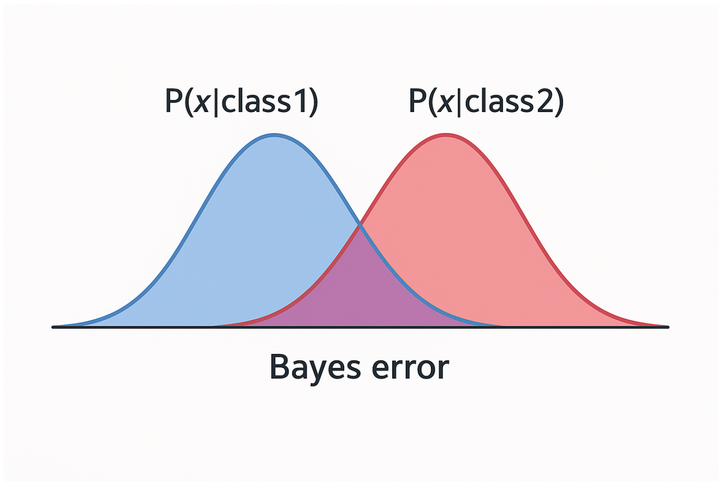


Figure 4.1: Illustration of the Bayes error rate in a binary classification task. Class 1 given by the blue distribution and class 2 given by the red distribution. Bayes error is given by the total area in the intersection of the two distributions. Samples coming from this region are ambiguous as both classes are probable in this interval.

This concept connects to the notion of margin in classification. In a geometric sense, a sample's difficulty can be associated with its distance to the decision boundary of an optimal classifier. [11] introduces the idea of maximizing margins in support vector machines; a point with a small margin, lying near the decision boundary, is more "hard" or borderline, whereas points deep inside a class region, with a large margin, are "easy". The margin serves as a direct measure of difficulty: samples on or inside the margin are those most often misclassified if the decision boundary shifts slightly, indicating high sensitivity.

Beyond margins, researchers have linked sample difficulty to more complex data characteristics. The manifold hypothesis in high-dimensional data suggests that each class forms a manifold, and classification involves separating these manifolds [16, 53, 58]. Recent work indicates that local geometric and topological properties of these manifolds can signal difficulty. For example, regions of high curvature, high intrinsic dimensionality, or complex topology can yield "hard" samples, while flatter, well-separated regions yield "easy" ones. In essence, class overlap (where different class manifolds intertwine) is a primary factor making an instance hard to classify. Indeed, a study by [71] found that as instance difficulty increases, so does class overlap: many hard examples reside in areas where different classes' data points intermingle. In such cases, even an optimal classifier may be uncertain, which aligns with the Bayes perspective.

In summary, theory suggests that a sample is difficult if it is in an intrinsically ambiguous or complex region: it might lie near class boundaries (small margin), be surrounded by other-class neighbors, or require a complex decision function to get right. Conversely, a sample is easy if it's well inside the territory of its correct class with little ambiguity. These theoretical definitions motivate many practical metrics for difficulty.

4.2. Metrics for Sample Difficulty

In practice, there are numerous metrics and heuristics to quantify the sample difficulty across different domains. These metrics are usually defined for any arbitrary model, not necessarily SNNs, and they can be broadly categorized into a few groups: (1) those based on a trained model's outputs (confidence or loss), (2) gradient-based measures, and (3) data geometry measures.

One of the most straightforward ways to gauge difficulty is to see how a well-trained model behaves on the sample. If a trained classifier assigns a low confidence to the true label (or equivalently, if the sample has a high loss), that sample can be deemed difficult for that model. For example, the prediction margin (difference between the predicted probability for the correct class and the highest incorrect class) is a useful indicator: a small margin or high entropy prediction suggests the model is unsure, often reflecting an inherently hard case. This idea is long-standing and has been used in active learning to pick out hard, uncertain samples [58].

Some lines of research examines when and how a sample is learned during training. One such measure is the Forgetting Score introduced by [76]. This score counts how many times a network learns and then “forgets” an example over the course of training. Concretely, each time a sample transitions from being classified correctly (at some point in training) to being misclassified later on, a forgetting event is recorded. The total count of forgetting events is the forgetting score. Intuitively, easy examples, once learned, stay learned, they have zero or very few forgetting events. Hard examples might flicker between learned and unlearned states as the model's decision boundary shifts, yielding multiple forgetting events.

A more recent approach is to look at a sample's impact on the model's gradients during training. In [1], they propose a Variance of Gradients (VoG) as a difficulty measure. The idea is to track how the gradient for a particular sample changes over the course of training. If a sample is easy, once the model starts to learn it, the gradients associated with that sample should become small and consistent (the model doesn't need to keep adjusting its parameters for that sample). If a sample is hard, the model's gradient on that sample will fluctuate a lot as the model oscillates in how it tries to fit it [42]. Thus, VoG computes the variance of the per-sample gradient over training. High variance indicates the sample causes learning instability, meaning the model keeps revisiting it, which correlates with difficulty. In experiments, [1] showed that the top VoG-scoring examples included many that were mislabeled or corrupted, and generally “the data points with high VoG scores are far more difficult for the model to learn”.

A natural way to quantify sample difficulty is via its distance, or margin, to the decision boundary (see Figure 4.2). For linear classifiers, this margin represents the signed Euclidean distance from the sample x_i to the classifier's separating hyperplane [82]. A small margin indicates that x_i lies near the decision boundary and is thus more susceptible to misclassification, even under slight perturbations of the model. This corresponds directly to intrinsic difficulty: samples with small margins possibly reside in regions where the true conditional probability $P(y|x)$ is ambiguous, approximately aligned with the Bayes error being high [49, 79]. In fact, the Bayes error, the irreducible error of the best possible classifier, arises precisely from these borderline regions where

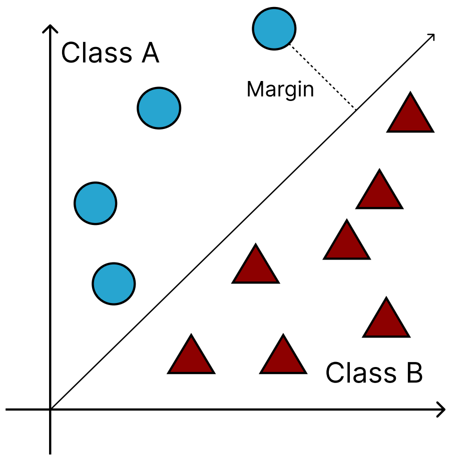


Figure 4.2: Illustration of a linear decision boundary separating two classes, Class A and Class B. The margin, defined as the shortest distance from the decision boundary to the nearest sample (indicated by the dotted line), represents the level of confidence in classification.

class-conditional distributions overlap.

The margin framework also integrates with manifold-based perspectives. Under the manifold hypothesis, each class is assumed to lie on a dimensional manifold embedded in the high-dimensional feature space [4]. Points deep within a class manifold, that is, far from any boundary, tend to have large margins and are intrinsically easy. In contrast, points near the intersections or regions of high curvature between manifolds often have small margins, indicating both a geometric proximity to other, class manifolds and inherently higher Bayes error [16, 26]. Empirically, margin-based metrics are widely used in active learning and uncertainty sampling precisely because they capture both local ambiguity and global class structure [3, 4].

Consequently, using the margin distance as a sample difficulty metric offers a principled, unified approach: it simultaneously reflects (1) the sample's geometric resilience, (2) its probabilistic certainty relative to Bayes-optimal decisions, and (3) its position in the manifold landscape. As a scalar, continuous, and interpretable measure, margin distance serves as a powerful bridge between theoretical insights and practical methodologies for quantifying and leveraging sample difficulty.

4.3. Gaussian Noise as a Proxy Measure of Difficulty

Margin-based metrics reliably capture a sample's intrinsic difficulty by measuring its distance to the (real) decision boundary. However, when working with real-world data such as MNIST or CIFAR images, directly estimating this true boundary is often infeasible [24, 69]. Therefore, rather than compute exact margins, we instead induce hardness by systematically adding Gaussian noise to each sample. This moves points towards more ambiguous, overlapping regions of the feature space, effectively simulating reduced margins and higher Bayes error, without requiring explicit knowledge of the decision boundary.

Gaussian noise has also been employed as a proxy measure for sample difficulty in machine learning. By artificially introducing Gaussian white noise to samples, researchers simulate conditions of ambiguity or uncertainty, closely relating to theoretical constructs such as Bayes error and manifold complexity. The intuition is that adding noise disturbs the clarity of the data representation, pushing samples towards more ambiguous or overlapping regions in the feature space [5].

From a theoretical perspective, the addition of Gaussian noise can be viewed through the lens of the data manifold hypothesis. Data typically reside on low-dimensional manifolds embedded within high-dimensional spaces [16]. Adding Gaussian noise effectively perturbs the points away from these manifolds, making it harder for models to discern the intrinsic structure and thus increasing the likelihood that samples become difficult to classify. This aligns well with the notion that difficult samples often lie near the manifold boundaries or in regions where different class manifolds intersect or overlap.

Furthermore, this proxy aligns neatly with the Bayes error concept. Introducing Gaussian noise to a sample effectively increases its conditional class uncertainty, shifting its true conditional probability $P(y|x)$ closer to the ambiguity region, e.g., near 0.5 for binary classification. Consequently, as noise intensity grows, the Bayes error for these samples increases, marking them as intrinsically more challenging to classify correctly.

Research examining Gaussian noise as a proxy for sample difficulty offers valuable insights [10, 30, 56]. While moderate noise addition can improve generalization by preventing models from overfitting, excessive noise generally leads to poorer performance due to elevated uncertainty and reduced confidence in predictions [25]. Moreover, noise-based augmentation is widely used in robust machine learning and adversarial training contexts because it generates difficult samples that enhance model robustness [93].

In summary, Gaussian noise offers a practical and theoretically consistent measure of sample difficulty. It effectively simulates ambiguous conditions by perturbing data points off their manifolds and increasing intrinsic uncertainty, thus directly correlating with theoretical difficulty measures such as Bayes error and manifold overlap.

4.4. Relevance to Latency-Coded SNNs

This chapter discussed various theoretical and practical views on sample difficulty, emphasizing ambiguity, margin distances, manifold complexities, and noise-based proxies. These concepts form the foundation for investigating how sample difficulty might influence latency in TTFS-coded SNNs.

Margin-based metrics reliably measure intrinsic difficulty but computing true margins or decision boundaries for datasets like MNIST or CIFAR is often infeasible. Therefore, we will use margin as a hardness metric when we can, and when it is intractable, we will induce hardness via Gaussian noise. This approach allows us to approximate difficulty either directly (via margins) or indirectly (via controlled perturbations).

Moreover, employing Gaussian noise as a difficulty proxy is particularly relevant for studying TTFS-coded SNNs. By artificially varying noise levels, we simulate increasing levels of uncertainty. Such controlled experiments enable us to systematically evaluate how TTFS-SNN latency responds to difficulty: specifically, we can observe whether increased noise leads to longer latency spikes and whether training with harder samples accelerates inference for subsequently encountered noisy samples.

5
Statistical Learning Theory

Classical statistical learning theory provides foundational insights into machine learning models, especially through its characterization of the bias-variance trade-off. This trade-off represents the balance that must be maintained between a model's complexity and its generalization capability. Highly complex models tend to fit training data closely (low bias) but fail to generalize well to new data (high variance), while simpler models generalize better but might underfit the data (high bias, low variance) [6]. In this chapter, we will investigate the question:

Could noisy training affect the synaptic weights of an SNN?

5.1. Bias-Variance Trade-off

In machine learning, the prediction error can be decomposed into three components: bias, variance, and irreducible error. The bias represents the error introduced by approximating a real-world problem with a simplified model. In contrast, variance quantifies how sensitive the model is to variations in the training data. Ideally, a model should minimize both; however, reducing one typically leads to increasing the other. This phenomenon is well-known as the bias-variance trade-off [21]. Finding an optimal balance is crucial as it directly affects the predictive performance and reliability of the trained models. Models exhibiting excessive complexity tend to overfit, capturing noise as if it were meaningful patterns, while overly simplistic models miss essential relationships inherent in the data [2].

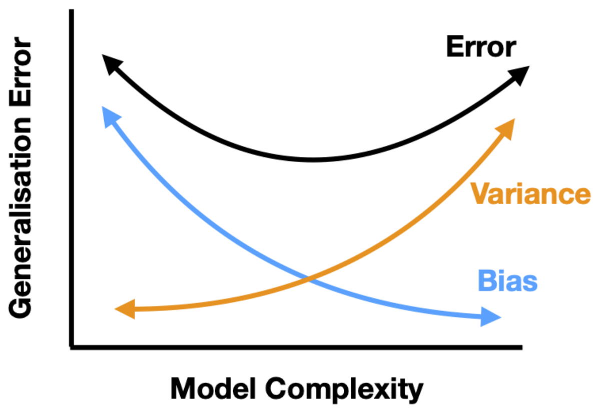


Figure 5.1: Figure illustrating the relationship between the true error of a model and its bias and variance

5.1.1. Parameter Estimation and Bias

In terms of parameter estimation, bias is the difference between the expected value of a model's parameter estimates and the true parameter value. If a parameter estimation method consistently produces results that deviate from the actual parameter, it is considered biased. When estimating a parameter β using $\hat{\beta}$ the bias is measured as:

$$\text{Bias}[\hat{\beta}] = \mathbf{E}[\hat{\beta}] - \beta \tag{5.1}$$

High bias in parameter estimation generally results from overly simplistic models or inappropriate assumptions, leading to systematic errors in predictions. Reducing bias typically involves increasing the complexity of the model [28].

5.1.2. Parameter Estimation and Variance

Variance in parameter estimation refers to the variability of parameter estimates when the estimation process is repeated on different datasets sampled from the same distribution. Similarly, when estimating a parameter β using $\hat{\beta}$, we can measure the variance of our estimator as:

$$\mathbf{Var}\left[\hat{\beta}\right] = \mathbf{E}\left[\left(\hat{\beta} - \mathbf{E}\left[\hat{\beta}\right]\right)^2\right] \tag{5.2}$$

High variance indicates that the estimated parameters change significantly with slight variations in the training dataset. This typically occurs in complex models that capture noise in the training data. Variance can be reduced by using regularization methods or by simplifying the model architecture [28].

5.2. Parameter Regularization

Regularization techniques are designed to manage the bias-variance trade-off by penalizing overly complex models, typically through adding a penalty term related to the magnitude of model parameters. Common approaches such as ℓ_2 regularization (ridge regression) penalize large parameter weights, encouraging smoother, less complex functions [33]. Regularization thus directly reduces variance by discouraging overfitting, improving generalization on unseen data. Another prevalent form of regularization is ℓ_1 regularization (lasso), which encourages sparsity in model parameters, effectively pruning irrelevant features and further enhancing generalization. Regularization methods can also be adaptive, adjusting penalty terms dynamically during training, providing flexibility in handling datasets with varying complexity and noise characteristics [95].

To better understand the role of regularization, we begin with the classical linear regression problem. In its standard form, linear regression attempts to model the relationship between input features and a continuous output by learning a parameter vector β. This model is written as:

$$y = X\beta + \varepsilon \tag{5.3}$$

where, $X \in \mathbb{R}^{n\times d}$ is the input data matrix, $\beta \in \mathbb{R}^d$ is the parameter vector we want to learn, $y \in \mathbb{R}^n$ is the target vector, and ε is the noise term. Since an exact solution $X\beta = y$ may not exist, the goal becomes finding the parameter vector β that minimizes the squared error between the predicted and observed outputs:

$$\min_{\beta} \|y - X\beta\|^2 \tag{5.4}$$

We can set the gradient with respect to β to 0, to find the β that minimizes the squared error.

$$\nabla_\beta = -2X^\top y + 2X^\top X\beta = 0 \tag{5.5}$$
$$-X^\top y + X^\top X\beta = 0 \tag{5.6}$$
$$X^\top X\beta = X^\top y \tag{5.7}$$
$$\beta = \left(X^\top X\right)^{-1} X^\top y \tag{5.8}$$

To address the limitations of OLS, e.g., the tendency to overfit when the number of features is large, ridge regression introduces a regularization term to the objective function. Specifically, ridge regression adds a ℓ_2 penalty to the squared loss, resulting in the following objective:

$$\min_{\beta} \|y - X\beta\|^2 + \lambda \|\beta\|^2 \tag{5.9}$$

where $\lambda \geq 0$ is a regularization hyperparameter controlling the strength of the penalty. Similar to linear regression, we can set the gradient to zero to find the solution to β:

$$\nabla_\beta = -2X^\top y + 2X^\top X\beta + 2\lambda\beta = 0 \tag{5.10}$$
$$-X^\top y + X^\top X\beta + \lambda\beta = 0 \tag{5.11}$$
$$X^\top X\beta + \lambda\beta = X^\top y \tag{5.12}$$
$$\beta\left(X^\top X + \lambda I\right) = X^\top y \tag{5.13}$$
$$\beta = \left(X^\top X + \lambda I\right)^{-1} X^\top y \tag{5.14}$$

5.2.1. Gaussian Noise as a Regularizer

Adding Gaussian noise to input data or activations during training can also act as a regularization technique. This approach, akin to data augmentation, helps smooth the model's learned function by reducing its sensitivity to specific data points. More fundamentally, the addition of Gaussian noise with zero mean can be shown to be mathematically equivalent to applying ℓ_2 regularization under certain assumptions, particularly in linear models [5]. In the following, we present the derivation that illustrates how adding Gaussian noise to inputs during training leads to a loss function that includes an ℓ_2 penalty term - effectively performing ridge regression.

Let's consider the same linear model from Equation 5.3. Suppose that we add noise to the inputs $X + \varepsilon$, where $\varepsilon \sim N(0, \Sigma)$. Now we can write our objective as:

$$\min_\beta \left\|y - (X + \varepsilon)\beta\right\|^2 \tag{5.15}$$

equivalently,

$$\min_\beta \left(y - X\beta\right)^\top \left(y - X\beta\right) - 2\left(y - X\beta\right)^\top \varepsilon\beta + \beta^\top \varepsilon^\top \varepsilon\beta \tag{5.16}$$

The term ε is a stochastic variable. Therefore, let's take the expectations with respect to ε. The expectations of each term are given by:

$$\mathbf{E}_\varepsilon\left[\left(y - X\beta\right)^\top \left(y - X\beta\right)\right] = \left(y - X\beta\right)^\top \left(y - X\beta\right) \tag{5.17}$$
$$\mathbf{E}_\varepsilon\left[2\left(y - X\beta\right)^\top \varepsilon\beta\right] = 0 \tag{5.18}$$
$$\mathbf{E}_\varepsilon\left[\beta^\top \varepsilon^\top \varepsilon\beta\right] = \beta^\top n\Sigma\beta \tag{5.19}$$

Thus, the expected squared error is:

$$\left(y - X\beta\right)^\top \left(y - X\beta\right) + \beta^\top n\Sigma\beta \tag{5.20}$$

or equivalently,

$$\left\|y - X\beta\right\|^2 + n\Sigma\left\|\beta\right\|^2 \tag{5.21}$$

We can further go on to show that the solution by setting the gradient to zero is given by:

$$\nabla_\beta = -2X^\top y + 2X^\top X\beta + 2n\Sigma\beta = 0 \tag{5.22}$$
$$-X^\top y + X^\top X\beta + n\Sigma\beta = 0 \tag{5.23}$$
$$X^\top X\beta + n\Sigma\beta = X^\top y \tag{5.24}$$
$$\beta\left(X^\top X + \Sigma\right) = X^\top y \tag{5.25}$$
$$\beta = \left(X^\top X + n\Sigma\right)^{-1} X^\top y \tag{5.26}$$

The derived solution under Gaussian noise (Equation 5.26) is equivalent to the solution of ridge regression (Equation 5.19) when $\lambda I = n\Sigma$. This shows that for $\lambda I = n\Sigma$ (or in 1D: $\lambda = n\sigma^2$), minimizing the expected loss under input noise is identical to minimizing the standard loss with added ℓ_2 penalty. Hence, Gaussian noise acts as a form of implicit regularization, with the noise variance Σ serving as the regularization strength.

5.3. Bias-Variance and Generalization in Neural Networks

The bias-variance principles discussed above extend naturally to modern neural architectures, but their displays differ according to the architectural priors and learning rules embedded in each model class. In over-parameterised deep networks, generalization hinges as much on how the parameters are used as on the raw parameter count itself, leading to phenomena such as double-descent in test error curves [2, 54]. Below we summarize the most important statistical-learning considerations for three families that are relevant to our case: fully-connected multilayer perceptrons (MLPs), convolutional neural networks (CNNs), and time-to-first-spike (TTFS) spiking neural networks (SNNs).

5.3.1. Deep and Convolutional Artificial Networks

Implicit Regularization and Flat Minima

Although very deep or wide networks can interpolate the training data perfectly, stochastic gradient descent (SGD) rarely drives them to all possible interpolating solutions. Instead, optimization and noise tend to locate flat minima in weight space whose associated functions are smoother and therefore lower-variance [31, 39]. From a statistical perspective, SGD acts as an implicit regularizer as it biases learning toward low-complexity solutions without an explicit penalty term.

Weight Sharing and Equivariance

CNNs further reduce variance by hard-coding locality and translation equivariance through weight sharing. The same $k \times k$ filter is applied across the whole receptive field, meaning only $O(k^2)$ independent weights model thousands of spatial correlations. This architectural bias lowers the effective model capacity relative to a similarly sized fully-connected layer, thereby shifting the bias–variance operating point toward better generalization on images [24].

Explicit Penalties

When implicit regularisation is insufficient, explicit techniques such as weight decay, dropout, or data augmentation are introduced. In CNNs, ℓ_2 decay keeps kernel norms small, suppressing sharp minima and improving robustness to small input perturbations [22, 41]. Regularization based on weight correlations constraints can additionally reduce redundancy, such as weight-vector correlations, further lowering variance and improving generalization [37].

5.3.2. TTFS Spiking Neural Networks

Event-Driven Sparsity

TTFS coding stipulates that each neuron is allowed to fire at most once and that the information is carried solely in the first-spike latency. As a consequence, only a fraction of synapses are active per timestep, yielding a form of activity sparsity that functions as an architectural prior against overfitting [50]. In statistical-learning terms, the network's effective capacity at test time is far smaller than its parameter count would suggest, because the majority of weights do not contribute for most stimuli.

Temporal margins and Robustness

In TTFS classifiers, the decision is triggered as soon as one output neuron fires; training objectives therefore seek to maximize the temporal margin, the time-to-first-spike gap between the target class and its nearest rival. In [67], they formalize the idea in their Temporal Support Vector Machine (T-SVM), showing that maximizing the dynamical margin in an SNN yields generalization bounds analogous to the large-margin theory of hard-margin SVMs. Empirically, deeper TTFS networks that achieve larger first-spike gaps exhibit better resilience to corruptions [57]. These results support the view that a larger temporal margin in TTFS models plays the same variance-reducing role that a wider geometric margin plays in classical statistical learning theory.

5.4. Synaptic Weights in TTFS-SNNs

The theoretical insights from statistical learning theory presented in this chapter lay a critical foundation for understanding the anticipated effects of Gaussian noise training on synaptic weights within TTFS SNNs. Regularization techniques systematically influence the characteristics of the learned synaptic weights. Specifically, introducing Gaussian noise during training functions analogously to explicit ℓ_2 regularization, which encourages synaptic weights to remain relatively small and stable by penalizing large parameter fluctuations. While this lets us approximately understand how the weights might behave in a spiking network, it still remains to be empirically tested.

As outlined previously, statistical learning theory emphasizes that regularization typically reduces parameter variance, leading to smoother, less sensitive weight distributions. Hence, based on our investigations in this chapter, we anticipate that synaptic weights trained with Gaussian noise will exhibit lower variability and increased robustness compared to weights learned without noise regularization. This reduction in variance may translate into more stable synaptic behavior, reflected in less volatile membrane potentials when neurons process input signals. Therefore, it is inevitable that this process will also affect spike times. In chapter 6 on stochastic processes, we model the neuron's membrane potential dynamics explicitly as stochastic systems and try to formalize how changes in synaptic weights should affect spike times.

6 Stochastic Processes

Stochastic processes describe systems evolving randomly over time, widely utilized in various fields such as finance, physics, biology, and computer science [7, 51, 66]. They form the foundational theory behind modeling unpredictable events and temporal evolution of variables influenced by randomness. In this chapter, we outline the framework to be able answer the question

> *How does changes in synaptic weights affect the first spike time of a spiking neuron when we model it as a stochastic process?*

Modeling synaptic weight changes within a first-hitting-time framework allows membrane potential's stochastic dynamics to be analytically related to expected first-spike latencies. Incorporating these stochastic process analyses into our research provides a mathematical foundation to quantify how synaptic weight dynamics and noise introduced during training modulate first-spike behavior, directly connecting weight dynamics to temporal coding performance under varying sample difficulties.

6.1. Random Walks

A random walk (RW) represents one of the simplest forms of a stochastic process, describing a path composed of successive random steps. Formally, consider the discrete-time process:

$$S_n = X_1 + X_2 + \cdots + X_n, \tag{6.1}$$

where X_i are independent and identically distributed (i.i.d.) random variables representing incremental steps. This model captures various phenomena including particle diffusion and financial price fluctuations.

6.1.1. Gaussian Random Walk

A specific and important instance is the Gaussian random walk (GRW), where increments are drawn from a Gaussian distribution, $X_i \sim N(\mu, \sigma^2)$. The evolution of the random walk then is given by:

$$S_n = S_{n-1} + X_n, \quad X_n \sim N(\mu, \sigma^2). \tag{6.2}$$

Gaussian random walks are particularly relevant because of their mathematical tractability and applicability in diffusion processes [17]. They exhibit stationary, normally distributed increments and are Markovian, meaning the next state depends only on the current position and not on the past history. Additionally, the variance of the walk grows linearly with time, a property that aligns closely with empirical observations in many physical and biological systems [20].

6.1.2. Wiener Processes (Brownian Motion)

When considering the continuous-time limit of a Gaussian random walk with infinitesimally small increments, the Wiener process, also known as Brownian motion, emerges. A Wiener process W_t has the following properties:

- $W_0 = 0$,

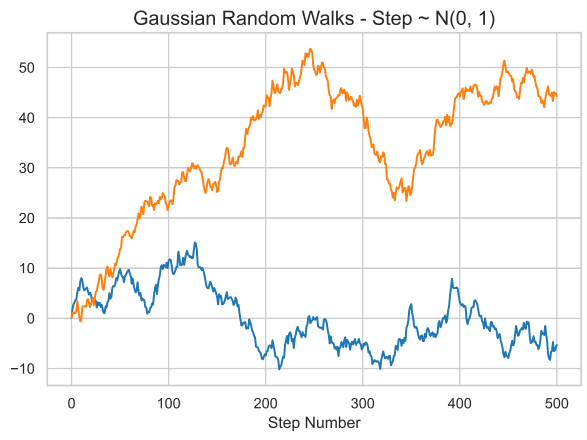


Figure 6.1: Two realizations of Gaussian random walks with zero mean and unit variance steps, i.e., increments drawn i.i.d. from $N(0, 1)$. Each walk begins at zero and exhibits distinct trajectories due to stochastic variation.

- Independent increments: for $0 \leq t_1 < t_2 < \cdots < t_n$, the increments $W_{t_2} - W_{t_1}, \ldots, W_{t_n} - W_{t_{n-1}}$ are independent,
- Gaussian increments: $W_t - W_s \sim N(0, t - s)$,
- Continuity: The paths $t \mapsto W_t$ are continuous.

Formally, Brownian motion serves as the standard model for random fluctuations observed in various scientific disciplines, particularly in modeling diffusive behavior and noise in biological systems [38].

6.2. First Hitting Time Problem

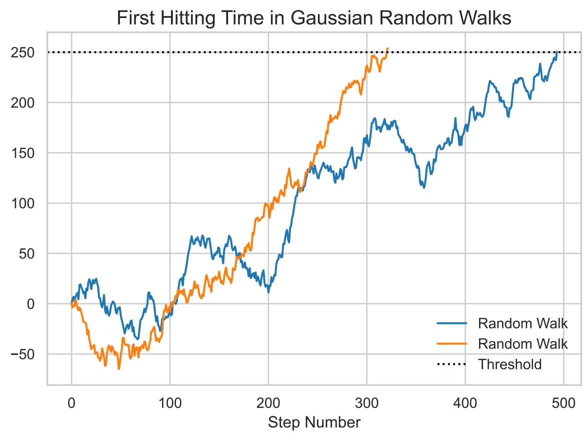


Figure 6.2: Illustration of the first hitting time in Gaussian random walks with i.i.d. steps. First hitting time is the first time the walk reaches over the threshold. The orange trajectory hits the threshold (dotted line) first, while the blue trajectory takes a bit longer.

The first hitting time problem is concerned with determining the time at which a stochastic process first reaches or surpasses a predefined threshold (see Figure 6.2). Mathematically, for a stochastic process (S_t) and a threshold level b, the first hitting time T_b is defined as:

$$T_b = \inf\{t \geq 0 : S_t \geq b\}. \tag{6.3}$$

The distribution of first hitting times is crucial for understanding various phenomena such as neuron firing in neuroscience, financial barrier options, and failure times in reliability analysis [64, 77]. In neuron models, the first hitting time can represent the moment when the membrane potential reaches a threshold, triggering a spike.

6.3. Modeling Neuron Spiking with First Hitting Times

In this part, we will model the membrane potential dynamics of a spiking neuron as a stochastic process and treat it as the first hitting time problem described above. Modeling the first spike time in a TTFS spiking neural network is essential for two main reasons. First, in latency encoding, the timing of the spikes is very crucial. Capturing its dynamics would allow us to better quantify how quickly a neuron responds to varying inputs. Secondly, by analyzing how the distribution of first spike times shifts under different circumstances, we can better understand its behavior and make more informed predictions about it.

Applying first-hitting-time models to neuronal spiking has been extensively studied in mathematical neuroscience [9, 12, 64, 72, 77, 78]. However, to the best of our knowledge, these approaches have not yet been explored in the context of TTFS spiking neural networks, where modeling the distribution and variability of first-spike latencies across network layers could yield new insights into coding efficiency and dynamic behavior.

6.3.1. Assumptions and Fundamental Model

Let's consider a non-leaky integrate-and-fire neuron model. Assuming each input to a neuron spikes at some point in time, the membrane potential after n inputs is given by

$$S_n = \sum_{i=1}^{n} w_i, \tag{6.4}$$

where w_i are the weights of the incoming edges of the neuron, since spikes are binary events taking values 0 or 1. We assume the incoming synaptic weights are independent and identically distributed Gaussian random variables with mean μ and variance σ^2, making $\{S_n\}$ a Gaussian random walk. The i.i.d assumption of edge weights is likely not correct in a trained SNN, however, for the purposes of the model, it will allow us to create a good approximation. To ensure that the first-hitting-time to a fixed threshold is almost surely finite and possesses finite moments, we assume a positive mean $\mu > 0$; if $\mu \leq 0$, the probability of ever crossing the threshold is strictly less than one and the expected hitting time diverges [64]. Finally, we adopt a unit threshold $V_{\text{th}} = 1$, as is common in many spiking neuron models.

6.3.2. Building the Model

First-Hitting-Time of Wiener Process

Although our spiking neurons operate in discrete time and are more akin to a Gaussian random walk, the continuous-time analogue given by the Wiener process admits a closed-form solution for its first-hitting-time distribution, whereas no closed-form solution is known for the Gaussian random walk. Consider a Wiener process $W(t)$ with drift μ and variance parameter σ^2 and a threshold $V_{\text{th}} = 1$. Given these parameters, the distribution of the first-hitting-time is given by the inverse Gaussian Distribution [18, 38]:

$$f_T(t) = \frac{1}{\sqrt{2\pi\,\sigma^2\,t^3}}\;\exp\Big(-\frac{(1-\mu\,t)^2}{2\,\sigma^2\,t}\Big), \quad t > 0. \tag{6.5}$$

The first two moments of this distribution take the simple form:

$$\mathbb{E}[T] = \frac{1}{\mu} \tag{6.6}$$

$$\mathrm{Var}(T) = \frac{\sigma^2}{\mu^3} \tag{6.7}$$

Figure 6.3 compares the simulated first-passage time histograms (in blue) with their inverse Gaussian fits (in red) across three different random walk variance levels. The histograms and the inverse Gaussian densities closely follow each other, showing that the inverse Gaussian provides a highly accurate approximation of the discrete-time first-hitting-time distribution as well.

Incorporating Membrane Decay

Most spiking neuron models include a leakage term that gradually resets the membrane potential toward rest. To capture this, we introduce the decay factor $\beta \in (0, 1)$ from Equation 3.1 so that after each incoming spike, the membrane potential evolves as

$$S_n = \beta\,S_{n-1} + w_n, \tag{6.8}$$

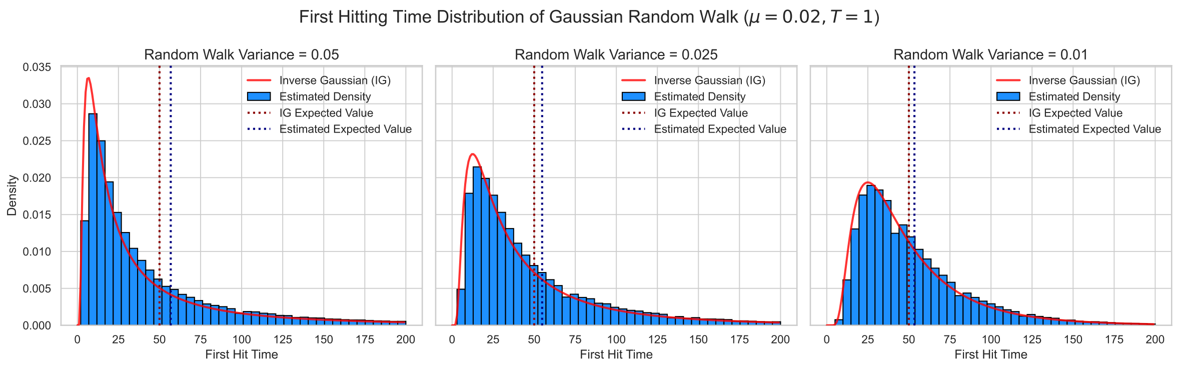


Figure 6.3: First hitting time distributions for Gaussian random walks with drift $\mu = 0.02$ and threshold $T = 1$ under different RW variance conditions. Blue bars show the histogram of simulated first-passage times, red curves denote the theoretical inverse Gaussian density, and vertical dashed lines mark the theoretical mean of the inverse Gaussian (red) and the empirical mean from simulations (blue).

where $w_n \sim N(\mu, \sigma^2)$ as before, and we retain a fixed threshold $V_{\text{th}} = 1$. The leaky random walk thus must "fight" against decay to accumulate enough potential to fire.

Figure 6.4 illustrates how decay $(\beta = 0.95)$ reshapes the first-spike-time distribution under three variance settings. Key observations include:

- **No Closed-Form Fit:** Leakage breaks the pure-diffusion assumptions, so the inverse Gaussian approximation no longer holds and is dropped in the next steps.
- **Leakage vs. Fluctuations:** High-variance inputs still overcome decay to spike more reliably, whereas low-variance inputs often decay away before the threshold is reached, dramatically delaying spikes.
- **Increased Variability:** Decay amplifies timing variability, especially in low variance regimes where drift is weak relative to leak.

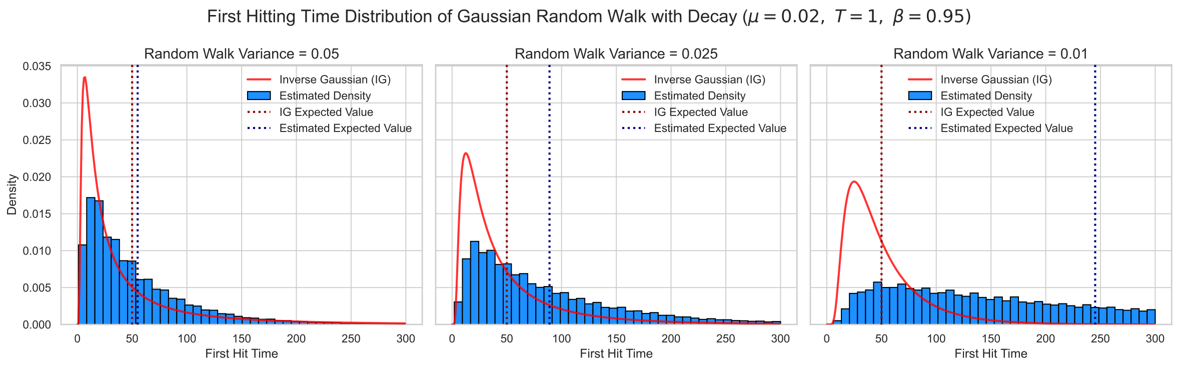


Figure 6.4: First-spike-time distributions for a leaky Gaussian random walk ($\beta = 0.95$, $V_{\text{th}} = 1$) under high, medium, and low variance. Decay suppresses early crossings in low-variance regimes, yielding heavier tails and fewer timely spikes.

Spike-Step Limitation and Discrete Time Bins

Our current membrane potential decay isn't exactly accurate regarding how the spiking neuron we use decays. We have to introduce timestep bins. At each timestep $t \in \{1, \dots, T\}$, we (1) accumulate all incoming spikes to form a synaptic current, then (2) apply decay once at the end of the step, so that

$$S_t = \beta\, S_{t-1} + \sum_{i=1}^{n_t} w_i, \tag{6.9}$$

where n_t is the number of spikes in bin/timestep t.

In the first convolutional layer of a spiking neural network, each neuron samples from a receptive field (see Figure 2.3 for an illustration of the receptive field) of size $K \times K \times C$. For example, $K = 3$ on an RGB image

($C = 3$) gives

$$N_{\max} \ = \ K \times K \times C \ = \ 3 \times 3 \times 3 \ = \ 27 \tag{6.10}$$

possible *input-spike events* per neuron. We refer to each such potential event as a random-walk step, since the input pattern over that receptive field is effectively a walk through at most $N_{\max}$ spike placements.

It is important to note that

$$N_{\max} = 27$$

is *not* the number of simulation time-steps: instead, $N_{\max}$ bounds the *total* number of input spikes a neuron can possibly receive across the entire receptive field. By contrast, we discretize real time into T bins with

$$t \in \{1, \dots, T\},$$

and within each bin t we may accumulate zero, one, or multiple spikes. Thus:

- *Random-walk steps* (maximum $N_{\max} = 27$) enumerate the *potential* spike events determined by the static receptive-field geometry.
- *Timesteps* (T) define the discrete intervals at which the neuron integrates incoming spikes and updates its membrane potential.

Hence, even if T exceeds $N_{\max}$, we will only ever see at most $N_{\max}$ incoming spikes in total; conversely, if $T < N_{\max}$, some potential spikes must coincide within the same time bin.

To approximate a uniform average-case assignment of the $N_{\max} \leq 27$ potential spikes into T bins, We proceed with two probabilistic steps:

1. Draw a probability vector from the Dirichlet distribution

$$\mathbf{p} = (p_1, \dots, p_B) \sim \mathrm{Dir}(\alpha, \dots, \alpha), \tag{6.11}$$

   where $\alpha > 0$ is a concentration parameter

2. Given $\mathbf{p}$, we convert these continuous probabilities into integer spike counts by sampling via the multinomial distribution:

$$(n_1, \dots, n_b) \sim \mathrm{Mult}\big(N_{\max}, \mathbf{p}\big), \quad \sum_{t=1}^{T} n_t = N_{\max}, \tag{6.12}$$

Needless to say, a real image probably will not have a uniform distribution of spikes. Pixels will have similar values to their neighbors, meaning that their spikes will arrive together and not spread apart in time. However, this should yield a good approximation of what should happen in an average case.

Another point to look out for is that binning can overestimate the true first-spike time, since the threshold may be crossed partway through a bin but the model only emits a spike at the bin's end. However, this isn't necessarily a problem, as the real neuron model works in the same way as well.

Figure 6.5 shows the impact of varying the number of bins T: as T increases, both the fraction of runs that fire within $N_{\max}$ steps and the mean first-spike time shift. Furthermore, similar to an effect observed Figure 6.4, increasing the number of timesteps both increases the number of times the membrane potential is decayed and increases the temporal sparsity of spikes, exacerbating the effect of decay.

Incorporating Kernel Bias

In convolutional networks, each filter typically includes a bias term b added after the weighted sum of its inputs. To account for this in our SNN model, we introduce a bias drawn once per neuron from a Gaussian distribution

$$b \sim N(\mu_b, \sigma_b^2), \tag{6.13}$$

and include it in the membrane-potential update at each timestep:

$$S_b = \beta\, S_{t-1} + \sum_{i=1}^{n_b} w_i + b \tag{6.14}$$

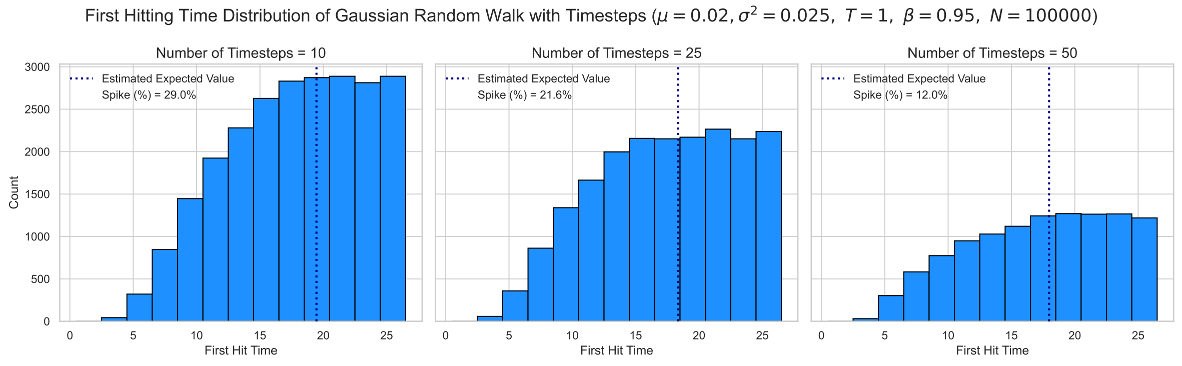


Figure 6.5: Monte-Carlo simulations of first-spike time histograms for a leaky Gaussian random walk ($\mu = 0.02$, $\sigma^2 = 0.025$, $V_{\text{th}} = 1$, $\beta = 0.95$, $N = 10^5$) when spikes are binned into $B = 10, 25, 50$ discrete timesteps. Bars show empirical counts, the vertical dashed line marks the mean spike time, and the annotation gives the percentage of simulations that fired within the allotted bins.

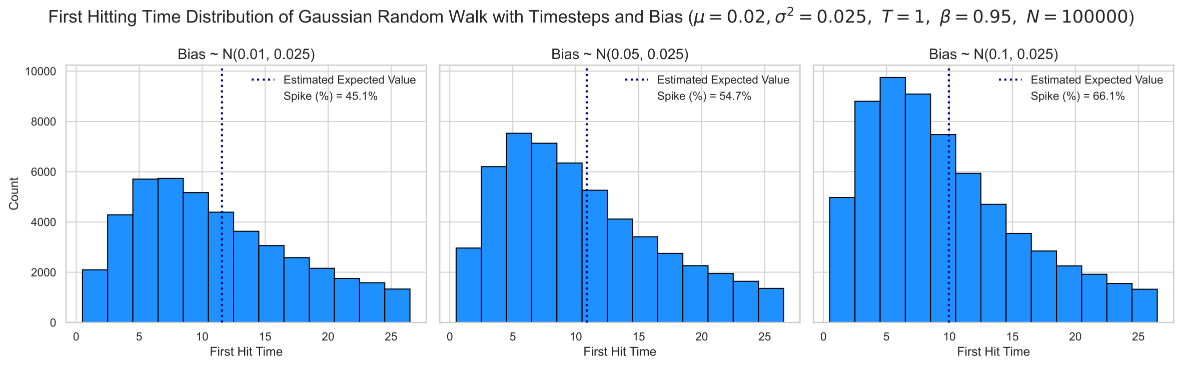


Figure 6.6: First-spike-time histograms for a leaky Gaussian random walk with bias $b \sim N(\mu_b, 0.025)$. Panels correspond to $\mu_b = 0.01, 0.05, 0.10$. Increasing bias adds drift, yielding earlier spikes and higher firing rates within the allotted bins (dashed line: mean spike time; annotation: % of runs that spiked).

where $w_i \sim N(\mu, \sigma^2)$ are the synaptic weights, β is the decay factor, and n_b is the number of spikes in bin t. The bias effectively adds a constant drift, shifting first-spike times earlier and increasing firing probability.

Figure 6.6 shows the impact of varying the bias mean μ_b (with fixed $\sigma_b^2 = 0.025$) on the first-spike-time distribution. As expected, increasing the mean decreases the expected first hitting time and increases the number of neurons that spike within 27 input spikes.

6.4. Verifying the Model

To assess the accuracy of our first-spike-time model, we compare its predictions on the first-spike time of the simulated neurons, against the observed spiking behavior in the first convolutional layer of a Spiking CNN (SCNN) trained on CIFAR-10. We will use a SCNN trained with parameters: $\beta = 0.95$, $V_{\text{th}} = 1$, and 10 timesteps. To this end, we will count the number of spikes each neuron receives before it spikes in the trained CNN and compare it with simulation results. Before we dive into the comparison, we will try to validate some of the assumptions we have made.

6.4.1. Gaussian Step Assumption

Figure 6.7 shows the empirical distributions of the biases and weights in layer 1, respectively. We fit Gaussians to each:

$$w \sim N(\hat{\mu}_w, \hat{\sigma}_w^2), \quad b \sim N(\hat{\mu}_b, \hat{\sigma}_b^2),$$

with estimated parameters $\hat{\mu}_w = 0.031$, $\hat{\sigma}_w^2 = 0.020$, $\hat{\mu}_b = 0.026$, and $\hat{\sigma}_b^2 = 0.010$. Although the agreement between histograms and normal curves isn't one-to-one, we can see that in both cases the distributions are approximately normal. These observations provide some empirical support for the Gaussian weight and bias assumptions introduced in subsection 6.3.1 and subsection 6.3.2.

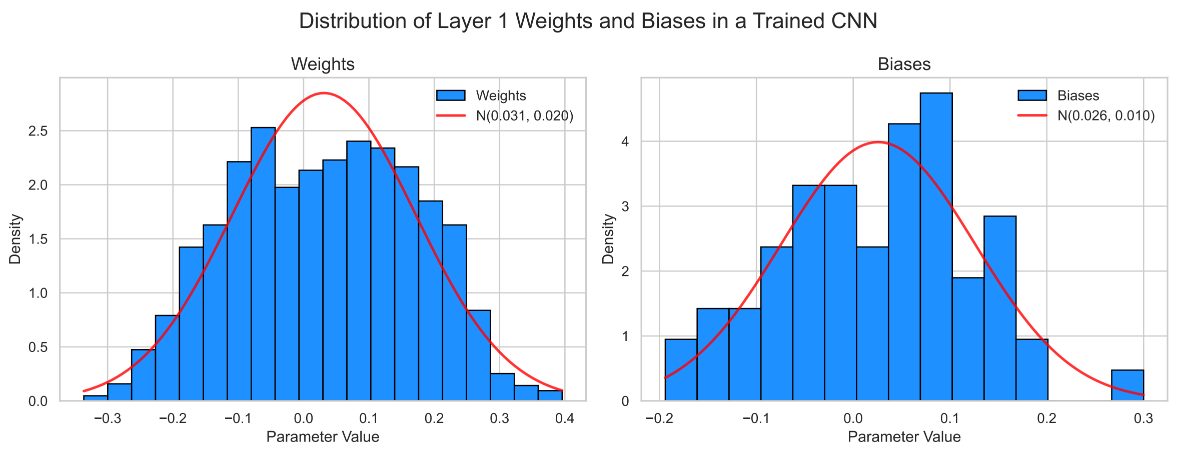


Figure 6.7: Empirical distributions of layer-1 convolutional kernel weights (left) and biases (right) in a Spiking CNN trained on CIFAR-10. Blue bars show the observed parameter histograms, and red curves denote the fitted Gaussian densities $N(\hat{\mu}_w, \hat{\sigma}_w^2)$ and $N(\hat{\mu}_b, \hat{\sigma}_b^2)$ with $\hat{\mu}_w = 0.031$, $\hat{\sigma}_w^2 = 0.020$, $\hat{\mu}_b = 0.026$, and $\hat{\sigma}_b^2 = 0.010$.

6.4.2. Spike Distribution of Images

In subsection 6.3.2, we previously mentioned, real images are unlikely to produce uniform spike counts across timesteps. For example, Figure 6.8 shows a CIFAR-10 cat image, and Figure 6.9 plots the empirical number of input spikes in each of 10 timesteps for that image. Clearly, the observed counts $(m_1, \ldots, m_T)$ deviate from a flat profile.

To incorporate this image-specific timing, we define the normalized spike ratios

$$\hat{p}_t = \frac{m_t}{\sum_{j=1}^{T} m_j}, \qquad t = 1, \ldots, T, \tag{6.15}$$

so that $\sum_{t=1}^{T} \hat{p}_t = 1$. We then allocate the $N_{\max} = 27$ potential spikes via

$$(n_1, \ldots, n_T) \sim \operatorname{Mult}(N_{\max}, \hat{p}), \qquad \sum_{t=1}^{T} n_t = N_{\max} \tag{6.16}$$

replacing the Dirichlet-multinomial scheme with a multinomial draw driven by the actual spike distribution of the image. This ensures our simulated first-spike times respect the temporal structure present in real inputs.

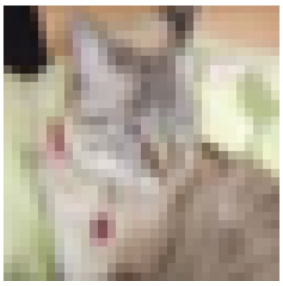


Figure 6.8: An image of a cat from CIFAR-10

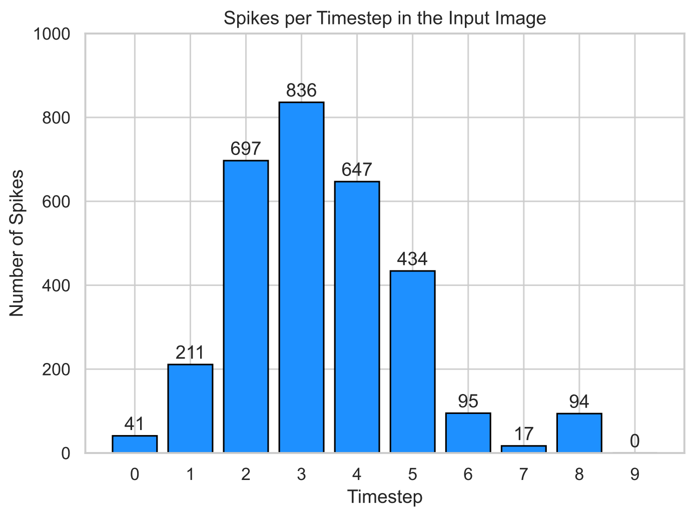


Figure 6.9: Empirical total spike counts per timestep for the CIFAR-10 "cat" image in Figure 6.8 with 10 timesteps. The non-uniform distribution $(m_1, \dots, m_{10})$ motivates our data-driven allocation of the $N_{\max} = 27$ spikes according to the normalized ratios $\hat{p}_t = m_t / \sum_j m_j$.

6.4.3. Model Results

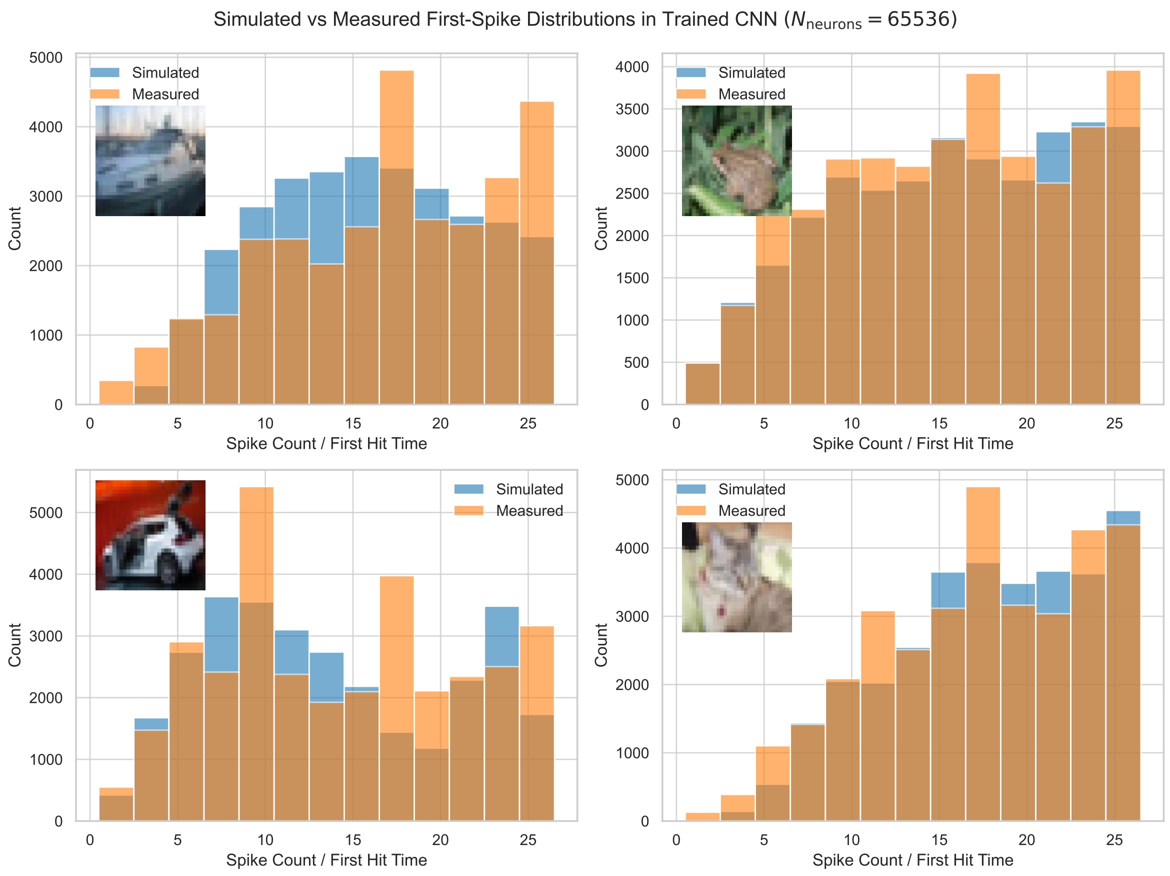


Figure 6.10: Simulated vs. measured first-spike-time distributions for layer 1 neurons across four CIFAR-10 images (Boat, Frog, Car, Cat). Blue histograms show model simulations using the fitted parameters $\hat{\mu}_w = 0.031$, $\hat{\sigma}_w^2 = 0.020$, $\hat{\mu}_b = 0.026$, $\hat{\sigma}_b^2 = 0.010$, $\beta = 0.95$, $V_{\text{th}} = 1$, and 10 timesteps. Orange histograms show the empirical first-spike times recorded from $N_{\text{neurons}} = 65\,536$ units. The close alignment demonstrates that our stochastic first-hitting-time model accurately mimics the spiking dynamics of a trained SCNN's first layer.

Figure 6.10 illustrates the comparison between measured and simulated first-spike-time distributions for layer 1 neurons across four CIFAR-10 images (Boat, Frog, Car, and Cat). For each image, we first recorded (over all $N_{\text{neurons}} = 65\,536$ neurons) the number of incoming spikes needed before each neuron emitted its first spike. We then generated the same number of simulated first-spike times using the random-walk model and the parameters estimated from the trained SCNN itself.

The blue histograms (simulated) and orange histograms (measured) exhibit a close overlay in all panels. This strong alignment confirms that our stochastic first-hitting-time framework accurately captures the timing behavior of real TTFS neurons, despite simplifying assumptions such as i.i.d. Gaussian weights and temporally independent input spikes.

6.5. Effect of Synaptic Weight Variance on First-Spike Timing

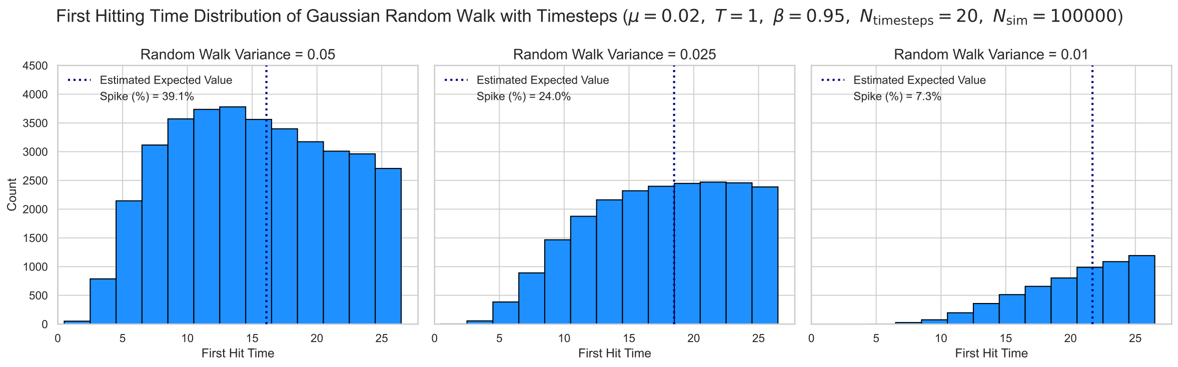


Figure 6.11: First-hitting-time distributions for a leaky Gaussian random walk ($\mu = 0.02$, $V_{\text{th}} = 1$, $\beta = 0.95$, $N_{\text{timesteps}} = 20$, $N_{\text{sim}} = 10^5$) under three variances σ^2. Lower σ^2 produces heavier tails and delays in spiking.

To investigate how synaptic weight variance σ^2 influences the latency of the first spike, we simulate our leaky Gaussian random-walk model with decay $\beta = 0.95$, threshold $V_{\text{th}} = 1$, and $N_{\text{timesteps}} = 20$ bins over $N_{\text{sim}} = 10^5$ trials. Figure 6.11 presents the resulting first-hitting-time histograms for three values of random walk variance. As the variance decreases from 0.05 to 0.01, the distributions shift steadily to the right, showing that lower variability in the weight increments yields fewer large jumps to overcome the leak, and thus more steps are required on average to reach the threshold.

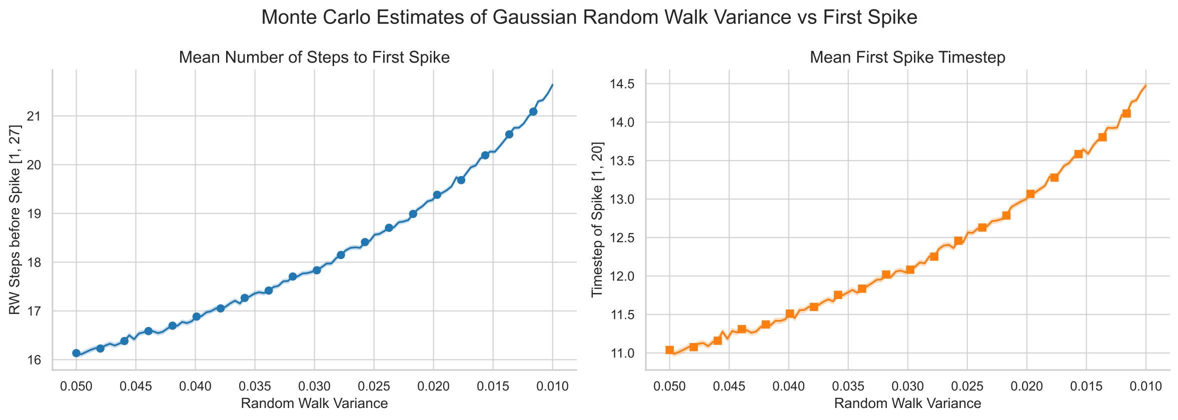


Figure 6.12: Monte Carlo ($N = 100\ 000$) estimates of (a) mean number of steps to first spike and (b) mean first-spike timestep as functions of synaptic variance σ^2. Decreasing σ^2 leads to slower accumulation relative to leak, increasing the expected latency. N.B., the x-axis is inverted.

We quantify this effect more precisely in Figure 6.12, which plots (a) the mean number of random-walk steps before the first spike and (b) the mean first-spike timestep versus σ^2. Both measures increase monotonically as σ^2 decreases, confirming that smaller step-size fluctuations exacerbate the competition between synaptic accumulation and membrane decay, thereby prolonging the first-hitting time.

6.5.1. Impact of Noise Training on Inference Latency

One of our primary research questions was whether injecting noise during training, without any data augmentation, could reduce inference latency for difficult samples. As previously discussed in chapter 5, we hypothesize that noise training will drive down the magnitude of learned weights to counteract variability during learning, thereby reducing effective random walk step size. According to our model, any decrease in the synaptic weights' variance must increase the expected first hitting time, i.e., slower first spikes. Consequently, the hypothesis that noise training reduces latency is not supported by our stochastic first-hitting-time framework, and furthermore, it should even increase latency during inference due to slower spike times.

7 Decision Making

In this chapter, we outline the theoretical framework used to investigate one of our primary research questions:

> *Does time-to-first-spike latency increase with sample difficulty?*

To answer this, we adopt the Drift Diffusion Model (DDM) and evidence accumulation theory. We discuss: (1) drift diffusion and its links to stochastic processes; (2) sample difficulty via margin-based metrics as defined in chapter 4; (3) how margin connects to evidence accumulation; and (4) concrete predictions from the DDM in the context of TTFS SNNs.

7.1. Drift Diffusion and Evidence Accumulation

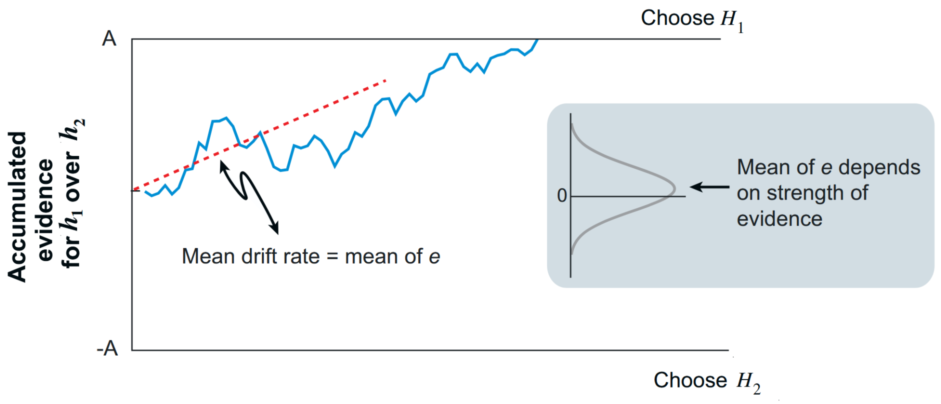

Choose H_1
A
Accumulated evidence for h_1 over h_2
Mean drift rate = mean of e
0
Mean of e depends on strength of evidence
-A
Choose H_2

Figure 7.1: Schematic of the drift diffusion model. The decision variable accumulates noisy momentary evidence $e(t)$ over time with mean drift rate μ (red dashed line), until it reaches the upper boundary $+A$ (choose H_1) or lower boundary $-A$ (choose H_2). The inset illustrates the probability density of momentary evidence e, whose mean shifts according to stimulus strength. Figure from [23].

Drift Diffusion Models (DDMs) are cognitive-level models that describe decision-making as an accumulation of evidence over time until a threshold is reached [60]. In the classical formulation [59], evidence starts at a point z, drifts with mean rate v toward one of two boundaries (separated by distance a), and terminates upon boundary crossing, yielding a choice and response time determined by the first hitting time [60]. DDMs have been highly successful in explaining neural response times and accuracy in tasks like perceptual decisions [29, 36, 52, 80]

DDMs decompose response time into decision time (accumulation to threshold) plus non-decision components (encoding and motor delays). As task difficulty increases (e.g., less discriminable stimuli), drift rate v reduces, resulting in slower and more variable decision times [52].

7.1.1. Bridging DDM to SNNs

Modern computer vision increasingly involves dynamic, sequential data (e.g. video streams or event camera outputs), where decisions must be made over time under uncertainty. Integrating DDMs into SNNs would provide a theoretically grounded approach: the SNN could accumulate visual evidence (such as object features or motion cues) in a drift-diffusion-like fashion until a decision threshold is reached.

In neural terms, a spiking neuron (e.g. a leaky integrate-and-fire unit) performs a similar computation: it accumulates input (drift), includes noise or leak, and fires an output spike once the membrane potential hits a threshold. There have been research outlining explicit parallels between the two [36, 80]. For example, [8] showed that the classic drift-diffusion process can be mapped onto a highly interactive neural network with pooled inhibition, effectively linking a DDM to a recurrent spiking circuit. Similarly, attractor network model proposed by [86], a biophysically detailed spiking network with excitatory and inhibitory pools, was shown to instantiate evidence accumulation to a threshold, producing decision behavior well-described by a DDM.

In summary, while DDMs have indeed been implemented within spiking neural networks, these efforts have primarily focused on biologically plausible models intended for simulation and neurophysiological interpretation, rather than deep learning oriented architectures [13, 80, 83, 84, 86, 89]. These classical approaches prioritize biological realism over computational efficiency or task performance. By contrast, applying DDM principles to TTFS SNNs remains unexplored. Thus, investigating whether TTFS networks naturally exhibit drift-diffusion-like behavior, and whether decision latency scales with sample difficulty under this formalism, could yield valuable insights.

7.2. Sample Difficulty and Evidence

We need a principled way to measure evidence directly from individual samples, since our central hypothesis is that low evidence corresponds to high difficulty. In typical supervised classification, margin-based metrics (see chapter 4) offer exactly this: the (signed) margin is defined as the distance of a sample to the decision boundary, which inherently should quantify how strongly the model supports the predicted class [14, 88, 94].

7.2.1. Margin Distance and Evidence Accumulation

In chapter 4, we describe how margin distance effectively captures sample difficulty. Here, we hypothesize that the signed margin not only reflects difficulty but also plays a role as evidence strength; a small or negative signed margin indicates that the model has weak or even conflicting evidence regarding which class the sample belongs to. Thus, by using signed margin as our metric, we obtain a unified metric that quantifies how much (or how little) evidence is available for the SNN to make a decision in the DDM framework.

In the DDM framework, as mentioned previously, drift rate v represents the average speed of evidence accumulation toward the correct decision boundary: high drift rates correspond to strong, clear evidence, and low drift rates to less discernible signals [52]. We can therefore draw a formal link: the signed margin of a sample maps to v, such that larger margins produce higher drift rates, while small or negative margins yield low–or even reverse–drift. Thus, as margin increases, the DDM predicts faster and more reliable decisions. Conversely, as margin decreases, or becomes negative, evidence is weak or misleading, and the model forecasts slower responses and higher error rates.

Moreover, the DDM also predicts that on trials resulting in an incorrect decision, the response time will tend to be longer than on correct trials. This arises from across-trial variability in the drift rate, equivalently, variability in per-sample evidence, which increases the likelihood of slow boundary crossings when the instantaneous drift is weak or even in the “wrong” direction [60, 62, 63]. In our TTFS framework, this directly corresponds to per-sample variability in the sample margin: samples with near zero or negative margins not only produce slower mean latencies but, when they do lead to a choice, tend to produce even longer first-spike times on error trials.

7.3. The SepDots Synthetic Classification Task

In this section, we introduce a simple, analytically tractable binary classification task “Separating the Dots” (or SepDots for short), inspired by Two-Alternative Forced Choice (2AFC) methods to measure the sensitivity of response times in humans or animals with respect to input stimuli. This experiment allows us to compute exact margins and directly relate them to time-to-first-spike of spiking neural networks.

7.3.1. Problem Definition

Let $\mathbf{x} = [x_1, x_2]^T \in \mathbb{R}^2$. We define two classes, each a bivariate normal distribution with identical, isotropic covariance:

$$\begin{aligned} \text{Class 1}: &\quad \mathbf{x} \sim N\big([-\mu, -\mu]^T, \sigma^2\mathbf{I}\big), \\ \text{Class 2}: &\quad \mathbf{x} \sim N\big([+\mu, +\mu]^T, \sigma^2\mathbf{I}\big), \end{aligned}$$

where $\mu > 0$ controls class separation and $\sigma^2 = 0.05$ is fixed. The optimal linear decision boundary between the two classes is

$$x_1 + x_2 = 0, \tag{7.1}$$

so that the signed margin (euclidean distance to the decision boundary) of any sample $\mathbf{x}$ is

$$m(\mathbf{x}) = \frac{x_1 + x_2}{\sqrt{2}}. \tag{7.2}$$

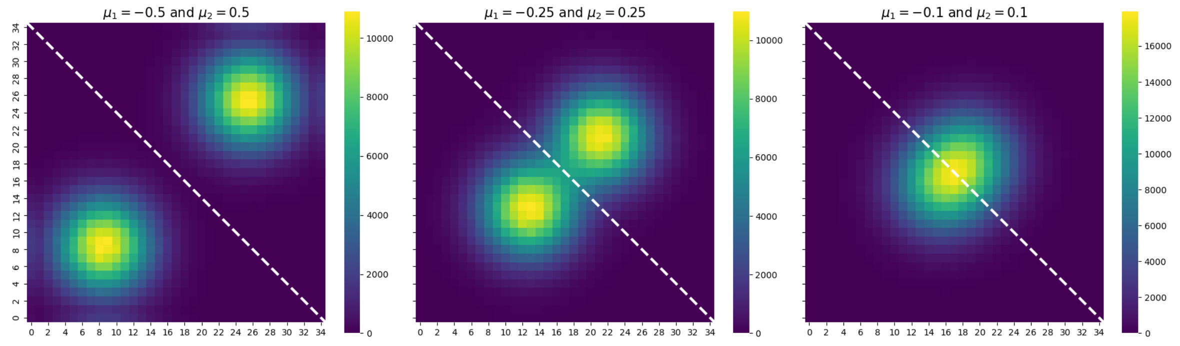


Figure 7.2: Heatmaps of the two-class bivariate normal distributions for three values of μ: (left) $\mu_1 = -0.5,\ \mu_2 = +0.5$, (center) $\mu_1 = -0.25,\ \mu_2 = +0.25$, (right) $\mu_1 = -0.1,\ \mu_2 = +0.1$. The dashed white line indicates the optimal decision boundary $x_1 + x_2 = 0$.

The separation between the two classes under different values of μ is visualized in Figure 7.2. As μ decreases, the two Gaussians overlap more heavily, increasing the Bayes error.

7.3.2. Mapping to TTFS SNN Inputs

To present SepDots samples to a TTFS SNN, we proceed as follows:

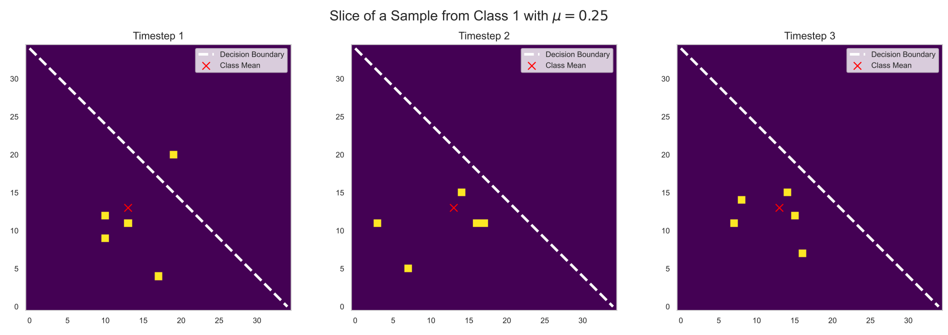
1. **Discrete Timesteps.** We simulate $T = 20$ time bins. At each timestep $t = 1, \ldots, T$, we draw $K = 5$ i.i.d. samples $\{\mathbf{x}_k^{(t)}\}_{k=1}^K$ from the true class distribution (each class chosen with probability $1/2$).
2. **Spatial Encoding.** Each $\mathbf{x} \in \mathbb{R}^2$ is first clipped to $[-1, 1]$ and then mapped onto a 35×35 binary image, such that $[0, 0]$ corresponds to the center of the image. The continuous coordinates $\mathbf{x}$ are linearly scaled and rounded to pixel indices:

$$i = \operatorname{round}(17 \cdot (x_1 + 1)), \quad j = \operatorname{round}(17 \cdot (x_2 + 1)),$$

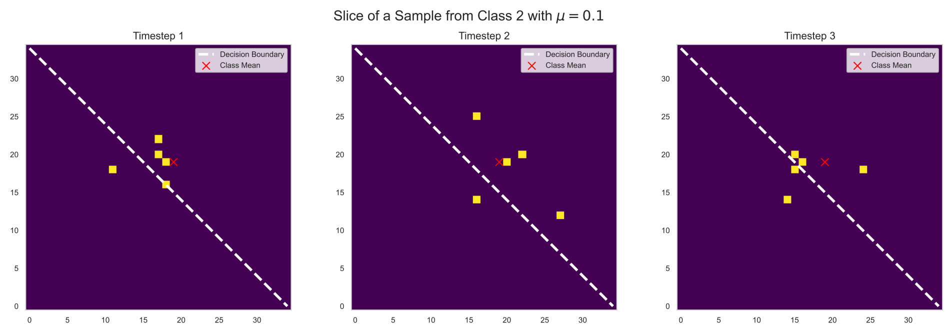
   and pixel (i, j) is set to 1.
3. **Aggregation.** The K dots at time t form the input image $\mathbf{I}^{(t)}$. This sequence $\{\mathbf{I}^{(t)}\}_{t=1}^T$ is streamed into the SNN.

To illustrate how individual samples look on the 35×35 grid over the first few timesteps, we show in Figure 7.3b a trial from Class 2 at $\mu = 0.1$, and in Figure 7.3a a trial from Class 1 at $\mu = 0.25$. The red cross marks the true class mean projection onto pixel-space, and the dashed line is again $x_1 + x_2 = 0$.

As we can see in Figure 7.3b, even for moderate separation ($\mu = 0.1$) the instantaneous pattern of dots may lie close to the decision boundary, yielding small signed margins; in contrast, the example in Figure 7.3a shows clearer separation and thus larger margins. These visualizations motivate our later analysis of how cumulative margin $M^{(t^*)}$ at the first-spike time correlates with TTFS latency.



(a) Class 1 sample ($\mu = 0.25$)



(b) Class 2 sample ($\mu = 0.1$)

Figure 7.3: Three-timestep slices of two SepDots trials. Each yellow square is one of the $K = 5$ dots at that timestep and the red cross is the true class mean.

7.3.3. Exact Margin Tracking

Because we know each $\mathbf{x}_k^{(t)}$ exactly, we first compute its raw signed margin $m(\mathbf{x}_k^{(t)})$. However, samples from Class 1 (mean $[-\mu, -\mu]$) produce negative margins by construction. To measure distance on the *correct* side of the decision boundary regardless of class label, we introduce

$$y = \begin{cases} +1, & \text{if sample from Class 2,} \\ -1, & \text{if sample from Class 1,} \end{cases} \qquad \tilde{m}(\mathbf{x}) = y\, m(\mathbf{x})\,. \tag{7.3}$$

Then at each timestep t with K samples, we define:

- *Instantaneous margin* at timestep t:

$$\tilde{m}^{(t)} = \frac{1}{K} \sum_{k=1}^{K} \tilde{m}\left(\mathbf{x}_k^{(t)}\right). \tag{7.4}$$

- *Cumulative margin* up to timestep t:

$$\widetilde{M}^{(t)} = \frac{1}{t} \sum_{\tau=1}^{t} \tilde{m}^{(\tau)}. \tag{7.5}$$

When the network emits its first output spike at time t^*, we record the cumulative margin $\widetilde{M}^{(t^*)}$ at the spike time, as the total evidence available at decision time.

7.3.4. Predictions

Under the DDM analogy, we hypothesize:

- **Class-correct margin drives speed:** Larger μ (greater class separation) leads to stronger evidence $\tilde{m}^{(t)}$ and higher average drift v, hence earlier first-spike times t^*.
- **Absolute margin governs latency regardless of correctness:** Because the network integrates magnitude of evidence, a large negative margin also produces strong drift. Thus $\left|\widetilde{M}^{(t)}\right|$ should inversely correlate with spike time even for incorrectly-classified trials, leading to fast but wrong decisions when $\tilde{m}(\mathbf{x})$ was large in magnitude but on the wrong side.
- **Incorrect outputs are will be slower:** DDM predicts that trials ending in error will on average have longer response times, due to across-trial variability in drift rate. In our TTFS context, this implies that on samples where the network misclassify, the first-spike latency should be longer than on correct trials, reflecting slower evidence accumulation in those cases.

7.4. Impact of Sample Difficulty on First-Spike Latency

One of our primary research questions in this chapter was whether time-to-first-spike latency in a TTFS SNN systematically increases with sample difficulty, as defined in Chapter 4. As we have argued under the DDM analogy, smaller signed margins should correspond to lower drift rates and therefore to slower evidence accumulation. To empirically test this prediction, we designed the SepDots task to manipulate margin-based difficulty and directly measure first-spike times. According to the DDM framework, reductions in per-sample margins will increase the expected first-hitting time due to decreased drift rates. Hence, the hypothesis that latency should increase with sample difficulty is supported with our DDM framework.

Effect of Sample Difficulty on the Time-to-First-Spike of Spiking Neural Networks

Eren Aydoslu
TU Delft

Abstract

Spiking neural networks (SNNs) with Time-to-First-Spike (TTFS) coding promise rapid, sparse, and energy-efficient inference. However, the impact of sample difficulty on TTFS dynamics remains underexplored. We investigate (i) how input hardness influences first-spike timing and (ii) whether training on hard samples expedites inference. By quantifying difficulty via geometric margins and Gaussian-noise perturbations, and modeling leaky integrate-and-fire dynamics as Gaussian random walks, we derive first-hitting-time predictions. We further show that training-time noise, akin to ridge regularization, reduces weight variance and increases expected spike latencies. Empirical results on a synthetic task, MNIST, NMNIST, and CIFAR-10 with spiking MLPs/CNNs confirm that harder inputs slow inference and noise-trained models trade robustness for latency. Our findings align TTFS behavior with drift-diffusion models and provide a framework for balancing speed and robustness in neuromorphic SNNs.

1. Introduction

Spiking neural networks (SNNs), inspired by biological neural systems, encode information using spike timings rather than continuous activation values. Among various neural coding schemes, Time-to-First-Spike (TTFS) coding has garnered attention for its efficiency and biological plausibility. TTFS-based SNNs inherently prioritize rapid, sparse, and energy-efficient computation, making them particularly appealing for low-power and real-time applications [10].

However, the performance and characteristics of TTFS-based SNNs under conditions of varying sample difficulty remain under-explored. Specifically, understanding how the time-to-first-spike behavior of latency-encoded SNNs changes as samples become intrinsically more challenging is crucial for both theoretical insights and practical applications. In classical machine learning, sample difficulty is often associated with ambiguity in the data distribution and proximity to decision boundaries [8, 41]. Inspired by this, our research investigates the following central questions:

1. What happens to the TTFS behavior of latency-encoded SNNs as samples become increasingly difficult?
2. Can we use harder samples during training to make inference faster?

We approach this question by considering sample difficulty through a proxy measure, additive Gaussian noise. Drawing parallels from statistical learning theory and ridge regression equivalence in linear models, we hypothesize that introducing Gaussian noise to training samples reduces the variance of the learned weights. Under this assumption, we model neuronal membrane potentials as Gaussian random walks, enabling analysis through the lens of first-hitting-time stochastic processes. We believe that reduced weight variance, induced by noisy training, leads to higher expected first-spike times, thus potentially decelerating inference.

To validate our hypothesis, we conducted extensive experiments utilizing spiking multi-layer perceptrons (MLPs) and convolutional neural networks (CNNs) using a synthetic dataset and across widely-used datasets, MNIST, NMNIST, and CIFAR-10, examining network responses under various noise strength. Through empirical analyses, we measure and explain how first-spike latency evolves relative to sample difficulty, thereby advancing the understanding of latency coding dynamics in spiking neural networks.

In summary, we make the following contributions:

1. We introduce a principled analytical framework that models TTFS dynamics under varying sample difficulty by treating membrane-potential accumulation as a Gaussian random walk with decay/mean-reversion and applying first-hitting-time theory.
2. We establish a theoretical link between training-time Gaussian noise (akin to parameter regularization) and increased first-spike latencies via reduced synaptic-weight variance.
3. We empirically validate our theory on both a synthetic SepDots task and three standard benchmarks (MNIST,

NMNIST, CIFAR-10) using spiking MLPs and CNNs, showing that harder inputs systematically slow inference and that noise training trades robustness for latency.

4. We demonstrate that TTFS latency under uncertainty aligns with classical drift-diffusion models.

2. Related Work

2.1. Sample Difficulty Metrics

In related literature, many different principal approaches have been proposed to quantify the intrinsic "difficulty" of individual samples. However, for the scope of this research, we are only interested in two: (i) geometric margin measures relative to a decision boundary, and (ii) synthetic hardness induced by Gaussian-noise perturbations. These metrics have been successfully applied in supervised learning, active learning and robustness studies, but have not yet been systematically linked to latency behavior in TTFS-coded SNNs.

2.1.1. Margin-Based Difficulty

A classical measure of sample difficulty is the signed distance (margin) from the sample to the optimal decision boundary. In support vector machines, maximizing the minimum margin yields better generalization, and samples with small margins are those most susceptible to misclassification under slight model perturbations [9, 43]. Varshney et al. [45] showed that margin directly captures both geometric resilience and Bayes error: points near the boundary lie in regions where class-conditional distributions overlap, incurring high irreducible error. More recent work has extended margin concepts to deep networks, demonstrating that margin distributions correlate with per-sample uncertainty and generalization gaps [1, 26]

2.1.2. Gaussian Noise as a Proxy for Difficulty

When the true decision boundary is unknown or intractable, hardness can be induced by adding isotropic Gaussian noise to inputs. The manifold hypothesis posits that high-dimensional data (e.g., images) lie near much lower-dimensional manifolds, each encoding semantic factors such as object identity or pose [12]. In modern vision embedding spaces, whether derived from contrastive models like CLIP or from GAN latent representations, individual semantic concepts (e.g., "cat"-ness, orientation, age) align with specific low-dimensional directions on these manifolds [22, 33]. Because isotropic Gaussian noise perturbs all coordinates equally at random, it is very unlikely to produce a change that aligns with a particular semantic direction (with probability near zero in high dimensions) and thus very unlikely to increase a sample's semantic class alignment or "margin" relative to its true manifold [2, 19]. In other words, to give an example, adding noise to an image of a cat is not very likely to make the image more cat-like. Instead, noise almost invariably pushes samples off their native manifolds toward regions of higher class overlap, increasing conditional uncertainty and Bayes error [13, 20]. Empirical studies confirm that moderate Gaussian perturbations can improve generalization by discouraging overfitting, whereas large noise amplitudes reliably degrade accuracy by elevating sample hardness [6, 21, 31].

2.2. Sample Difficulty and SNNs

Drift Diffusion Models (DDMs) describe the decision process as the accumulation of noisy evidence to one of two decision thresholds (see Figure 1), successfully capturing response-time distributions and choice accuracies across tasks and species [34, 35]. In their simplest form, DDMs introduce a one-dimensional decision variable $X(t)$ that evolves according to

$$dX = v\,dt + \sigma\,dW(t), \tag{1}$$

where v is the drift rate (mean evidence per unit time), σ the diffusion coefficient (noise magnitude), and $W(t)$ a standard Wiener process; choice and response time correspond to the first-passage time of $X(t)$ to one of two absorbing boundaries at $\pm a$ [35]. This framework naturally accounts for both the mean and variability of reaction times as well as speed–accuracy trade-offs: higher v yields faster, more consistent responses, while lower v produces slower, more variable decisions with increased error rates [3].

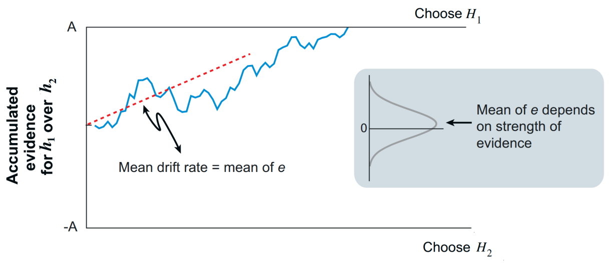

Choose H_1
A
Accumulated evidence for h_1 over h_2
Mean drift rate = mean of e
0
Mean of e depends on strength of evidence
-A
Choose H_2

Figure 1. Schematic of the drift diffusion model. The decision variable accumulates noisy momentary evidence $e(t)$ over time with mean drift rate μ (red dashed line), until it reaches the upper boundary $+A$ (choose H_1) or lower boundary $-A$ (choose H_2). The side-panel plot illustrates the probability density of momentary evidence e, whose mean shifts according to stimulus strength. Emphasizing how strong evidence leads to faster decisions. Figure taken from [18].

Although DDMs provide an elegant account of behavior, their abstract variables lack a clear mapping onto biophysically realistic neurons, motivating neural-level implementations that can measure diffusion dynamics in spiking activity. Analyses have directly linked neuron firing rates to accumulator processes in perceptual decisions, motivating neural-level diffusion analogues of DDM [18, 27, 29, 32].

In computational neuroscience, spiking neural network implementations have long been proposed as neural substrates for evidence accumulation, with early work demonstrating that interconnected pools of excitatory and inhibitory neurons can instantiate drift-like dynamics mapped onto diffusion model parameters [3, 47]. For instance, the biophysically detailed model by Wang (2002) [46] and its extension by Wong and Wang (2006) [47] showed that recurrent spiking circuits could replicate behavioral data by varying input "drift rate" and synaptic parameters corresponding to decision thresholds. Subsequent analyses mapped manipulations of spiking circuit parameters, such as input sensitivity, background excitation, and recurrent connectivity, to drift rate, boundary separation, and non-decision time in the diffusion model, elucidating concrete neural implementations of cognitive-level variables [44]. Additionally, spiking decision-making models with learning rules have been proposed to bridge cognitive models and SNNs, showing that accumulation can emerge from trainable spiking networks [23, 28].

However, these efforts have primarily focused on biological plausibility and parameter-mapping, often overlooking task-performance and latency effects under varying sample difficulties in deep learning oriented TTFS SNN architectures [3, 28, 47]. To date, the impact of sample-specific difficulty metrics, such as margin distance or noise-induced hardness, on first-spike latency within a drift-diffusion framework in TTFS SNNs has not been systematically addressed. This gap underscores the need to integrate sample difficulty measures into evidence-accumulation analyses of TTFS networks to predict and control inference latency under uncertainty.

In the DDM framework, the impact of sample difficulty on decision times is captured by the drift rate parameter v. Harder samples, such as stimuli with lower discriminability or higher noise, yield smaller drift rates, resulting in longer times for the decision variable to reach a boundary and thus slower and more variable response times (RT) [3, 35]. Across diverse perceptual tasks, formal fittings of the DDM consistently show that difficulty-induced reductions in drift rate account for observed changes in RT distributions under ambiguous or noisy conditions [27, 37]. Classic random-dot motion tasks illustrate this relationship: as motion coherence decreases (i.e., the stimuli becomes harder), drift rate diminishes, producing increased mean reaction times and heavier right tails in RT distributions [30, 34, 37]. Consequently, when mapping sample difficulty metrics to TTFS-coded SNNs, higher difficulty should correspond to lower effective drift rates in a DDM interpretation, predicting systematically later first-spike latencies for harder samples.

Moreover, the DDM also predicts that trials ending in an incorrect choice tend to have longer response times than correct trials. This follows from across-trial variability in drift rate: high drift rates produce fast, accurate boundary crossings, whereas low drift rates both increase error probability and slow the accumulation process, leading to a distribution of missclassified response times that is shifted toward longer latencies [35, 38]. In our TTFS context, this implies that samples with particularly small or negative margins, not only accumulate evidence more slowly on average, but when they do lead to misclassification, will evoke the longest first-spike latencies.

2.3. Spiking Neuron as a Stochastic Process

2.3.1. Membrane Potential as a Gaussian Random Walk

The sub-threshold membrane potential of leaky integrate-and-fire neurons has long been modeled as a stochastic process whose increments converge to a Gaussian distribution under the Central Limit Theorem (CLT) when driven by many weak synaptic inputs [14, 16]. Gerstein and Mandelbrot [17] first employed an inverse-Gaussian framework to describe first-passage problems in neural spiking, demonstrating that membrane dynamics under constant drift and Gaussian noise can be treated as a random walk with drift. Later works formalized the discrete-time analogue, showing that if (i) the number of presynaptic inputs is large, (ii) synaptic weights have finite variance, and (iii) presynaptic spike trains are weakly correlated, then the membrane potential sum approaches Gaussian [4, 25]. These analyses, however, predominantly consider continuous-time or asymptotic regimes and do not fully characterize the influence of finite-step effects or input "difficulty" in Time-to-First-Spike coding schemes.

2.3.2. First Hitting Time Models for Time-to-First-Spike

First-hitting time (or first-passage time) models have been extensively used to predict spike latencies in stochastic neuron models. Classic results by Siegert derived the inverse-Gaussian density for barrier crossing times in diffusion approximations of integrate-and-fire neurons [39, 40]. Chang and Peres [7] later provided rigorous bounds showing that discrete-time Gaussian random walks converge to the inverse-Gaussian limit as the time step vanishes. More recent numerical methods have improved the estimation of first-passage time densities for Ornstein–Uhlenbeck neuron models [5, 42]. Despite these foundational contributions, existing work largely addresses homogeneous synaptic inputs and continuous diffusion models; the specific effects of discrete timesteps, synaptic weight variability, sample and difficulty, remain underexplored in the context of TTFS SNNs architectures.

3. Approach

In this section, we detail the methodologies and key theoretical formulations employed in our research.

$$t_{spike}(x,y,c) = \begin{cases} \text{round}((1 - I(x,y,c)) \times T_{max}), & \text{if } I(x,y,c) > \epsilon \\ \text{no spike}, & \text{otherwise} \end{cases} \tag{2}$$

3.1. Leaky Integrate-and-Fire Neuron Model

We use the discrete-time Leaky Integrate-and-Fire (LIF) neuron model, characterized by the following update equation:

$$V_i[t+1] = \beta V_i[t] + \sum_j W_{ij} S_j[t] - V_{th} S_i[t] \tag{3}$$

Here, $V_i[t]$ denotes the membrane potential of neuron i at discrete timestep t, $\beta \in [0,1]$ is the leak parameter controlling the decay of the potential over time, W_{ij} represents the synaptic weight between neuron j (presynaptic) and neuron i (postsynaptic), $S_j[t]$ indicates whether neuron j emitted a spike at timestep t, and V_{th} is the threshold potential. If $V_i[t]$ surpasses V_{th}, neuron i emits a spike ($S_i[t] = 1$) and its membrane potential resets accordingly.

3.2. Latency Encoding of Inputs

In our implementation, latency encoding converts pixel intensities from input images into spike timings. Each pixel value, initially in the range $[0,1]$, is transformed into a spike timing such that pixels with higher intensities (closer to 1) spike earlier, and pixels with lower intensities spike later. Formally, this is described in Equation 2, where $I(x,y,c)$ represents the intensity of the pixel at location (x,y) in channel c, T_{max} denotes the maximum number of timesteps, and ϵ is a clipping threshold below which pixel intensities do not generate spikes.

3.3. Temporal Mean Squared Error Loss

To effectively train our latency-encoded SNN, we utilize a temporal mean squared error (MSE) loss function defined over spike timings [10]. Consider an output layer of neurons, each producing spike times represented as normalized timings within $[0,1]$. Given example spike times $[t_1, t_2, t_3]$ for three neurons, normalized by dividing each by the total number of timesteps T_{max} (e.g., for spike times $[1,2,3]$ and $T_{max} = 5$, the normalized times are $[0.2, 0.4, 0.6]$), the temporal MSE loss is computed as follows:

$$L = \frac{1}{N} \sum_{i=1}^{N} (t_i - y_i)^2 \tag{4}$$

where t_i is the normalized spike time of neuron i, y_i is the desired normalized target time (0 for the correct class, meaning spike as early as possible, and 1 for incorrect classes, meaning spike as late as possible or ideally never), and N is the total number of output neurons. This approach encourages the network to minimize the latency of correct class spikes while delaying or inhibiting incorrect class spikes.

One significant challenge in training with the temporal MSE loss arises from the fundamental discontinuity in spike timing mechanisms. Since the precise moment a neuron fires depends on a threshold-crossing event, the derivative of spike timing with respect to membrane potential is mathematically undefined. To circumvent this issue, we employ a commonly used custom gradient approximation during backpropagation [10]. Specifically, we set:

$$\frac{\partial t_{spike}}{\partial U} = -1 \tag{5}$$

This sign estimator establishes the directional relationship that increasing membrane potential leads to earlier firing times.

3.4. Surrogate Gradients and Backpropagation Through Time

Training SNNs presents a fundamental challenge: the spike generation mechanism is inherently non-differentiable, creating a discontinuity in the gradient flow. To address this, we employ surrogate gradients that approximate the derivative of the spiking function. For the forward pass, we use the Heaviside step function:

$$S = \begin{cases} 1 & \text{if } U \geq U_{thr} \\ 0 & \text{if } U < U_{thr} \end{cases} \tag{6}$$

For the backward pass, we utilize the gradient of a shifted arc-tangent function as our surrogate:

$$S \approx \frac{1}{\pi} \arctan(\pi U \frac{\alpha}{2}) \tag{7}$$

$$\frac{\partial S}{\partial U} = \frac{1}{\pi} \frac{1}{(1 + (\pi U \frac{\alpha}{2})^2)} \tag{8}$$

where α is a hyperparameter controlling the smoothness of approximation [11]. Additionally, since our network processes information across multiple timesteps, we use Backpropagation Through Time (BPTT), which unfolds the SNN temporally and applies backpropagation across the resulting computational graph. This allows gradients to flow backward through both spatial connections and temporal dynamics, enabling end-to-end training while preserving the temporal characteristics essential to latency-encoded networks.

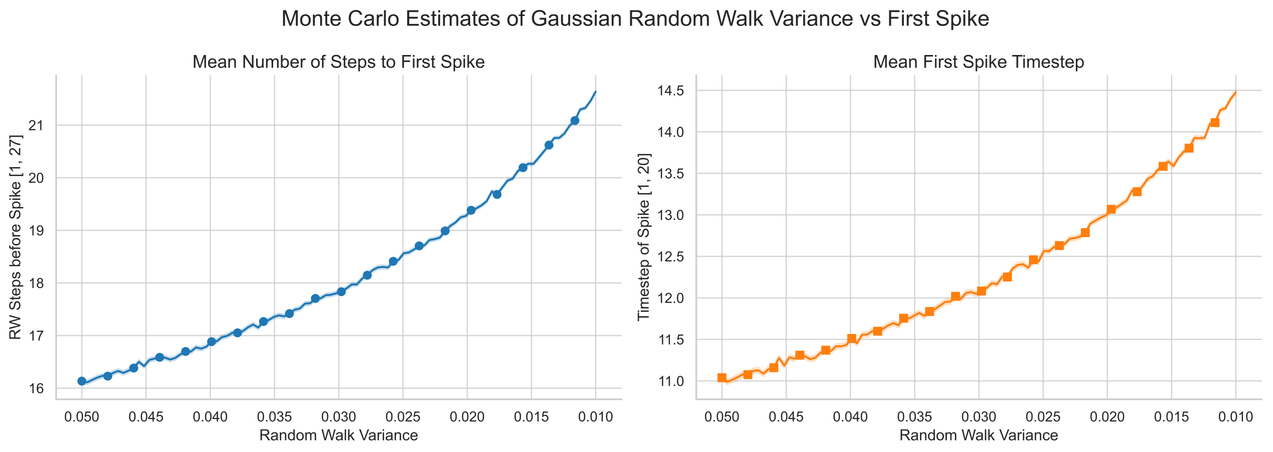


Figure 2. Monte Carlo ($N = 100\,000$) estimates of (a) mean number of steps to first spike and (b) mean first-spike timestep as functions of synaptic variance σ^2. Decreasing σ^2 leads to slower accumulation relative to leak, increasing the expected latency. N.B., the x-axis is inverted.

3.5. Discrete-Time Spiking Neuron Model and Variance Effects

We model each neuron as a non-leaky integrate-and-fire unit operating in discrete timesteps. Let S_t denote the membrane potential at timestep t. Assuming each presynaptic spike contributes a weight $w_i \in \{0,1\} \cdot \mathcal{N}(\mu, \sigma^2)$, the subthreshold dynamics form a Gaussian random walk:

$$S_t = S_{t-1} + \sum_{i=1}^{n_t} w_i \tag{9}$$

with $S_0 = 0$ and threshold $V_{\text{th}} = 1$. Under the i.i.d. Gaussian weight assumption, the first-hitting time T to reach V_{th} admits a continuous-time approximation via a Wiener process with drift $\mu > 0$ and diffusion σ^2, whose hitting-time density is inverse Gaussian [15, 24]:

$$f_T(t) = \frac{1}{\sqrt{2\pi\sigma^2 t^3}} \exp\Big(-\tfrac{(1-\mu t)^2}{2\sigma^2 t}\Big), \tag{10}$$

$$\mathbb{E}[T] = \frac{1}{\mu}, \quad \text{Var}(T) = \frac{\sigma^2}{\mu^3} \tag{11}$$

To capture realistic membrane leakage and finite input spikes, we introduce a decay factor $\beta \in (0, 1]$ and bin inputs into T discrete steps:

$$S_t = \beta\, S_{t-1} + \sum_{i=1}^{n_t} w_i, \qquad \sum_{t=1}^{T} n_t = N_{\max} \le 27. \tag{12}$$

Monte Carlo simulations of this leaky random walk (with $\beta = 0.95$, varying σ^2, up to $N_{\max} = 27$ for a 3×3 RGB receptive field in a convolutional layer) reveal that decreasing synaptic-weight variance shifts the first-spike distribution to the right, i.e., lower σ^2 yields heavier tails and longer mean latencies (see Figure 2).

Quantitatively, both the expected number of steps to threshold and the mean first-spike timestep increase monotonically as σ^2 decreases, confirming that reduced weight variability slows evidence accumulation relative to leak, thereby prolonging time-to-first-spike.

3.6. SepDots Synthetic Classification Task

SepDots (short for separating the dots) is a synthetic binary classification dataset designed to provide analytical control over sample difficulty via tunable class separation, enabling exact margin computation and direct investigation of first-spike latency under varying difficulty.

3.6.1. Problem Definition

$$\mathbf{x} = \begin{cases} \mathcal{N}\big([-\mu, -\mu]^T,\ \sigma^2 I\big), & \text{Class 1,} \\ \mathcal{N}\big([+\mu, +\mu]^T,\ \sigma^2 I\big), & \text{Class 2,} \end{cases} \tag{13}$$

where $\mu > 0$ controls class separation (and thus difficulty) and $\sigma^2 = 0.05$ is fixed. We simulate 20 discrete timesteps and at each timestep we draw $K = 5$ i.i.d. samples from the true class distribution (see Figure 3), where each class is chosen with probability $1/2$. Finally, sampled values are mapped to a binary 35×35 image in a linear fashion, such that [0, 0] corresponds to the center of the image.

3.6.2. Margin Computation

The optimal linear decision boundary is

$$x_1 + x_2 = 0\,. \tag{14}$$

Then, we can calculate the signed margin of a sample $\mathbf{x}$ as

$$m(\mathbf{x}) = \frac{x_1 + x_2}{\sqrt{2}}\,. \tag{15}$$

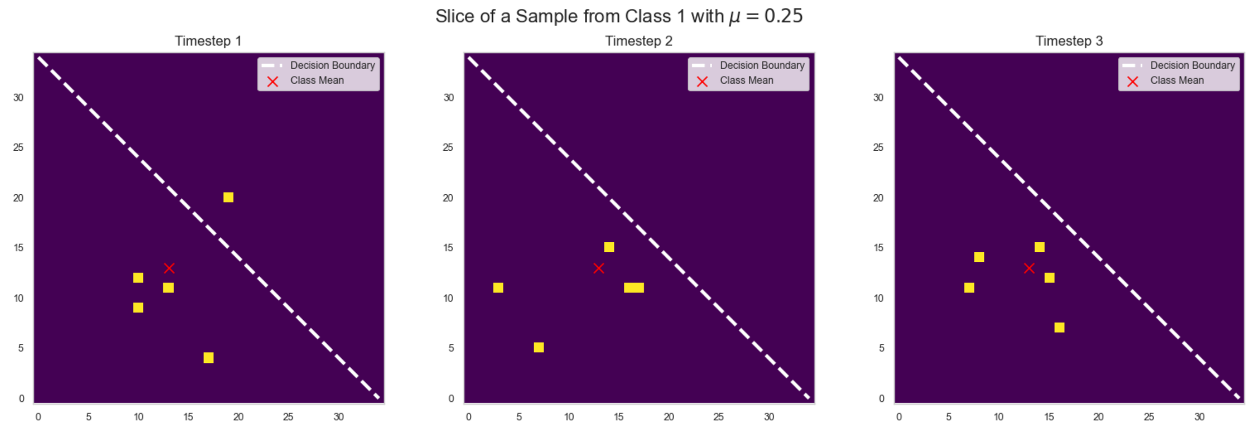


Figure 3. Three-timestep slices of a SepDots sample. Each yellow square is one of the $K = 5$ dots at that timestep and the red cross is the true class mean.

Dataset	Type	Levels
SepDots	Class mean, μ	$\{0.01, 0.025, 0.05, 0.075, 0.1, 0.15, 0.25\}$
MNIST	Gaussian noise, σ	$\{0, 0.025, 0.05, 0.075, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4\}$
NMNIST	Pixel-flip, p_{flip}	$\{0, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1\}$
CIFAR-10	Gaussian noise, σ	$\{0, 0.025, 0.05, 0.075, 0.1, 0.15, 0.2, 0.25\}$
CIFAR-10	Gaussian blur (33×33 kernel), σ	$\{0, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2\}$

Table 1. Perturbation parameters used to induce sample difficulty.

Let the true label be

$$y = \begin{cases} +1, & \mathbf{x} \text{ from Class } 2, \\ -1, & \mathbf{x} \text{ from Class } 1, \end{cases} \qquad (16)$$

so that the class-corrected margin is

$$\tilde{m}(\mathbf{x}) = y\, m(\mathbf{x})\,. \qquad (17)$$

3.6.3. Temporal Margin Aggregation

At each discrete timestep t, we draw K i.i.d. samples $\{\mathbf{x}_k^{(t)}\}_{k=1}^K$. We then compute:

$$\tilde{m}^{(t)} = \frac{1}{K}\sum_{k=1}^{K} \tilde{m}(\mathbf{x}_k^{(t)}) \quad \text{(instantaneous margin)}, \qquad (18)$$

$$\widetilde{M}^{(t)} = \frac{1}{t}\sum_{\tau=1}^{t} \tilde{m}^{(\tau)} \quad \text{(cumulative margin)}. \qquad (19)$$

These exact margin metrics allow us to quantitatively link sample difficulty, via $\tilde{m}$ and $\widetilde{M}$.

The key advantage of SepDots is its tunable difficulty via μ: as μ decreases, class overlap increases and the Bayes error rises, offering a fine-grained control over sample hardness. This synthetic task thus provides a clear testbed for linking evidence-accumulation predictions under the Drift Diffusion Model to actual TTFS latency under varying difficulty levels.

3.7. Experiments

We evaluate our theoretical predictions on four datasets, SepDots, MNIST, NMNIST, and CIFAR-10, by systematically varying both testing and training sample hardness. Furthermore, we repeat each experiment 30 times to increase the reliability of the results. Our two primary hypotheses are:

1. Increasing *testing* hardness will slow inference (longer first-spike latencies) due to reduced effective drift (weaker evidence accumulation).
2. Increasing *training* hardness will also slow inference, because noise-driven reductions in learned synaptic-weight variance will decrease membrane potential threshold-crossing speed.

3.7.1. SepDots

Hardness is controlled by the class-separation parameter μ, which directly modulates Gaussian overlap and margins. We train and test networks under multiple separation settings, observing how reduced μ (smaller margins) affects first-spike latency.

3.7.2. MNIST, NMNIST, CIFAR-10

For MNIST and CIFAR-10, hardness is induced by adding isotropic Gaussian noise or (only for CIFAR-10) Gaussian blur to each image prior to training and/or evaluation. For event-based NMNIST, hardness is induced by flipping each pixel with a fixed probability. However, we don't want to augment the data. Therefore, we sample each noise or blur pattern once before training and use the same perturbed dataset both during training. To keep overall spike-count statistics constant, we histogram-match the noisy inputs to their clean counterparts.

Additionally, for CIFAR-10 we use a small CNN (CNN-S) with $\approx$ 86k parameters and a large CNN (CNN-L) with $\approx$ 20M parameters.

3.7.3. Training and Evaluation Regime

The perturbation parameters we use to induce hardness in each dataset are defined in Table 1.

1. For each dataset and training hardness setting, train a TTFS-coded SNN from scratch.
2. For each trained model, evaluate on each testing hardness setting, recording first-spike latencies across output neurons.
3. Analyze the mean and variance of first-spike times as functions of training and testing hardness.

This process should allow us to separate the effects of *testing* difficulty, slower evidence accumulation, from those of *training* difficulty, changed synaptic weight statistics, on TTFS inference latency.

4. Results

Before we go into classic datasets, let's first analyze the behavior of the first spike times in our synthetic classification task SepDots.

4.1. Time-to-First-Spike Behavior

4.1.1. SepDots

Figure 4 reports the classification accuracy matrix as a function of the training-distribution mean μ_{train} (x–axis) and the testing-distribution mean μ_{test} (y–axis). Accuracy remains near 100% when $\mu_{\text{test}} \geq 0.1$ regardless of training hardness, but degrades sharply when testing separation falls below $\mu_{\text{test}} \approx 0.05$.

To probe the latency effects, we first examine the marginal relationship between cumulative margin at spike time and latency (Figure 5). Latency is longest for near-zero cumulative margins, where evidence is weakest, and decreases monotonically as $\widetilde{M}$ grows in magnitude. This brings prominence to the drift-diffusion prediction that higher net evidence yields faster first-spike times.

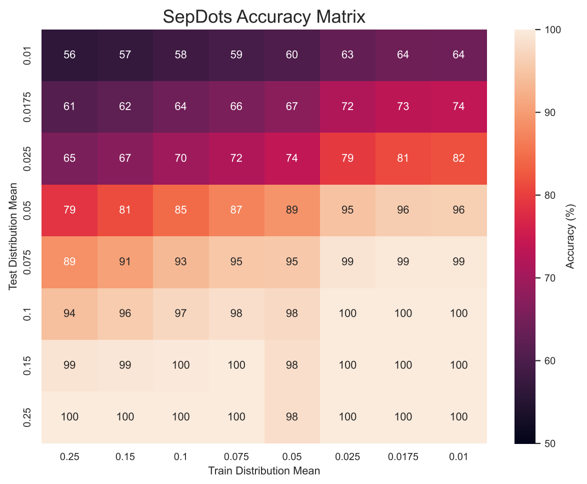


Figure 4. SepDots: Classification accuracy for varying train and test distribution means μ_{train} (x-axis) and μ_{test} (y-axis). High test separation ($\mu_{\text{test}} \geq 0.2$) yields near-perfect accuracy, while low separation degrades performance, especially when networks are trained on very hard (small-μ_{train}) data. From left-to-right harder training; from bottom-to-top harder testing.

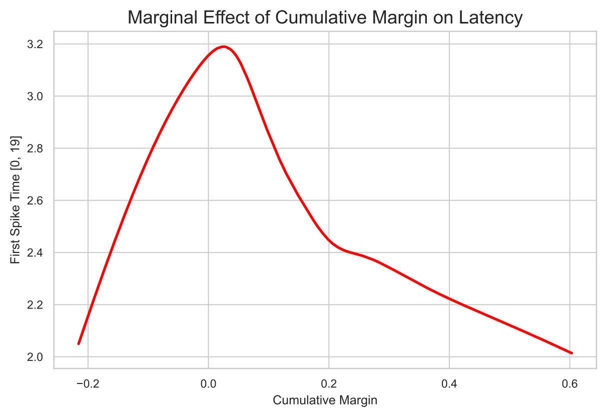
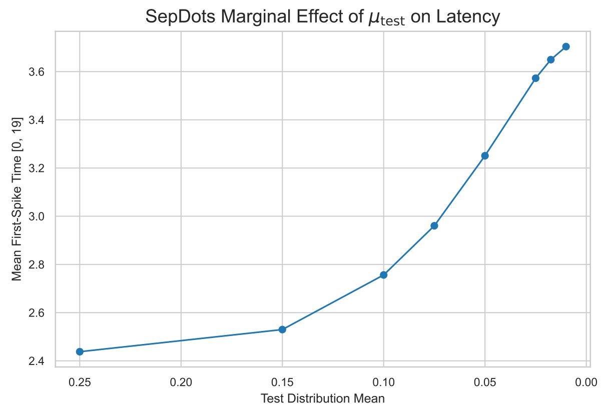
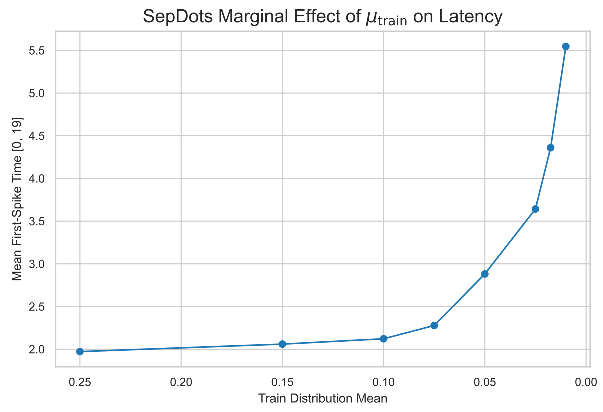


Figure 5. SepDots: Marginal effect of cumulative margin $\widetilde{M}$ at first-spike time on latency, estimated via LOWESS regression. Latency peaks near zero margin (weak evidence) and decreases as the absolute cumulative margin grows, confirming that stronger net evidence speeds first spikes.

Next, Figure 6a and Figure 6b show how mean first-spike time varies with μ_{test} and μ_{train}, respectively. Averaging out training effects, increasing μ_{test} reduces latency from $\approx$ 3.7 timesteps at $\mu_{\text{test}} = 0.01$ down to $\approx$ 2.4 at $\mu_{\text{test}} = 0.25$. Conversely, when averaging out testing hardness, raising μ_{train} speeds inference even more dramatically, from over 5.5 timesteps at $\mu_{\text{train}} = 0.01$ to just under 2.0 at $\mu_{\text{train}} = 0.25$. Thus both testing and training hardness could be independently controlling spike-latency via



(a) SepDots: Mean first-spike time versus test distribution mean μ_{test} (averaging over μ_{train}). Decreasing μ_{test} yields slower spikes. Networks tested on easier data exhibit faster inference. N.B. the x-axis is inverted to show the effect of going from easy to hard.



(b) SepDots: Mean first-spike time versus train distribution mean μ_{train} (averaging over μ_{test}). Networks trained on harder data exhibit slower inference. N.B. the x-axis is inverted to show the effect of going from easy to hard

Figure 6. Marginal effects of test-set and train-set distribution means on TTFS first-spike latency.

evidence-accumulation dynamics.

Finally, the combined heatmap in Fig. Figure 7 visualizes mean first-spike time across the full (train, test) grid. The lower-left corner (small $\mu_{\mathrm{train}}, \mu_{\mathrm{test}}$) exhibits the largest latencies, while the upper-right corner (large $\mu_{\mathrm{train}}, \mu_{\mathrm{test}}$) compresses latencies. The smooth gradient along both axes seems to provide evidence for our two hypotheses.

4.1.2. Qualitative Analysis

To motivate our quantitative analysis (which will come later in subsection 4.2) and better understand its implications, we visualize the results obtained on the NMNIST dataset. While our earlier qualitative study focused on the SepDots

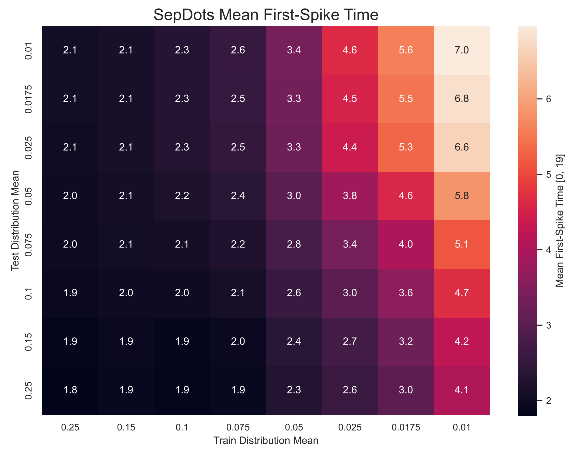


Figure 7. SepDots: Heatmap of mean first-spike time across the grid of (μ_{train}, μ_{test}) values. Latency increases smoothly from easy–easy settings in the bottom left to from hard–hard settings in the top-right, confirming both training and testing hardness effects.

dataset, we now extend this investigation to the classic NMNIST benchmark. In particular, we examine:

1. **Accuracy under noise:** how varying levels of additive input noise affect overall classification accuracy.
2. **Spiking behavior:** the temporal spike patterns produced during correctly and incorrectly classified samples as noise increases.
3. **Latency dynamics:** the impact of noise injected during training on the time-to-first-spike inference latency of the network.

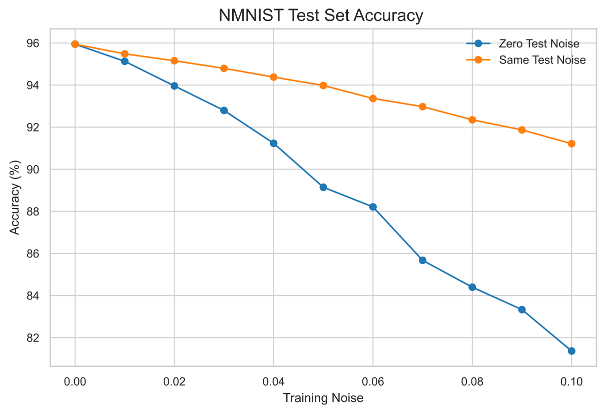


Figure 8. NMNIST test-set accuracy as a function of the training noise level. The blue curve shows performance when evaluated with zero test noise, while the orange curve shows performance when evaluated with the same noise level used during training.

Figure 8 illustrates how increasing the level of additive noise during training impacts classification accuracy on the

NMNIST test set. When evaluated without any test noise (blue line), accuracy degrades sharply as training noise increases, dropping from about 96% at zero noise to around 81% at a noise level of 0.1. By contrast, evaluating with matching test noise level with training noise level yields a more shallow decline, demonstrating that training with noise produces robustness to the similar perturbations at inference time.

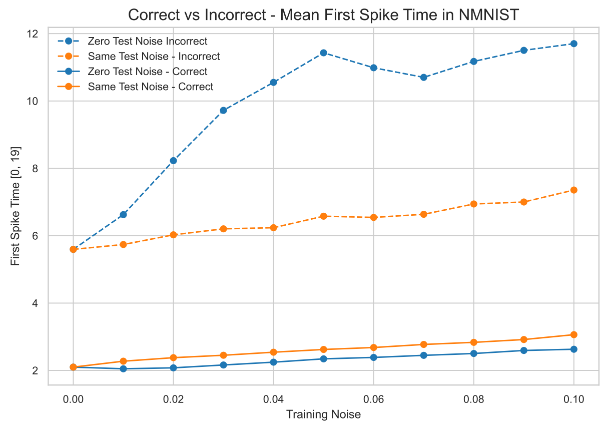


Figure 9. Mean time-to-first-spike for correctly and incorrectly classified NMNIST samples as a function of training noise. Solid lines denote correctly classified samples (blue: zero test noise; orange: matching test noise with training noise), while dashed lines denote incorrectly classified samples.

Figure 9 reveals a key insight: incorrectly classified samples (dashed lines) exhibit substantially longer TTFS compared to correctly classified ones (solid curves), indicating that misclassifications are associated with delayed spiking responses.

Figure 10 focuses on the TTFS of the correctly classified samples. In the figure, the TTFS under matching test noise (orange) consistently lies above that under zero test noise (blue), demonstrating that higher inference noise reliably increases latency. Moreover, for both test-noise settings, the mean TTFS increases monotonically with the training noise level, showing that increased training hardness prolongs output latency in all these cases.

4.2. Quantitative Analysis of Results

Prior to detailed latency analysis, we first examined how first-spike times varies between correct versus incorrect classifications (see Table 2). Since classification accuracy itself varied substantially with both training and testing noise, pooling all TTFS values would merge the effects correct vs incorrect spike times with sample difficulty. If we used all, the estimated parameters might be dominated by the change in accuracy rather than observing the real effect of training and testing noise. To isolate changes in latency

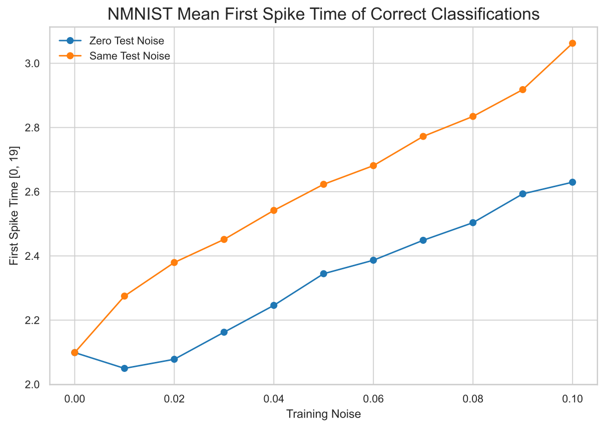


Figure 10. Mean time-to-first-spike of correctly classified NMNIST samples under zero test noise (blue) and matching test noise (orange) as a function of training noise. This figure is equivalent to Figure 9 except it's zoomed in on correct classifications

Dataset	Hardness	Model	Correct TTFS	Incorrect TTFS
SepDots	Distribution Overlap	CNN	3.04	3.47
MNIST	GWN	MLP	0.91	6.82
MNIST	GWN	CNN	2.61	8.24
NMNIST	Pixel Flip	CNN	2.75	8.39
CIFAR-10	GWN	CNN-S	3.85	4.62
CIFAR-10	GWN	CNN-L	3.88	4.58
CIFAR-10	Gaussian Blur	CNN-S	3.91	4.84

Table 2. Mean first spike times for correct and incorrect predictions across different datasets and models. The table highlights the significant difference between the first spike times of correctly classified samples and incorrectly classified samples. GWN = Gaussian White Noise

from accuracy, we do the following analysis exclusively on correctly classified samples.

To estimate quantify the effects of testing and training hardness, we fit a ordinary least squares (OLS) regression to predict TTFS as a function of training noise, testing noise, and their interaction. Formally, for test sample i:

$$\mathrm{TTFS}_i = \beta_0 + \beta_1\,\sigma_i^{\text{train}} + \beta_2\,\sigma_i^{\text{test}} + \beta_3\left(\sigma_i^{\text{train}} \times \sigma_i^{\text{test}}\right) + \varepsilon_i \qquad (20)$$

where σ_i^{train} and σ_i^{test} are the noise levels during training and testing, respectively, and ε_i is the residual error.

Although our primary focus is on the marginal effects

Dataset	Hardness	Model Type	Intercept β_0 ($\pm$SE)	Training Noise β_1 ($\pm$SE)	Test Noise β_2 ($\pm$SE)	Interaction β_3 ($\pm$SE)
SepDots	Distribution Overlap	CNN	4.83 (< 0.01)	$-15.5^{\dagger}$ (0.01)	$-9.17^{\dagger}$ (0.01)	45.1 (0.06)
MNIST	Gaussian White Noise	MLP	0.66 (< 0.01)	-0.30 (< 0.01)	2.23 (0.01)	-2.30 (0.02)
MNIST	Gaussian White Noise	CNN	2.34 (< 0.01)	0.37 (0.01)	1.10 (0.01)	0.62 (0.03)
NMNIST	Pixel Flip	CNN	1.94 (< 0.01)	4.13 (0.03)	23.7 (0.03)	$-212.$ (0.44)
CIFAR-10	Gaussian White Noise	CNN-S	3.78 (< 0.01)	1.21 (0.01)	-1.07 (0.01)	4.53 (0.08)
CIFAR-10	Gaussian White Noise	CNN-L	3.81 (< 0.01)	0.14 (< 0.01)	0.14 (< 0.01)	3.76 (0.06)
CIFAR-10	Gaussian Blur	CNN-S	3.70 (< 0.01)	0.11 (< 0.01)	0.12 (< 0.01)	-0.02 (< 0.01)

† For **SepDots** we *expect* β_1 and β_2 to be *negative* due to decreasing hardness with increasing distribution μ. For all other datasets, we expect $\beta_1, \beta_2 > 0$.

Table 3. OLS estimates for predicting TTFS from training noise, test noise, and their interaction (correctly classified samples only). Standard errors in parentheses. N.B., because each dataset's noise levels are defined over a different scale and exhibits a distinct functional relationship with TTFS, the magnitudes of $\beta_1, \beta_2, \beta_3$ estimates aren't directly comparable across datasets.

of training and test noise on TTFS, we additionally include the interaction term $\sigma^{\text{train}} \times \sigma^{\text{test}}$ to guard against potential moderation effects. In this context, a significant interaction would indicate that the impact of increasing test-time noise on TTFS depends on the noise regime under which the network was trained, for instance, a model trained with high noise might exhibit smaller latency shifts when exposed to further noise at test time compared to a model trained on clean data. By modelling this non-additivity, we ensure that the estimates of the main effects β_1 and β_2 remain unbiased and that the overall model fit is improved.

To evaluate how training-noise (hardness) acts as a regularizer on the learned synaptic weights, we fit the following two OLS regression models. Let σ^{train} denote the noise level used during training, and let

$$y_i^{(\text{std})} = \mathrm{StdDev}\big(\mathbf{w}_i^{\text{learn}}\big), \quad y_i^{(\text{mean})} = \mathrm{Mean}\big(\mathbf{w}_i^{\text{learn}}\big)$$

be, respectively, the standard deviation and mean of the synaptic weights measured across the 30 repeats. We then estimate

$$y_i^{(\text{std})} = \beta_4 + \beta_5\,\sigma^{\text{train}} + \varepsilon_i, \tag{21}$$

$$y_i^{(\text{mean})} = \beta_6 + \beta_7\,\sigma^{\text{train}} + \varepsilon_i. \tag{22}$$

Table 4 reports the estimates of β_4 and β_5 for (21), while Table 5 reports the estimates of β_6 and β_7 for (22).

As summarized in Table 3, we fit a OLS to predict the time-to-first-spike based on training noise intensity, test noise intensity, and their interaction. The intercept term, β_0, captures the baseline TTFS for each dataset under zero noise. For the SepDots dataset, both the training-hardness coefficient $\beta_1 = -15.5$ and the testing-hardness coefficient $\beta_2 = -9.17$ are significantly negative[1], providing evidence for our hypothesis that increasing sample difficulty decelerates inference latency. In contrast, for MNIST, NMNIST, and CIFAR-10, the noise coefficients are mostly positive, indicating that higher noise levels tend to delay spike emission, in agreement with our prior research. Finally, only SepDots exhibits a strong positive interaction ($\beta_3 = 45.1$), suggesting a compounding effect when both training and test noises are elevated.

[1]For SepDots we a priori expected $\beta_1, \beta_2 < 0$ due to decreasing hardness (and sample margins) with increasing distribution separation; for all other datasets we hypothesized $\beta_1, \beta_2 > 0$.

4.3. Effect of Test-Time Difficulty on TTFS

The estimates of the test-noise coefficient β_2 across most datasets are positive, indicating that as samples become more difficult at test time (higher σ^{test}), the mean time-to-first-spike increases. This provides direct evidence for our first research question *"What happens to the TTFS behavior of latency-encoded SNNs as samples become increasingly difficult?"*. Furthermore, the results are consistent with drift-diffusion predictions that lower drift rates yield longer response times as harder inputs slow down inference latency [3, 35]. An exception arises for CIFAR-10 with Gaussian white noise, where $\beta_2 = -1.07$ (SE = 0.01), suggesting a slight decrease in latency under added test noise. One plausible explanation is that the CNN never learned a robust feature representation on CIFAR-10 (peak accuracy $\approx 60\%$ across all training-testing combinations), so moderate noise may inadvertently regularize activations in a way that triggers earlier spikes. To an extent, this is supported by the near–zero $\beta_2 = 0.12$ in the Gaussian-blur condition, indicating very minimal sensitivity.

4.4. Effect of Training-Time Difficulty on TTFS

The training-noise coefficient β_1 quantifies whether exposing the network to harder samples during training speeds up or slows down inference, our second research question, *"Can we use harder samples during training to make inference faster?"* Except for SepDots (where hardness is inversely correlated with μ), almost all β_1 estimates are pos-

Dataset	Hardness	Model Type	Intercept β_4 ($\pm$SE)	Training Noise β_5 ($\pm$SE)
SepDots	Distribution Overlap	CNN	0.099 (0.001)	$0.163^{\dagger}$ (0.006)
MNIST	Gaussian White Noise	MLP	0.046 (< 0.001)	-0.008 (< 0.001)
MNIST	Gaussian White Noise	CNN	0.114 (< 0.001)	0.009 (0.001)
NMNIST	Pixel-Flip	CNN	0.056 (< 0.001)	-0.024 (< 0.001)
CIFAR-10	Gaussian White Noise	CNN-S	0.075 (< 0.001)	-0.018 (< 0.001)
CIFAR-10	Gaussian White Noise	CNN-L	0.017 (< 0.001)	-0.003 (< 0.001)
CIFAR-10	Gaussian Blur	CNN-S	0.075 (< 0.001)	-0.002 (< 0.001)

[†] For **SepDots** we *expect* β_5 to be *positive* due to decreasing hardness with increasing distribution μ, i.e., increased standard deviation in weights with increased training distribution μ. For all other datasets, we expect $\beta_5 < 0$.

Table 4. OLS estimates for the effect of training noise on the model's **synaptic weight standard deviation**. In all cases but MNIST-CNN, the standard deviation of synaptic weights decreases with increased training hardness. Giving us a reasonable indication that increased latency during inference due to increased training noise might be stemming from decreased weight variance.

Dataset	Hardness	Model Type	Intercept β_6 ($\pm$SE)	Training Noise β_7 ($\pm$SE)
SepDots	Distribution Overlap	CNN	0.028 (< 0.001)	0.065 (0.004)
MNIST	Gaussian White Noise	MLP	-0.003 (< 0.001)	0.014 (< 0.001)
MNIST	Gaussian White Noise	CNN	0.012 (< 0.001)	0.009 (0.001)
NMNIST	Pixel-Flip	CNN	-0.003 (< 0.001)	-0.005 (0.001)
CIFAR-10	Gaussian White Noise	CNN-S	0.001 (< 0.001)	0.003 (< 0.001)
CIFAR-10	Gaussian White Noise	CNN-L	-0.002 (< 0.001)	< 0.001 (< 0.001)
CIFAR-10	Gaussian Blur	CNN-S	0.001 (< 0.001)	< 0.001 (< 0.001)

Table 5. OLS estimates for the effect of training noise on model's learned **synaptic weights**. Overall, the coefficient β_7 values are near zero, except for SepDots. The effect of training hardness seems to be minimal on mean weight.

itive, indicating that noisier training regimes systematically increase TTFS and thus *slow* inference. Hence, on standard vision benchmarks, harder training does not accelerate first spikes but rather delays them, reflecting a trade-off between robustness and latency. Furthermore, in SepDots, where increasing μ makes samples easier, $\beta_1 = -15.5$ (SE = 0.01) also confirms that training on "easier" distributions quickens inference, consistent with our first-hitting-time model of spiking neuron predictions.

An interesting exception also appears for the MNIST MLP, which exhibits a slightly negative $\beta_1 = -0.30$ (SE < 0.01), suggesting that moderate Gaussian-noise training speeds up TTFS. We hypothesize this arises from an unexpected increase in the mean of synaptic weights under noise (see β_7 for MNIST-MLP in Table 5): whereas statistical learning theory predicts that adding input noise should regularize the network, reducing both the mean and variance of learned weights, the MLP instead showed an upward trend in weight means. Within our stochastic integrate-and-fire framework, an increased mean input current corresponds to a higher effective drift rate, thereby reducing first-hitting times and yielding faster spikes.

As shown in Table Table 4, the coefficient β_5 indicates that increased training hardness reduces parameter variance in all cases except the MNIST-CNN. This aligns with the concept of noise as a regularizer and provides reasonable evidence as to why inference slows down with increased training hardness. In contrast, Table Table 5 shows that β_7 remains mostly near zero, demonstrating that noise has minimal effect on the mean synaptic weight.

Finally, Figure 5 demonstrates that cumulative decision margin $\widetilde{M}$ at the spike time exhibits the inverted-U relationship with latency, very closely mimicking what drift diffusion models predict [18, 35, 36, 38]: TTFS peaks near zero margin and decreases as $|\widetilde{M}|$ grows. This suggests that in SepDots, margin could be an effective proxy for instantaneous drift rate, and that TTFS SNNs embody classic evidence-accumulation dynamics when sample difficulty is measured geometrically.

5. Conclusion

In this work, we have shown that the classical drift-diffusion model, long used to describe decision times in biological neurons and detailed spiking-network simulators, also provides a principled framework for understanding latency-encoded TTFS SNNs in machine-learning settings [44, 46, 47]. Specifically, we hypothesized that (i) harder samples, whether defined by geometric margin or synthetic Gaussian perturbations, would slow down inference by reducing the effective drift rate, and (ii) misclassified trials would exhibit longer first-spike times, mirroring DDM predictions about error-driven RT elongation. Our empirical results confirm both implications: test-noise coefficients (β_2) are overwhelmingly positive (indicating longer TTFS for harder inputs) and mean TTFS for incorrect trials is substantially higher than for correct ones across all datasets. Furthermore, the relationship between cumulative margin $\widetilde{M}$ and latency (Figure 5) exhibits the characteristic inverted-U shape of DDM first-passage times, peaking near zero evidence and decaying as $|\widetilde{M}|$ grows, thus validating geometric margin as a drift-rate proxy in TTFS SNNs.

We built a model based on the classical first-hitting-time theory for static thresholds, most notably the inverse-Gaussian model, to discrete-time leaky integrate-and-fire neurons under sample-dependent inputs. Statistical learning theory predicts that adding Gaussian noise during training is mathematically equivalent to a ridge penalty, thus reducing the variance of the learned synaptic weights [2]. Our Monte Carlo simulations then show that a reduction in weight variance alone systematically increases first-hitting times, and hence TTFS, across discrete timesteps. Moreover, the empirical training-noise coefficients β_1 from our OLS analyses provide strong evidence for this mechanism, as increased training-time hardness led to longer inference latencies in nearly all dataset–model combinations, yielding a clear accuracy-latency trade-off: test-time corruption degraded accuracy sharply unless matched by comparable training-time noise, but this matching incured slower inference. These findings bridge a gap between mathematical neuroscience and ML-oriented TTFS SNNs, demonstrating that (i) first-hitting-time analyses are directly applicable to predict SNN latency under discrete timesteps [39], and (ii) controlling synaptic-weight variance via noise-equivalent regularization provides a tunable mechanism for trading off robustness against speed in neuromorphic classifiers.

References

[1] Aurélien Bellet, Amaury Habrard, and Marc Sebban. A survey on metric learning for feature vectors and structured data. 2013. 2

[2] Christopher M Bishop. *Neural networks for pattern recognition*. 1995. 2, 12

[3] Rafal Bogacz, Eric Brown, Jeff Moehlis, Philip Holmes, and Jonathan D. Cohen. The physics of optimal decision making: a formal analysis of models of performance in two-alternative forced-choice tasks. *Psychological review*, 113: 700–765, 2006. 2, 3, 10

[4] Nicolas Brunel. Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons. *Journal of computational neuroscience*, 8:183–208, 2000. 3

[5] A. N. Burkitt. A review of the integrate-and-fire neuron model: I. homogeneous synaptic input. *Biological Cybernetics*, 95:1–19, 2006. 3

[6] Alexander Camuto, Xiaoyu Wang, Lingjiong Zhu, Chris Holmes, Mert Gürbüzbalaban, and Umut Şimşekli. Asymmetric heavy tails and implicit bias in gaussian noise injections, 2021. 2

[7] Joseph T. Chang and Yuval Peres. Ladder heights, gaussian random walks and the riemann zeta function. *Annals of Probability*, 25:787–802, 1997. 3

[8] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. 1

[9] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. 2

[10] Jason K. Eshraghian, Max Ward, Emre O. Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D. Lu. Training spiking neural networks using lessons from deep learning. *Proceedings of the IEEE*, 111(9):1016–1054, 2023. 1, 4

[11] Wei Fang, Zhaofei Yu, Yanqi Chen, Timothee Masquelier, Tiejun Huang, and Yonghong Tian. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2641–2651, 2021. 4

[12] Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049, 2013. 2

[13] Charles Fefferman, Sergei Ivanov, Matti Lassas, and Hariharan Narayanan. Fitting a manifold of large reach to noisy data. *Journal of Topology and Analysis*, 2019. 2

[14] William Feller. *An Introduction to Probability Theory and Its Applications*. John Wiley & Sons, 1968. 3

[15] J. L. Folks and R. S. Chhikara. The inverse gaussian distribution and its statistical application—a review. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 40:263–275, 1978. 5

[16] Crispin W. Gardiner. *Handbook of Stochastic Methods for Physics, Chemistry and the Natural Sciences*. Springer, 1985. 3

[17] George L. Gerstein and Benoit Mandelbrot. Random walk models for the spike activity of a single neuron. *Biophysical journal*, 4:41–68, 1964. 3

[18] Joshua I. Gold and Michael N. Shadlen. The neural basis of decision making. *Annual Review of Neuroscience*, 30(1): 535–574, 2007. 2, 11

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`. 2

[20] Matthias Hein and Markus Maier. Manifold denoising. *NIPS 2006: Proceedings of the 19th International Conference on Neural Information Processing Systems*, pages 561–568, 2007. 2

[21] Dan Hendrycks and Thomas G. Dietterich. Benchmarking neural network robustness to common corruptions and surface variations, 2018. 2

[22] Irina Higgins, Le Chang, Victoria Langston, Demis Hassabis, Christopher Summerfield, Doris Tsao, and Matthew

Botvinick. Unsupervised deep learning identifies semantic disentanglement in single inferotemporal face patch neurons. *Nature Communications 2021 12:1*, 12:1–14, 2021. 2

[23] Sophie Jaffard, Giulia Mezzadri, Patricia Reynaud-Bouret, and Etienne Tanré. Spiking neural models for decision-making tasks with learning. 2025. 3

[24] Ioannis Karatzas and Steven E. Shreve. *Brownian Motion and Stochastic Calculus*. Springer, 1991. 5

[25] Bruce W. Knight. Dynamics of encoding in a population of neurons. *The Journal of general physiology*, 59:734–766, 1972. 3

[26] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. MIT Press, 2nd edition, 2018. 2

[27] M. J. Mulder, L. van Maanen, and B. U. Forstmann. Perceptual decision neurosciences - a model-based review, 2014. 2, 3

[28] Bernhard Nessler, Michael Pfeiffer, Lars Buesing, and Wolfgang Maass. Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity. *PLOS Computational Biology*, 9:e1003037, 2013. 3

[29] Redmond G. O'Connell, Paul M. Dockree, and Simon P. Kelly. A supramodal accumulation-to-bound signal that determines perceptual decisions in humans. *Nature Neuroscience*, 15, 2012. 2

[30] John Palmer, Alexander C. Huk, and Michael N. Shadlen. The effect of stimulus strength on the speed and accuracy of a perceptual decision. *Journal of Vision*, 5, 2005. 3

[31] Da Costa Gabriel B. Paranhos, Welinton A. Contato, Tiago S. Nazare, João E. S. Batista Neto, and Moacir Ponti. An empirical study on the effects of different types of noise in image classification tasks, 2016. 2

[32] Marios G. Philiastides, Hauke R. Heekeren, and Paul Sajda. Human scalp potentials reflect a mixture of decision- related signals during perceptual choices. *Journal of Neuroscience*, 34, 2014. 2

[33] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *Proceedings of Machine Learning Research*, 139:8748–8763, 2021. 2

[34] Roger Ratcliff. A theory of memory retrieval. *Psychological Review*, 85:59–108, 1978. 2, 3

[35] Roger Ratcliff and Gail McKoon. The diffusion decision model: theory and data for two-choice decision tasks. *Neural computation*, 20:873–922, 2008. 2, 3, 10, 11

[36] Roger Ratcliff and Philip L. Smith. A comparison of sequential sampling models for two-choice reaction time. *Psychological Review*, 111:333–367, 2004. 11

[37] Roger Ratcliff and Francis Tuerlinckx. Estimating parameters of the diffusion model: Approaches to dealing with contaminant reaction times and parameter variability. *Psychonomic Bulletin and Review*, 9, 2002. 3

[38] Roger Ratcliff, Philip L. Smith, and Gail McKoon. Modeling regularities in response time and accuracy data with the diffusion model. *Current directions in psychological science*, 24:458, 2015. 3, 11

[39] Arnold J.F. Siegert. On the first passage time probability problem. *Physical Review*, 81:617, 1951. 3, 12

[40] David Siegmund. *Sequential Analysis*. Springer New York, 1985. 3

[41] Michael Reed Smith, Tony Martinez, and Cristophe Giraud-Carrier. An empirical study of instance hardness, 2010. 1

[42] Jonathan Touboul and Olivier Faugeras. A characterization of the first hitting time of double integral processes to curved boundaries. *Advances in Applied Probability*, 40:501–528, 2008. 3

[43] K. Tumer and J. Ghosh. Estimating the bayes error rate through classifier combining. In *Proceedings of 13th International Conference on Pattern Recognition*, pages 695–699 vol.2, 1996. 2

[44] Akash Umakantha, Braden A. Purcell, and Thomas J. Palmeri. Relating a spiking neural network model and the diffusion model of decision-making. *Computational brain & behavior*, 5:279, 2022. 3, 12

[45] Kush R Varshney, Alan S Wilskv, and Edwin Sibley Webster. Frugal hypothesis testing and classification. 2010. 2

[46] Xiao Jing Wang. Probabilistic decision making by slow reverberation in cortical circuits. *Neuron*, 36:955–968, 2002. 3, 12

[47] Kong Fatt Wong and Xiao Jing Wang. A recurrent network mechanism of time integration in perceptual decisions. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 26:1314–1328, 2006. 3, 12

A. Performance of Trainings

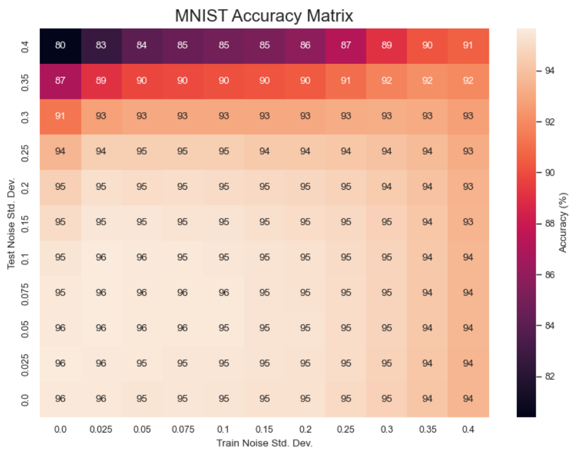


Figure 11. Accuracy matrix for MNIST-MLP

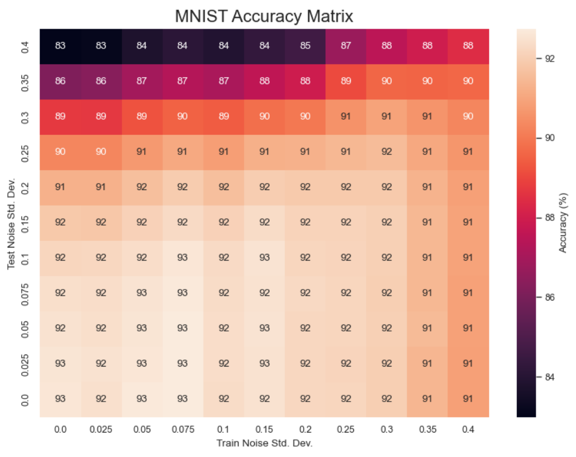


Figure 12. Accuracy matrix for MNIST-CNN

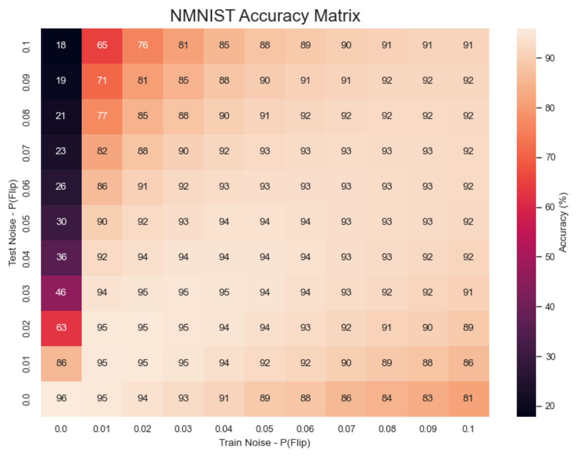


Figure 13. Accuracy matrix for NMNIST-CNN

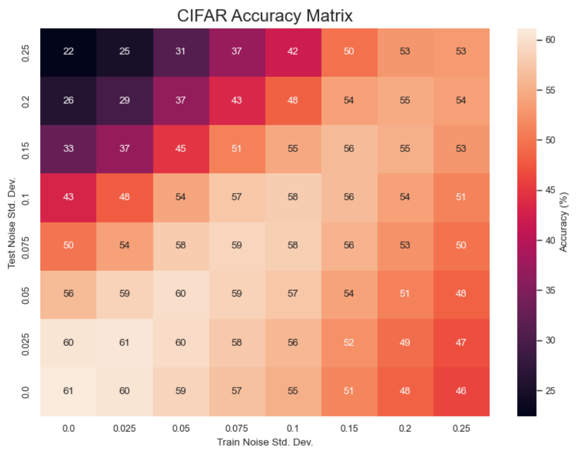


Figure 14. Accuracy matrix for CIFAR-CNN

Figure 11 up to Figure 15 present heatmaps of test accuracy (%) as a function of training-time noise (x-axis) and test-time noise (y-axis) for each dataset–model combination. Several patterns are apparent:

- **MNIST-MLP (Fig. 11).** High accuracy ($> 90\%$) is maintained across low to moderate noise levels; performance only degrades when both training and test noise exceed approximately $\sigma = 0.3$. Models trained with nonzero noise exhibit greater robustness to test-time corruption than those trained on clean images.
- **MNIST-CNN (Fig. 12).** The convolutional network achieves peak accuracies near 93% under clean conditions and shows a more gradual decline under increasing test noise compared to the MLP, sustaining $\geq 90\%$ accuracy up to $\sigma \approx 0.3$ even when trained without noise.
- **NMNIST-CNN (Fig. 13).** Baseline accuracy on event-based inputs is around 95%. Test-time pixel-flip rates above 0.02 causes very steep drops in accuracy in the clean training case, whereas injecting 5–10% flips during training shifts the robust region upward, preserving 80–90% accuracy under moderate corruption.
- **CIFAR-CNN with Gaussian Noise (Fig. 14).** Clean accuracy peaks near 60%. Small amounts of training noise ($\sigma \leq 0.05$) modestly improve clean and noisy inference, but heavy noise degrades performance across the board.
- **CIFAR-CNN with Gaussian Blur (Fig. 15).** Test-time blur with $\sigma \geq 1.0$ reduces accuracy by 10–15%. Light blur augmentation during training ($\sigma = 0.25$–0.5) partially mitigates this drop, whereas strong blur training harms both clean and blurred inference.

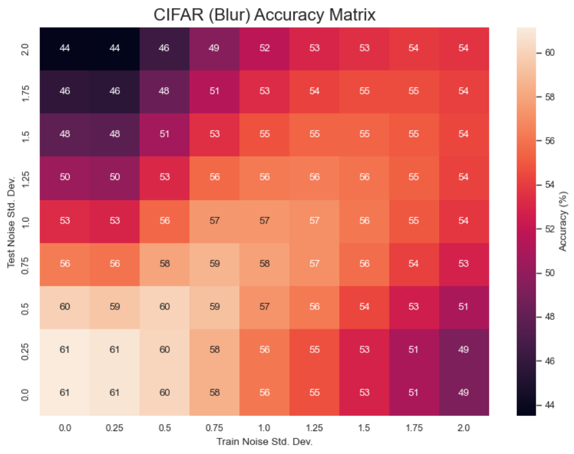


Figure 15. Accuracy matrix for CIFAR-CNN with blur

B. Noise and Synaptic Weights

The line plots in Figure Figure 16 summarize the overall shift in the first-layer weight distribution as a function of training noise standard deviation. The left plot shows the mean deviation from the noiseless baseline, and the right plot shows the variance. As σ increases further, both metrics decrease monotonically, reflecting a global *smoothing* or regularization effect as the network learns to ignore high-frequency fluctuations and focus on robust features.

However, these aggregate statistics obscure important *spatial* patterns. The heatmaps in Figure 17 reveal that at low noise levels ($\sigma = 0.0125$–0.05), the model develops large negative weights on the border pixels (bright red regions), effectively "subtracting" noisy edges instead of just ignoring them with zero-mean edges. This pixel-wise overfitting is invisible in the line plots, which simply report an average increase in variance. As noise grows beyond $\sigma \approx 0.1$, the border artifacts fade and the receptive field becomes smoother and more centrally focused on the digit shape—precisely the smoothing trend hinted at by the declining variance in Figure 13. Thus, while the line plots capture the *magnitude* of regularization, the heatmaps expose the *location* and *nature* of the weight adjustments induced by noise training.

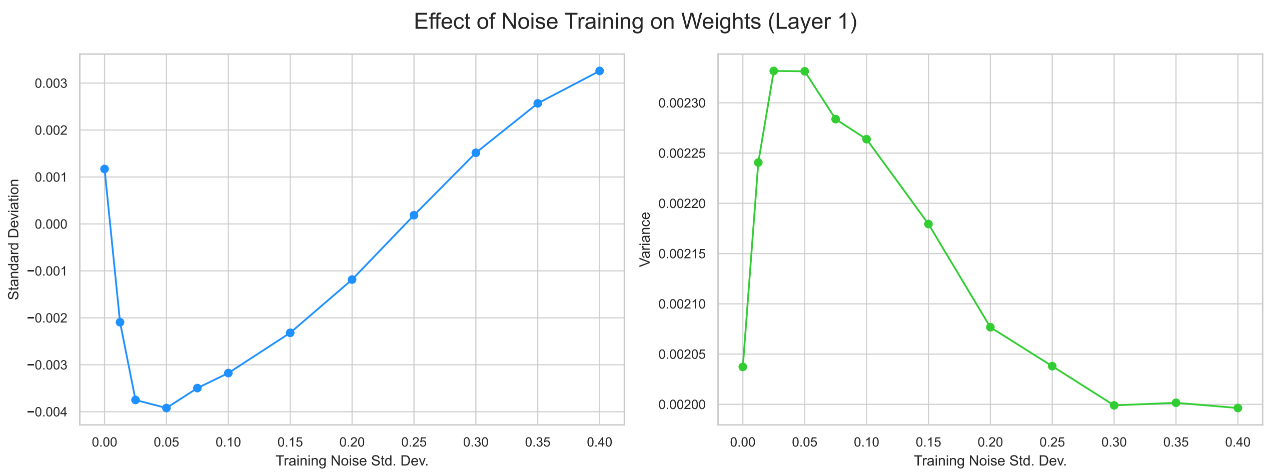


Figure 16. Statistics of first-layer weights in MNIST-MLP as a function of training noise standard deviation.

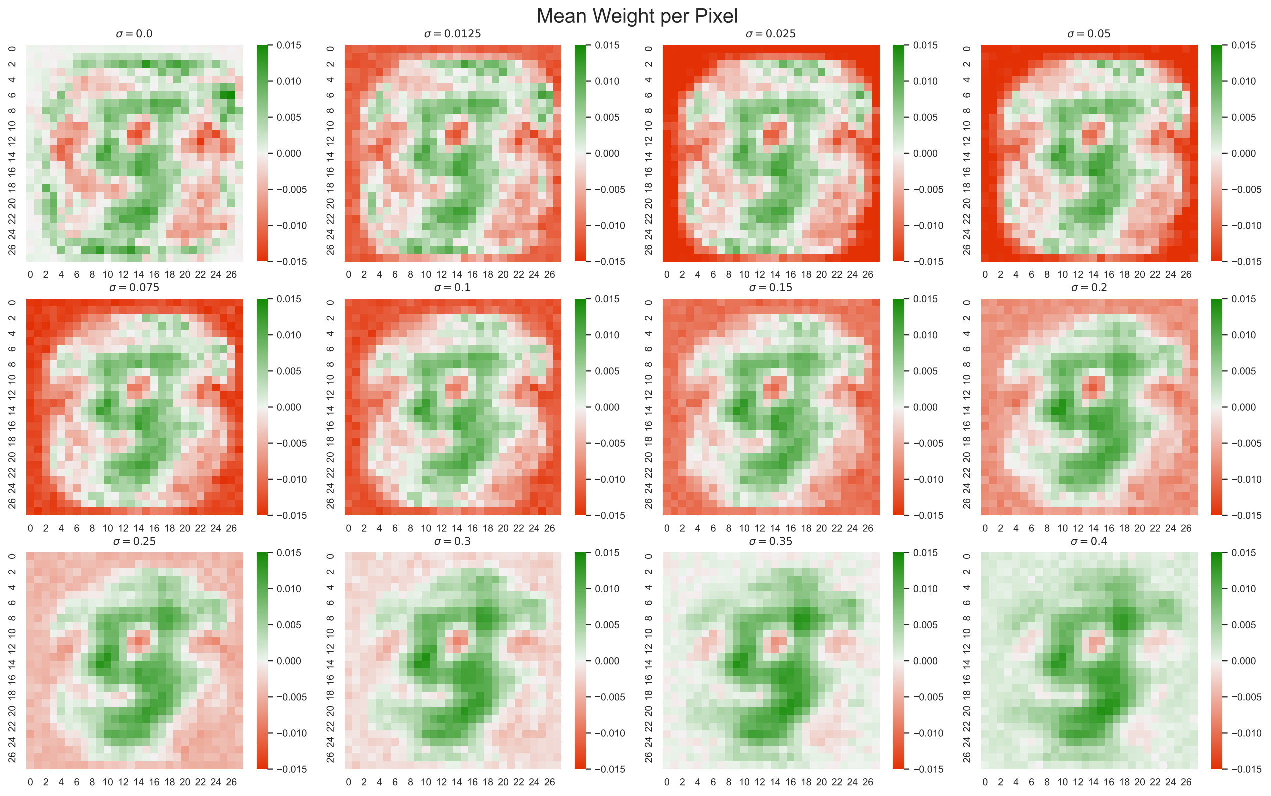


Figure 17. Pixel-wise mean edge weights in MNIST-MLP for various training noise levels.

References

[1] Chirag Agarwal, Daniel D'souza, and Sara Hooker. *Estimating example difficulty using variance of gradients*. Aug. 2020. URL: https://arxiv.org/abs/2008.11600.

[2] Mikhail Belkin et al. "Reconciling modern machine-learning practice and the classical bias–variance trade-off". In: *Proceedings of the National Academy of Sciences* 116.32 (2019), pp. 15849–15854. DOI: 10.1073/pnas.1903070116. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.1903070116. URL: https://www.pnas.org/doi/abs/10.1073/pnas.1903070116.

[3] Aurélien Bellet, Amaury Habrard, and Marc Sebban. "A Survey on Metric Learning for Feature Vectors and Structured Data". In: (June 2013). URL: https://arxiv.org/abs/1306.6709v4.

[4] Benyamin Ghojogh BGHOJOGH et al. "Spectral, Probabilistic, and Deep Metric Learning: Tutorial and Survey". In: (Jan. 2022). URL: https://arxiv.org/abs/2201.09267v1.

[5] Christopher M Bishop. *Neural networks for pattern recognition*. Nov. 1995. DOI: 10.1093/oso/9780198538493.001.0001. URL: https://doi.org/10.1093/oso/9780198538493.001.0001.

[6] Christopher M. Bishop. "Pattern Recognition and Machine Learning". In: *Springer* (2006). DOI: 10.1007/978-0-387-45528-0.

[7] Fischer Black and Myron Scholes. "The pricing of options and corporate liabilities". In: *Journal of Political Economy* 81.3 (May 1973), pp. 637–654. DOI: 10.1086/260062. URL: https://doi.org/10.1086/260062.

[8] Rafal Bogacz et al. "The physics of optimal decision making: a formal analysis of models of performance in two-alternative forced-choice tasks". In: *Psychological review* 113 (4 Oct. 2006), pp. 700–765. ISSN: 0033-295X. DOI: 10.1037/0033-295X.113.4.700. URL: https://pubmed.ncbi.nlm.nih.gov/17014301/.

[9] A. N. Burkitt. "A review of the integrate-and-fire neuron model: I. Homogeneous synaptic input". In: *Biological Cybernetics* 95 (1 July 2006), pp. 1–19. ISSN: 03401200. DOI: 10.1007/S00422-006-0068-6/METRICS. URL: https://link.springer.com/article/10.1007/s00422-006-0068-6.

[10] Alexander Camuto et al. *Asymmetric heavy tails and implicit bias in gaussian noise injections*. Feb. 2021. URL: https://arxiv.org/abs/2102.07006.

[11] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine Learning* 20.3 (Sept. 1995), pp. 273–297. DOI: 10.1007/bf00994018. URL: https://doi.org/10.1007/bf00994018.

[12] Susanne Ditlevsen and Petr Lansky. "Parameters of stochastic diffusion processes estimated from observations of first-hitting times: Application to the leaky integrate-and-fire neuronal model". In: *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics* 76 (4 Oct. 2007), p. 041906. ISSN: 15393755. DOI: 10.1103/PHYSREVE.76.041906. URL: https://journals.aps.org/pre/abstract/10.1103/PhysRevE.76.041906.

[13] Peter Duggins and Chris Eliasmith. "A spiking neural model of decision making and the speed–accuracy trade-off." In: *Psychological Review* (Dec. 12, 2024). DOI: 10.1037/rev0000520. URL: https://doi.org/10.1037/rev0000520.

[14] Gamaleldin F. Elsayed et al. "Large Margin Deep Networks for Classification". In: *Advances in Neural Information Processing Systems* 2018-December (Mar. 2018), pp. 842–852. ISSN: 10495258. URL: https://arxiv.org/abs/1803.05598v2.

[15] Jason K. Eshraghian et al. "Training Spiking Neural Networks Using Lessons From Deep Learning". In: *Proceedings of the IEEE* 111.9 (Sept. 2023), pp. 1016–1054. ISSN: 1558-2256. DOI: 10.1109/JPROC.2023.3308088.

[16] Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. "Testing the manifold hypothesis". In: *Journal of the American Mathematical Society* 29.4 (Oct. 2013), pp. 983–1049. DOI: 10.1090/jams/852. URL: https://doi.org/10.1090/jams/852.

[17] William Feller. *An Introduction to Probability Theory and Its Applications*. Vol. 1. John Wiley & Sons, 1968.

[18] J. L. Folks and R. S. Chhikara. “The Inverse Gaussian Distribution and its Statistical Application—A Review”. In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 40 (3 July 1978), pp. 263–275. ISSN: 1369-7412. DOI: 10.1111/J.2517-6161.1978.TB01039.X. URL: https://dx.doi.org/10.1111/j.2517-6161.1978.tb01039.x.

[19] Steve Furber. “Large-scale neuromorphic computing systems”. In: *Journal of neural engineering* 13 (5 Aug. 2016). ISSN: 1741-2552. DOI: 10.1088/1741-2560/13/5/051001. URL: https://pubmed.ncbi.nlm.nih.gov/27529195/.

[20] Crispin W. Gardiner. *Handbook of Stochastic Methods for Physics, Chemistry and the Natural Sciences*. Springer, 1985.

[21] Stuart Geman, Elie Bienenstock, and René Doursat. “Neural Networks and the Bias/Variance Dilemma”. In: *Neural Computation* 4.1 (Jan. 1992), pp. 1–58. ISSN: 0899-7667. DOI: 10.1162/neco.1992.4.1.1. eprint: https://direct.mit.edu/neco/article-pdf/4/1/1/812244/neco.1992.4.1.1.pdf. URL: https://doi.org/10.1162/neco.1992.4.1.1.

[22] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. *Time matters in regularizing deep networks: weight decay and data augmentation affect early learning dynamics, matter little near convergence*. May 2019. URL: https://arxiv.org/abs/1905.13277.

[23] Joshua I. Gold and Michael N. Shadlen. “The neural basis of decision making”. In: *Annual Review of Neuroscience* 30.1 (Mar. 2007), pp. 535–574. DOI: 10.1146/annurev.neuro.29.051605.113038. URL: https://doi.org/10.1146/annurev.neuro.29.051605.113038.

[24] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[25] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[26] A. N. Gorban and I. Y. Tyukin. “Blessing of dimensionality: mathematical foundations of the statistical physics of data”. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 376 (2118 Apr. 2018). ISSN: 1364503X. DOI: 10.1098/RSTA.2017.0237. URL: https://royalsocietypublishing.org/doi/10.1098/rsta.2017.0237.

[27] Felix Grün et al. *A taxonomy and library for visualizing learned features in convolutional neural networks*. June 2016. URL: https://arxiv.org/abs/1606.07757.

[28] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning*. Jan. 2009. DOI: 10.1007/978-0-387-84858-7. URL: https://doi.org/10.1007/978-0-387-84858-7.

[29] Daniel Hausmann and Damian Läge. “Sequential evidence accumulation in decision making: The individual desired level of confidence can explain the extent of information acquisition”. In: *Judgment and Decision Making* 3 (3 Mar. 2008), pp. 229–243. ISSN: 1930-2975. DOI: 10.1017/S1930297500002436. URL: https://www.cambridge.org/core/journals/judgment-and-decision-making/article/sequential-evidence-accumulation-in-decision-making-the-individual-desired-level-of-confidence-can-explain-the-extent-of-information-acquisition/032EC02561554A883F5B06D31B5A687F.

[30] Dan Hendrycks and Thomas G. Dietterich. *Benchmarking neural network robustness to common corruptions and surface variations*. July 2018. URL: https://arxiv.org/abs/1807.01697.

[31] Sepp Hochreiter and Jürgen Schmidhuber. “Flat Minima”. In: *Neural Computation* 9.1 (Jan. 1997), pp. 1–42. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.1.1. eprint: https://direct.mit.edu/neco/article-pdf/9/1/1/813385/neco.1997.9.1.1.pdf. URL: https://doi.org/10.1162/neco.1997.9.1.1.

[32] A. L. Hodgkin and A. F. Huxley. “A quantitative description of membrane current and its application to conduction and excitation in nerve”. In: *The Journal of Physiology* 117 (4 Aug. 1952), p. 500. ISSN: 14697793. DOI: 10.1113/JPHYSIOL.1952.SP004764. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC1392413/.

[33] Arthur E. Hoerl and Robert W. Kennard. “Ridge regression: Biased estimation for nonorthogonal problems”. In: *Technometrics* 12.1 (Feb. 1970), pp. 55–67. DOI: 10.1080/00401706.1970.10488634. URL: https://doi.org/10.1080/00401706.1970.10488634.

[34] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. In: *Neural Networks* 2.5 (1989), pp. 359–366. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(89)90020-8. URL: https://www.sciencedirect.com/science/article/pii/0893608089900208.

[35] Giacomo Indiveri and Shih-Chii Liu. “Memory and information processing in neuromorphic systems”. In: *Proceedings of the IEEE* 103 (8 June 2015), pp. 1379–1397. DOI: 10.1109/JPROC.2015.2444094. URL: http://arxiv.org/abs/1506.03264%20http://dx.doi.org/10.1109/JPROC.2015.2444094.

[36] Sophie Jaffard et al. “Spiking Neural Models for Decision-Making Tasks with Learning”. In: (June 2025). URL: https://arxiv.org/abs/2506.09087v1.

[37] Gaojie Jin et al. *How does Weight Correlation Affect the Generalisation Ability of Deep Neural Networks*. Oct. 2020. URL: https://arxiv.org/abs/2010.05983.

[38] Ioannis Karatzas and Steven E. Shreve. *Brownian Motion and Stochastic Calculus*. Springer, 1991.

[39] Nitish Shirish Keskar et al. *On Large-Batch training for deep learning: generalization gap and sharp minima*. Sept. 2016. URL: https://arxiv.org/abs/1609.04836.

[40] Diederik P. Kingma and Jimmy Ba. *Adam: A method for stochastic optimization*. Dec. 2014. URL: https://arxiv.org/abs/1412.6980.

[41] Anders Krogh and John A. Hertz. “A simple weight decay can improve generalization”. In: *Proceedings of the 5th International Conference on Neural Information Processing Systems*. NIPS’91. Denver, Colorado: Morgan Kaufmann Publishers Inc., 1991, pp. 950–957. ISBN: 1558602224.

[42] Devin Kwok et al. “Dataset Difficulty and the Role of Inductive Bias”. In: (Jan. 2024). URL: http://arxiv.org/abs/2401.01867.

[43] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *Nature* 521.7553 (May 2015), pp. 436–444. DOI: 10.1038/nature14539. URL: https://pubmed.ncbi.nlm.nih.gov/26017442/.

[44] Peter Lennie. “The cost of cortical computation”. In: *Current Biology* 13.6 (Mar. 1, 2003), pp. 493–497. DOI: 10.1016/s0960-9822(03)00135-0. URL: https://doi.org/10.1016/s0960-9822(03)00135-0.

[45] Jonathan Long, Evan Shelhamer, and Trevor Darrell. *Fully convolutional networks for semantic segmentation*. Nov. 2014. URL: https://arxiv.org/abs/1411.4038.

[46] Wenjie Luo et al. *Understanding the effective receptive field in deep convolutional neural networks*. Jan. 2017. URL: https://arxiv.org/abs/1701.04128.

[47] Wolfgang Maass. “Networks of spiking neurons: The third generation of neural network models”. In: *Neural Networks* 10 (9 Dec. 1997), pp. 1659–1671. ISSN: 0893-6080. DOI: 10.1016/S0893-6080(97)00011-7.

[48] Paul A. Merolla et al. “A million spiking-neuron integrated circuit with a scalable communication network and interface”. In: *Science* 345 (6197 Aug. 2014), pp. 668–673. ISSN: 10959203. DOI: 10.1126/SCIENCE.1254642/SUPPL_FILE/MEROLLA.SM.REV1.PDF. URL: https://www.science.org/doi/10.1126/science.1254642.

[49] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. 2nd. MIT Press, 2018.

[50] Hesham Mostafa. “Supervised Learning Based on Temporal Coding in Spiking Neural Networks”. In: *IEEE Transactions on Neural Networks and Learning Systems* 29.7 (2018), pp. 3227–3235. DOI: 10.1109/TNNLS.2017.2726060.

[51] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Jan. 1995. URL: https://dl.acm.org/citation.cfm?id=234327.

[52] Catherine E. Myers, Alejandro Interian, and Ahmed A. Moustafa. “A practical introduction to using the drift diffusion model of decision-making in cognitive psychology, neuroscience, and health sciences”. In: *Frontiers in Psychology* 13 (Dec. 2022), p. 1039172. ISSN: 16641078. DOI: 10.3389/FPSYG.2022.1039172/BIBTEX.

[53] Gregory Naitzat, Andrey Zhitnikov, and Lek-Heng Lim. “Topology of deep neural networks”. In: *Journal of Machine Learning Research* 21.184 (Apr. 2020), pp. 1–40. URL: https://jmlr.org/papers/volume21/20-345/20-345.pdf.

[54] Brady Neal et al. “A modern take on the Bias-Variance tradeoff in neural networks”. In: *arXiv (Cornell University)* (Jan. 2018). DOI: 10.48550/arxiv.1810.08591. URL: https://arxiv.org/abs/1810.08591.

[55] Emre O. Neftci, Hesham Mostafa, and Friedemann Zenke. “Surrogate Gradient Learning in Spiking Neural Networks”. In: (Jan. 2019). URL: https://arxiv.org/abs/1901.09948v2.

[56] Da Costa Gabriel B. Paranhos et al. *An empirical study on the effects of different types of noise in image classification tasks*. Sept. 2016. URL: https://arxiv.org/abs/1609.02781.

[57] Seongsik Park, Dongjin Lee, and Sungroh Yoon. *Noise-Robust Deep Spiking Neural Networks with Temporal Information*. Apr. 2021. URL: https://arxiv.org/abs/2104.11169.

[58] Pawel Pukowski and Haiping Lu. *Investigating the impact of hard samples on accuracy reveals in-class data imbalance*. Sept. 2024. URL: https://arxiv.org/abs/2409.14401.

[59] Roger Ratcliff. “A theory of memory retrieval”. In: *Psychological Review* 85 (2 Mar. 1978), pp. 59–108. ISSN: 0033295X. DOI: 10.1037/0033-295X.85.2.59.

[60] Roger Ratcliff and Gail McKoon. “The diffusion decision model: theory and data for two-choice decision tasks”. In: *Neural computation* 20 (4 Apr. 2008), pp. 873–922. ISSN: 0899-7667. DOI: 10.1162/NECO.2008.12-06-420. URL: https://pubmed.ncbi.nlm.nih.gov/18085991/.

[61] Roger Ratcliff and Philip L. Smith. “A Comparison of Sequential Sampling Models for Two-Choice Reaction Time”. In: *Psychological Review* 111 (2 Apr. 2004), pp. 333–367. ISSN: 0033295X. DOI: 10.1037/0033-295X.111.2.333.

[62] Roger Ratcliff, Philip L. Smith, and Gail McKoon. “Modeling Regularities in Response Time and Accuracy Data with the Diffusion Model”. In: *Current directions in psychological science* 24 (6 Dec. 2015), p. 458. ISSN: 14678721. DOI: 10.1177/0963721415596228. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC4692464/.

[63] Roger Ratcliff et al. “Diffusion Decision Model: Current Issues and History”. In: *Trends in cognitive sciences* 20 (4 Apr. 2016), p. 260. ISSN: 1879307X. DOI: 10.1016/J.TICS.2016.01.007. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC4928591/.

[64] Sidney Redner. “A Guide to First-Passage Processes”. In: *A Guide to First-Passage Processes* (Aug. 2001). DOI: 10.1017/CBO9780511606014. URL: https://www.cambridge.org/core/books/guide-to-firstpassage-processes/59066FD9754B42D22B028E33726D1F07.

[65] F. Rosenblatt. “The perceptron: A probabilistic model for information storage and organization in the brain.” In: *Psychological Review* 65.6 (Jan. 1958), pp. 386–408. DOI: 10.1037/h0042519. URL: https://doi.org/10.1037/h0042519.

[66] Sheldon M. Ross. *Introduction to probability models*. Academic Press, Dec. 2009.

[67] Ran Rubin and Haim Sompolinsky. *Temporal support vectors for spiking neuronal networks*. May 2022. URL: https://arxiv.org/abs/2205.14544.

[68] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. *(1986) D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," Parallel Distributed Processing: Explorations in the Microstructures of Cognition, Vol. I, D. E. Rumelhart and J. L. McClelland (Eds.) Cambridge, MA: MIT Press, pp. 318-362*. Apr. 1988, pp. 675–695. DOI: 10.7551/mitpress/4943.003.0128. URL: https://doi.org/10.7551/mitpress/4943.003.0128.

[69] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From theory to Algorithms*. Jan. 2015. URL: https://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/understanding-machine-learning-theory-algorithms.pdf.

[70] Karen Simonyan and Andrew Zisserman. *Very deep convolutional networks for Large-Scale image recognition*. Sept. 2014. URL: https://arxiv.org/abs/1409.1556.

[71] Michael Reed Smith, Tony Martinez, and Cristophe Giraud-Carrier. *An empirical study of instance hardness*. 2010. URL: https://axon.cs.byu.edu/papers/smith.empirical.pdf.

[72] Mustafa Sungkar, Toby Berger, and William B. Levy. “Capacity achieving input distribution to the generalized inverse Gaussian neuron model”. In: *55th Annual Allerton Conference on Communication, Control, and Computing, Allerton 2017* 2018-January (July 2017), pp. 860–869. DOI: 10.1109/ALLERTON.2017.8262829.

[73] Christian Szegedy et al. *Going Deeper with Convolutions*. Sept. 2014. URL: https://arxiv.org/abs/1409.4842.

[74] Simon Thorpe, Arnaud Delorme, and Rufin Van Rullen. “Spike-based strategies for rapid processing”. In: *Neural Networks* 14 (6-7 July 2001), pp. 715–725. ISSN: 0893-6080. DOI: 10.1016/S0893-6080(01)00083-1.

[75] Simon Thorpe, Denis Fize, and Catherine Marlot. “Speed of processing in the human visual system”. In: *Nature 1996 381:6582* 381 (6582 1996), pp. 520–522. ISSN: 1476-4687. DOI: 10.1038/381520a0. URL: https://www.nature.com/articles/381520a0.

[76] Mariya Toneva et al. *An Empirical Study of Example Forgetting during Deep Neural Network Learning*. Dec. 2018. URL: https://arxiv.org/abs/1812.05159.

[77] Jonathan Touboul. “Stochastic Processes and Hitting Times in Mathematical Neurosciences”. In: (2006). DOI: 10.34894/VQ1DJA. URL: https://inria.hal.science/inria-00311967%20https://inria.hal.science/inria-00311967/document.

[78] Jonathan Touboul and Olivier Faugeras. “A characterization of the first hitting time of double integral processes to curved boundaries”. In: *Advances in Applied Probability* 40 (2 June 2008), pp. 501–528. ISSN: 0001-8678. DOI: 10.1239/AAP/1214950214. URL: https://www.cambridge.org/core/journals/advances-in-applied-probability/article/characterization-of-the-first-hitting-time-of-double-integral-processes-to-curved-boundaries/0B1A26BC686C8A5D5869AEA269CB016B.

[79] K. Tumer and J. Ghosh. “Estimating the Bayes error rate through classifier combining”. In: *Proceedings of 13th International Conference on Pattern Recognition*. Vol. 2. 1996, 695–699 vol.2. DOI: 10.1109/ICPR.1996.546912.

[80] Akash Umakantha, Braden A. Purcell, and Thomas J. Palmeri. “Relating a Spiking Neural Network Model and the Diffusion Model of Decision-Making”. In: *Computational brain & behavior* 5 (3 Sept. 2022), p. 279. ISSN: 2522087X. DOI: 10.1007/S42113-022-00143-4. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC9673774/.

[81] Rufin VanRullen and Simon J. Thorpe. “The time course of visual processing: from early perception to decision-making”. In: *Journal of cognitive neuroscience* 13 (4 2001), pp. 454–461. ISSN: 0898-929X. DOI: 10.1162/08989290152001880. URL: https://pubmed.ncbi.nlm.nih.gov/11388919/.

[82] Kush R Varshney, Alan S Wilskv, and Edwin Sibley Webster. “Frugal hypothesis testing and classification”. In: (2010). URL: https://dspace.mit.edu/handle/1721.1/60182.

[83] Xiao Jing Wang. “Probabilistic Decision Making by Slow Reverberation in Cortical Circuits”. In: *Neuron* 36 (5 Dec. 2002), pp. 955–968. ISSN: 0896-6273. DOI: 10.1016/S0896-6273(02)01092-9.

[84] Hui Wei, Yijie Bu, and Dawei Dai. “A decision-making model based on a spiking neural circuit and synaptic plasticity”. In: *Cognitive Neurodynamics* 11 (5 Oct. 2017), p. 415. ISSN: 18714099. DOI: 10.1007/S11571-017-9436-2. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC5637713/.

[85] Paul J. Werbos. “Backpropagation Through Time: What It Does and How to Do It”. In: *Proceedings of the IEEE* 78 (10 1990), pp. 1550–1560. ISSN: 15582256. DOI: 10.1109/5.58337.

[86] Kong Fatt Wong and Xiao Jing Wang. “A recurrent network mechanism of time integration in perceptual decisions”. In: *The Journal of neuroscience : the official journal of the Society for Neuroscience* 26 (4 Jan. 2006), pp. 1314–1328. ISSN: 1529-2401. DOI: 10.1523/JNEUROSCI.3733-05.2006. URL: https://pubmed.ncbi.nlm.nih.gov/16436619/.

[87] Kashu Yamazaki et al. “Spiking Neural Networks and Their Applications: A Review”. In: *Brain sciences* 12 (7 July 2022). ISSN: 2076-3425. DOI: 10.3390/BRAINSCI12070863. URL: https://pubmed.ncbi.nlm.nih.gov/35884670/.

[88] Tom Yan and Chicheng Zhang. “Margin-distancing for safe model explanation”. In: (2022).

[89] Leijun Ye and Chunhe Li. “Quantifying the Landscape of Decision Making From Spiking Neural Networks”. In: *Frontiers in Computational Neuroscience* 15 (Oct. 2021), p. 740601. ISSN: 16625188. DOI: 10.3389/FNCOM.2021.740601/BIBTEX. URL: www.frontiersin.org.

[90] Jason Yosinski et al. *Understanding neural networks through deep visualization*. June 2015. URL: https://arxiv.org/abs/1506.06579.

[91] Wei Yu et al. *Visualizing and comparing convolutional neural networks*. Dec. 2014. URL: https://arxiv.org/abs/1412.6631.

[92] Matthew D Zeiler and Rob Fergus. *Visualizing and understanding convolutional networks*. Nov. 2013. URL: https://arxiv.org/abs/1311.2901.

[93] Mo Zhou et al. *Toward understanding the importance of noise in training neural networks*. May 2019. URL: https://proceedings.mlr.press/v97/zhou19d.html.

[94] Weiyao Zhu et al. “Exploring the Learning Difficulty of Data: Theory and Measure”. In: *ACM Transactions on Knowledge Discovery from Data* 18 (4 Feb. 2024). ISSN: 1556472X. DOI: 10.1145/3636512. URL: https://dl.acm.org/doi/10.1145/3636512.

[95] Hui Zou and Trevor Hastie. “Regularization and Variable Selection Via the Elastic Net”. In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 67.2 (Mar. 2005), pp. 301–320. ISSN: 1369-7412. DOI: 10.1111/j.1467-9868.2005.00503.x. eprint: https://academic.oup.com/jrsssb/article-pdf/67/2/301/49795094/jrsssb_67_2_301.pdf. URL: https://doi.org/10.1111/j.1467-9868.2005.00503.x.