Neural Network Approximation of Lévy Processes: Theory and Applications in Finance

Ali Abdul Rahim

December 14, 2024

Abstract

Uncertainty is a defining feature of financial systems, with implications that transcend borders and influence global economic stability. The intertwined nature of financial markets, coupled with their inherent volatility, necessitates advanced mathematical tools for effective risk quantification and management. Stochastic processes provide a rigorous framework for understanding and modeling the randomness inherent in financial markets. However, the increasing complexity of the modern financial landscape demands an interdisciplinary approach, integrating classical stochastic methods with contemporary computational advancements such as machine learning. This paper examines the theoretical foundations and recent advances in approximating Lévy processes using neural networks, with particular emphasis on Neural Jump Stochastic Differential Equations. I begin with the measure-theoretic foundations of Lévy processes, establishing key results about their structure and properties. The connection to neural network approximation is then developed through the lens of simple process density theorems and universal approximation results. I provide detailed convergence proofs and explicit error bounds for neural approximations of Lévy processes, demonstrating how architectural choices affect approximation quality. The synthesis reveals both the power of combining classical stochastic process theory with modern machine learning and the challenges that remain.

1 Introduction

The mathematical modeling of financial markets has been profoundly shaped by the Black-Scholes model[1, 3], which assumes asset prices follow geometric Brownian motion:

$$dS_t = \mu S_t \, dt + \sigma S_t \, dB_t$$

This elegant formulation, where B_t represents Brownian motion, has become the foundation of modern financial theory. However, high-frequency trading data reveals significant departures from this model's predictions[4]. Asset prices exhibit sudden jumps, returns show heavy tails, and volatility clusters in ways that Brownian motion cannot capture[4].

Lévy processes provide a natural mathematical framework for addressing these limitations. These processes generalize Brownian motion by allowing for jumps while maintaining crucial properties like independent increments. A Lévy process can be thought of as a continuous motion (like Brownian motion) combined with random jumps of various sizes[5]. This combination captures both the day-to-day fluctuations and the sudden price changes observed in financial markets.

However, working with Lévy processes presents significant challenges. Their distributions are often complex, parameters are difficult to estimate, and simulation can be computationally intensive. Traditional numerical methods struggle to capture both the continuous and jump components efficiently.

Recent advances in deep learning suggest a promising direction[10, 7, 6]. Neural networks, with their universal approximation properties, offer powerful tools for modeling complex stochastic processes. Universal approximation theorems state that for any continuous function on a compact domain, and for any $\epsilon > 0$, there exists a neural network with a single hidden layer that can approximate the function uniformly within ϵ. Yet, it is often impractical to do so for most functions. There have been recent attempts to design neural networks that can approximate Lévy processes allowing for faster computation than traditional numerical methods. However, the integration Lévy processes with deep learning raises fundamental mathematical questions:

1. How can we ensure neural networks preserve the essential properties of Lévy processes?

2. What theoretical guarantees can we establish for such approximations?

3. How do we handle the interplay between continuous and jump components?

This paper develops a mathematical framework for addressing these questions. We begin with the foundations of Lévy processes, establish key theoretical results for neural network approximation, and demonstrate practical applications in high-frequency trading. While motivated by financial applications, our results extend to any domain involving jump processes.

The remainder of this paper is organized as follows. Section 2 establishes the measure-theoretic foundations of Lévy processes. Section 3 develops the core theory of neural network approximation for jump processes, including novel convergence results. Section 4 presents numerical methods and implementation details, while Section 5 is a brief overview applications to trading and markets environments.

2 Foundations of Lévy Processes

The rigorous study of Lévy processes requires careful consideration of their measure-theoretic foundations. This section develops the necessary mathematical framework, beginning with filtered probability spaces and building toward the characterization of Lévy processes.

2.1 Filtered Probability Spaces and Cadlag Functions

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a complete probability space. A filtration $(\mathcal{F}_t)_{t\geq 0}$ is a family of σ-algebras satisfying:

1. $\mathcal{F}_s \subseteq \mathcal{F}_t$ for $s \leq t$ (monotonicity)

2. $\mathcal{F}_t = \bigcap_{s>t} \mathcal{F}_s$ (right-continuity)

3. $\mathcal{F}_0$ contains all $\mathbb{P}$-null sets (completeness)

Definition 2.1 (Cadlag Function)**.** A function $f : [0, \infty) \to \mathbb{R}$ is cadlag if at every point $t \geq 0$:

1. $\lim_{s\to t^-} f(s)$ exists (left limits exist)

2. $\lim_{s\to t^+} f(s) = f(t)$ (right-continuous)

The space of cadlag functions, denoted $\mathbb{D}[0,\infty)$, plays a crucial role in the study of Lévy processes. Unlike continuous functions, cadlag functions can have countably many jumps while maintaining enough regularity for mathematical analysis. Cadlag is an acronym for the French “continu à droite, limites à gauche”, meaning continuous to the right and has a limit to the left. The space of these functions is named Skorokhod space, after A.V. Skorokhod who showed the space was separable and complete [11]. The following proof of completeness is adapted from Billingsley's *Convergence of Probability Measures* [**?**].

Theorem 1 (Skorokhod Space)**.** The space $\mathbb{D}[0,\infty)$ equipped with the Skorokhod topology is complete.

Proof. We prove that $\mathbb{D}[0,\infty)$, the space of right-continuous functions with left limits, equipped with the Skorokhod metric d_J, is a complete metric space. The Skorokhod metric is defined as:

$$d_J(x,y) = \inf_{\lambda\in\Lambda}\left\{\sup_{t\geq 0}|\lambda(t)-t| \vee \sup_{t\geq 0}|x(\lambda(t))-y(t)|\right\},$$

where Λ is the set of all strictly increasing, continuous bijections $\lambda : [0,\infty)\to[0,\infty)$ with $\lambda(0)=0$.

Let $(x^{(n)})_{n=1}^{\infty}$ be a Cauchy sequence in $(\mathbb{D}[0,\infty), d_J)$. By definition, for any $\varepsilon > 0$, there exists N such that for all $m,n\geq N$, we have $d_J(x^{(m)},x^{(n)}) < \varepsilon$. This implies that for each pair (m,n) with $m,n\geq N$, there exists $\lambda_{m,n}\in\Lambda$ satisfying

$$\sup_{t\geq 0}|\lambda_{m,n}(t)-t| < \varepsilon \quad\text{and}\quad \sup_{t\geq 0}|x^{(m)}(\lambda_{m,n}(t)) - x^{(n)}(t)| < \varepsilon.$$

To show completeness, we must extract a subsequence that converges to a limit in $\mathbb{D}[0,\infty)$. To do so, fix an element of the sequence, say $x^{(N)}$, as a reference. For each $n\geq N$, since $(x^{(n)})$ is a Cauchy sequence, we have $d_J(x^{(n)},x^{(N)}) < \varepsilon$. Thus, there exists $\lambda_n\in\Lambda$ such that

$$\sup_{t\geq 0}|\lambda_n(t)-t| < \varepsilon \quad\text{and}\quad \sup_{t\geq 0}|x^{(n)}(\lambda_n(t)) - x^{(N)}(t)| < \varepsilon,$$

where $\lambda_n := \lambda_{n,N}$ emphasizes that λ_n aligns $x^{(n)}$ with $x^{(N)}$.

Since $\lambda_n(0) = 0$, λ_n is strictly increasing, and continuous, and $\sup_{t\in[0,T]} |\lambda_n(t) - t| < \varepsilon$ for all $T > 0$, the family (λ_n) is equicontinuous and uniformly bounded on compacts.

By the Arzela–Ascoli theorem (see Remark 1), there exists a subsequence (λ_{n_k}) and a limiting continuous, strictly increasing function $\lambda : [0, \infty) \to [0, \infty)$ such that $\lambda_{n_k} \to \lambda$ uniformly on every compact interval. Furthermore, because each λ_{n_k} satisfies $\sup_{t\in[0,T]} |\lambda_{n_k}(t) - t| < \varepsilon$, taking the limit as $k \to \infty$ yields $\sup_{t\in[0,T]} |\lambda(t) - t| \leq \varepsilon$, and since $\varepsilon > 0$ was arbitrary, it follows that $\lambda(t) = t$ for all $t \geq 0$. Thus, we have constructed a subsequence (λ_{n_k}) that converges uniformly on compacts to the identity map.

Now, because $\lambda_{n_k}(t) \to t$ uniformly on compacts, and because

$$\sup_{t\geq 0} |x^{(n_k)}(\lambda_{n_k}(t)) - x^{(N)}(t)| < \varepsilon,$$

we have that, for each fixed $T > 0$,

$$\sup_{t\in[0,T]} |x^{(n_k)}(\lambda_{n_k}(t)) - x^{(N)}(t)| < \varepsilon.$$

As $k \to \infty$, we see that $\lambda_{n_k}(t) \to t$ uniformly on $[0, T]$, and because each $x^{(n_k)}$ and $x^{(N)}$ is right-continuous, we can pass to the limit inside the supremum, and we see that:

$$\lim_{k\to\infty} \sup_{t\in[0,T]} |x^{(n_k)}(\lambda_{n_k}(t)) - x^{(N)}(t)| \leq \varepsilon.$$

Since ε was chosen based on the Cauchy property, and can be made arbitrarily small, we have that

$$\sup_{t\in[0,T]} |x^{(n_k)}(\lambda_{n_k}(t)) - x^{(N)}(t)| \to 0 \text{ as } k \to \infty.$$

We have shown that $x^{(n_k)}(\lambda_{n_k}(t))$ converges uniformly on compacts to $x^{(N)}(t)$. Since $\lambda_{n_k}(t) \to t$ uniformly on compacts, and because $x^{(n_k)}$ are cadlag, we have that

$$x^{(n_k)}(t) - x^{(n_k)}(\lambda_{n_k}(t)) \to 0 \text{ uniformly on compacts,}$$

because $\lambda_{n_k}(t)$ differs from t by at most ε, and cadlag functions are uniformly continuous on compact sets. More precisely, fix $T > 0$. For large k, we have that $\sup_{t\in[0,T]} |t - \lambda_{n_k}(t)| < \varepsilon$. By the right-continuity of $x^{(n_k)}$ and the fact that cadlag functions have bounded variation on compacts, we can control $|x^{(n_k)}(t) - x^{(n_k)}(\lambda_{n_k}(t))|$, and for sufficiently large k, this difference can be made arbitrarily small. Thus:

$$\sup_{t\in[0,T]} |x^{(n_k)}(t) - x^{(N)}(t)| \to 0$$

as $k \to \infty$.

We have shown that the subsequence $x^{(n_k)}$ converges uniformly on compacts to the cadlag function $x^{(N)}$, and since $x^{(N)} \in \mathbb{D}[0,\infty)$, this limit is in $\mathbb{D}[0,\infty)$. Thus, we have shown that every Cauchy sequence in $\mathbb{D}[0,\infty)$ under the Skorokhod metric converges to a limit in $\mathbb{D}[0,\infty)$, and therefore the Skorokhod space is complete. □

2.2 Stochastic Processes and Stopping Times

A stochastic process is formally defined as a collection of random variables indexed by time, $\{X_t : t \in T\}$, where t represents time and X_t denotes the state of the system at time t. Each X_t is defined on a probability space $(\Omega, \mathcal{F}, P)$, where:

$$X_t : \Omega \to \mathbb{R}, \quad \forall t \in T,$$

and T (the index set) typically represents either discrete time ($\mathbb{Z}$) or continuous time ($\mathbb{R}^+$).

Definition 2.2 (Adapted Process)**.** A stochastic process $(X_t)_{t\geq 0}$ is adapted to filtration $(\mathcal{F}_t)_{t\geq 0}$ if X_t is $\mathcal{F}_t$-measurable for each $t \geq 0$.

Definition 2.3 (Stopping Time)**.** A random variable $\tau : \Omega \to [0,\infty]$ is a stopping time if $\{\tau \leq t\} \in \mathcal{F}_t$ for all $t \geq 0$.

Stopping times are essential for analyzing the jump behavior of Lévy processes. They allow us to precisely define when jumps occur and study their properties.

2.3 Jump Processes and Random Measures

For a cadlag process $(X_t)_{t\geq 0}$, we define its jump at time t as:

$$\Delta X_t = X_t - X_{t^-}$$

Definition 2.4 (Random Measure)**.** A random measure μ on $[0, \infty) \times \mathbb{R}$ is a family of measures $\mu(\omega, \cdot)$ such that $\omega \mapsto \mu(\omega, A)$ is measurable for all Borel sets A.

The jump measure of a cadlag process X is defined as:

$$\mu_X((0, t] \times A) = \#\{s \in (0, t] : \Delta X_s \in A\}$$

2.4 Lévy Processes: Definition and Basic Properties

We can now formally define Lévy processes:

Definition 2.5 (Lévy Process)**.** An adapted cadlag process $(L_t)_{t \geq 0}$ is a Lévy process if:

1. $L_0 = 0$ a.s.
2. For any $0 \leq s < t$, $L_t - L_s$ is independent of $\mathcal{F}_s$
3. For any $0 \leq s < t$, $L_t - L_s \overset{d}{=} L_{t-s}$

A fundamental result about Lévy processes is their path decomposition:

Theorem 2 (Lévy-Itô Decomposition)**.** Every Lévy process $(L_t)_{t \geq 0}$ can be uniquely decomposed as:

$$L_t = \gamma t + \sigma B_t + \int_{|x| \leq 1} x(\mu - \nu)(dx, dt) + \int_{|x|>1} x\mu(dx, dt)$$

where:

1. $\gamma \in \mathbb{R}$ is the drift
2. $\sigma \geq 0$ and B_t is standard Brownian motion
3. μ is the jump measure of L
4. $\nu(dx)dt$ is the "compensator" of μ

The compensator ν is a deterministic measure that characterizes the intensity of jumps, while μ is a random measure counting actual jumps. The difference $\mu - \nu$ essentially centers the jump measure. This decomposition is crucial for understanding how Lévy processes combine continuous and jump behavior.

Here are some key theorems and proofs that build on our foundation and will be crucial for neural network approximation later:

2.5 Characteristic Functions and the Lévy-Khintchine Formula

Theorem 3 (Lévy-Khintchine Representation)**.** Let $(L_t)_{t\geq 0}$ be a Lévy process. Then its characteristic function has the form:

$$\mathbb{E}[e^{iuL_t}] = e^{t\psi(u)}$$

where

$$\psi(u) = i\gamma u - \frac{\sigma^2}{2}u^2 + \int_{\mathbb{R}\setminus\{0\}} (e^{iux} - 1 - iux\mathbf{1}_{\{|x|<1\}})\nu(dx)$$

and ν is a measure on $\mathbb{R} \setminus \{0\}$ satisfying

$$\int_{\mathbb{R}\setminus\{0\}} (1 \wedge x^2)\nu(dx) < \infty$$

Proof. We begin by establishing that L_t is infinitely divisible, a fundamental property for deriving the characteristic function's form. For any $n \in \mathbb{N}$, we can partition the interval $[0, t]$ into n equal subintervals:

$$0 = t_0 < t_1 < \cdots < t_n = t, \text{ where } t_k = \frac{k}{n}t.$$

Define the increments $\Delta L_k = L_{t_k} - L_{t_{k-1}}$ for $k = 1, \dots, n$. Due to the properties of a Lévy process, the increments $(\Delta L_k)_{k=1}^n$ are independent, and $\Delta L_k \overset{d}{=} L_{t_k - t_{k-1}} \overset{d}{=} L_{t/n}$ are identically distributed. We can then express L_t as:

$$L_t = L_{t_n} - L_{t_0} = \sum_{k=1}^{n}(L_{t_k} - L_{t_{k-1}}) = \sum_{k=1}^{n} \Delta L_k,$$

thus showing that L_t is infinitely divisible.

Let $\phi_t(u) = \mathbb{E}[e^{iuL_t}]$ denote the characteristic function of L_t. Since L_t is infinitely divisible, $\phi_t(u)$ does not vanish. Exploiting the independence and stationarity of increments in Lévy processes, we have:

$$\phi_t(u) = \mathbb{E}[e^{iuL_t}] = \mathbb{E}[e^{iuL_{t/n} + iu(L_{2t/n} - L_{t/n}) + \cdots + iu(L_t - L_{(n-1)t/n})}] = \left(\mathbb{E}[e^{iuL_{t/n}}]\right)^n,$$

which implies

$$\phi_{t/n}(u) = \phi_t(u)^{1/n}.$$

Fixing $t = 1$ and defining $\psi(u) = \log(\phi_1(u))$, we see that since $\phi_t(u)$ does not vanish, its logarithm is well-defined, and we have:

$$\phi_t(u) = \phi_1(u)^t = \left(e^{\log(\phi_1(u))}\right)^t = e^{t\psi(u)}.$$

The function $\psi(u)$ is called the Lévy symbol.

Now, using the Lévy-Itô decomposition theorem, we can write any Lévy process L_t uniquely as:

$$L_t = \gamma t + \sigma B_t + \int_{|x|\leq 1} x(\mu - \nu)(dx, dt) + \int_{|x|>1} x\mu(dx, dt),$$

where:

- $\gamma \in \mathbb{R}$ is the drift;
- $\sigma \geq 0$ and B_t is standard Brownian motion;
- μ is the jump measure;
- ν is the compensator of μ, satisfying $\int_{\mathbb{R}\setminus\{0\}}(1 \wedge x^2)\nu(dx) < \infty$.

Since the components of the Lévy-Itô decomposition are also Lévy processes and are independent, the characteristic function of their sum is the product of their individual characteristic functions:

$$\phi_t(u) = \mathbb{E}[e^{iuL_t}] = \mathbb{E}[e^{iu\gamma t}]\mathbb{E}[e^{iu\sigma B_t}]\mathbb{E}[e^{iu\int_{|x|\leq 1} x(\mu-\nu)(dx,dt)}]\mathbb{E}[e^{iu\int_{|x|>1} x\mu(dx,dt)}].$$

We compute each of these characteristic functions separately.

1. The characteristic function of the drift term γt is:

$$\mathbb{E}[e^{iu\gamma t}] = e^{iu\gamma t}.$$

2. The characteristic function of σB_t is:

$$\mathbb{E}[e^{iu\sigma B_t}] = e^{-\frac{1}{2}\sigma^2 u^2 t}.$$

3. For the compensated small jump integral, we have:

$$\mathbb{E}\left[e^{iu\int_{|x|\le 1} x(\mu-\nu)(dx,dt)}\right] = \exp\left(t\int_{|x|\le 1}(e^{iux}-1-iux)\nu(dx)\right).$$

4. For the large jump integral, we have:

$$\mathbb{E}\left[e^{iu\int_{|x|>1} x\mu(dx,dt)}\right] = \exp\left(t\int_{|x|>1}(e^{iux}-1)\nu(dx)\right).$$

In the interest of brevity we do not provide the full proof for these, which can be found in [18] Combining these results, and using the fact that $\phi_t(u) = e^{t\psi(u)}$, we obtain:

$$\begin{aligned}
\psi(u) &= \frac{1}{t}\log\left(\mathbb{E}[e^{iuL_t}]\right)\\
&= \frac{1}{t}\log\left(e^{iu\gamma t}e^{-\frac{1}{2}\sigma^2u^2t}\exp\left(t\int_{|x|\le 1}(e^{iux}-1-iux)\nu(dx)\right)\exp\left(t\int_{|x|>1}(e^{iux}-1)\nu(dx)\right)\right)\\
&= iu\gamma - \frac{1}{2}\sigma^2u^2 + \int_{|x|\le 1}(e^{iux}-1-iux)\nu(dx) + \int_{|x|>1}(e^{iux}-1)\nu(dx)\\
&= i\gamma u - \frac{\sigma^2}{2}u^2 + \int_{\mathbb{R}\setminus\{0\}}(e^{iux}-1-iux\mathbf{1}_{\{|x|<1\}})\nu(dx).
\end{aligned}$$

Thus, we arrive at the Lévy-Khintchine representation for the Lévy symbol:

$$\psi(u) = i\gamma u - \frac{\sigma^2}{2}u^2 + \int_{\mathbb{R}\setminus\{0\}}(e^{iux}-1-iux\mathbf{1}_{\{|x|<1\}})\nu(dx),$$

where the Lévy measure ν satisfies

$$\int_{\mathbb{R}\setminus\{0\}}(1\wedge x^2)\nu(dx) < \infty.$$

□

2.6 Generator and Semigroup Properties

Theorem 4 (Infinitesimal Generator)**.** For $f \in C_0^2(\mathbb{R})$, the infinitesimal generator $\mathcal{A}$ of a Lévy process is:

$$\mathcal{A}f(x) = \gamma f'(x) + \frac{\sigma^2}{2}f''(x) + \int_{\mathbb{R}\setminus\{0\}}[f(x+y) - f(x) - yf'(x)\mathbf{1}_{\{|y|<1\}}]\nu(dy)$$

Proof. Let $(L_t)_{t\geq 0}$ be a Lévy process and $f \in C_0^2(\mathbb{R})$. The infinitesimal generator $\mathcal{A}$ is defined as:

$$\mathcal{A}f(x) = \lim_{t\downarrow 0} \frac{\mathbb{E}[f(x+L_t)] - f(x)}{t}$$

Using the Lévy-Khintchine formula, we know that $\mathbb{E}[e^{iuL_t}] = e^{t\psi(u)}$, where

$$\psi(u) = i\gamma u - \frac{\sigma^2}{2}u^2 + \int_{\mathbb{R}\setminus\{0\}} (e^{iux} - 1 - iux\mathbf{1}_{\{|x|<1\}})\nu(dx).$$

By the inverse Fourier transform, we have

$$f(x+L_t) = \frac{1}{2\pi}\int_{\mathbb{R}} e^{-iu(x+L_t)}\hat{f}(u)du$$

where $\hat{f}(u)$ is the Fourier transform of f. Taking the expectation and plugging in the Lévy-Khintchine formula yields

$$\mathbb{E}[f(x+L_t)] = \frac{1}{2\pi}\int_{\mathbb{R}} e^{-iux}e^{t\psi(-u)}\hat{f}(u)du.$$

Then:

$$\frac{\mathbb{E}[f(x+L_t)] - f(x)}{t} = \frac{1}{2\pi t}\int_{\mathbb{R}} e^{-iux}(e^{t\psi(-u)} - 1)\hat{f}(u)du$$

Taking the limit as $t \to 0$ and using the fact that since ψ is continuous at u, $\lim_{t\downarrow 0} \frac{e^{t\psi(-u)}-1}{t} = \psi(-u)$ gives

$$\mathcal{A}f(x) = \lim_{t\downarrow 0} \frac{\mathbb{E}[f(x+L_t)] - f(x)}{t} = \frac{1}{2\pi}\int_{\mathbb{R}} e^{-iux}\psi(-u)\hat{f}(u)du.$$

Plugging in the expression for $\psi(u)$, and using the properties of the inverse Fourier transform (i.e. the fact that $-iu$ turns into a derivative with respect to position and $-u^2$ turns into a second derivative; see Remark 2) :

$$\begin{aligned}
\mathcal{A}f(x) &= \frac{1}{2\pi}\int_{\mathbb{R}} e^{-iux}\left(-i\gamma u - \frac{\sigma^2}{2}u^2 + \int_{\mathbb{R}\setminus\{0\}} (e^{-iuy} - 1 + iuy\mathbf{1}_{\{|y|<1\}})\nu(dy)\right)\hat{f}(u)du \\
&= \gamma f'(x) + \frac{\sigma^2}{2}f''(x) + \int_{\mathbb{R}\setminus\{0\}} \left(\frac{1}{2\pi}\int_{\mathbb{R}} e^{-iux}(e^{-iuy} - 1 + iuy\mathbf{1}_{\{|y|<1\}})\hat{f}(u)du\right)\nu(dy) \\
&= \gamma f'(x) + \frac{\sigma^2}{2}f''(x) + \int_{\mathbb{R}\setminus\{0\}} [f(x+y) - f(x) - yf'(x)\mathbf{1}_{\{|y|<1\}}]\nu(dy).
\end{aligned}$$

This we have that that the infinitesimal generator $\mathcal{A}$ of the Lévy process is:

$$\mathcal{A}f(x) = \gamma f'(x) + \frac{\sigma^2}{2} f''(x) + \int_{\mathbb{R}\setminus\{0\}} [f(x+y) - f(x) - yf'(x)\mathbf{1}_{\{|y|<1\}}]\nu(dy).$$

$\square$

2.7 Approximation Properties

This theorem will be crucial for neural network approximation, because it shows we can approximate any Lévy process by simple, piecewise constant processes - which is exactly what a neural network with finite parameters will need to do when learning to approximate a Lévy process.

Theorem 5 (Density of Simple Processes)**.** Let L_t be a Lévy process and fix $T > 0$. For any $\epsilon > 0$, there exists a simple process S_t of the form:

$$S_t = \sum_{i=1}^{n} X_i \mathbf{1}_{[t_i, t_{i+1})}(t)$$

such that

$$\mathbb{E}\left[\sup_{t\in[0,T]} |L_t - S_t|^2\right] < \epsilon$$

Theorem 6 (Density of Simple Processes)**.** Let L_t be a Lévy process and fix $T > 0$. For any $\epsilon > 0$, there exists a simple process S_t of the form:

$$S_t = \sum_{i=0}^{n-1} X_i \mathbf{1}_{[t_i, t_{i+1})}(t)$$

such that

$$\mathbb{E}\left[\sup_{t\in[0,T]} |L_t - S_t|^2\right] < \epsilon$$

Proof. By the Lévy-Itô decomposition, we have

$$L_t = \gamma t + \sigma B_t + \int_{|x|\le 1} x(\mu - \nu)(dx, dt) + \int_{|x|>1} x\mu(dx, dt)$$

Let $L_t^{(c)} = \gamma t + \sigma B_t$ be the continuous part and $L_t^{(j)} = \int_{|x|\le 1} x(\mu-\nu)(dx, dt) + \int_{|x|>1} x\mu(dx, dt)$ be the jump part.

Fix $\delta > 0$ and partition the interval $[0,T]$ into n equal subintervals $0 = t_0 < t_1 < \cdots < t_n = T$, with $\delta = \frac{T}{n}$. Construct a simple process S_t as:

$$S_t = \sum_{i=0}^{n-1} (L_{t_i}^{(c)} + L_{t_i}^{(j)}) \mathbf{1}_{[t_i, t_{i+1})}(t).$$

Then $L_t - S_t = (L_t^{(c)} - S_t^{(c)}) + (L_t^{(j)} - S_t^{(j)})$. Now, by the fact that $(a+b)^2 \leq 2a^2 + 2b^2$:

$$\mathbb{E}\left[\sup_{t\in[0,T]} |L_t - S_t|^2\right] \leq 2\mathbb{E}\left[\sup_{t\in[0,T]} |L_t^{(c)} - S_t^{(c)}|^2\right] + 2\mathbb{E}\left[\sup_{t\in[0,T]} |L_t^{(j)} - S_t^{(j)}|^2\right]$$

Let's bound each term:

First, the continuous part:

Define $S_t^{(c)} = \sum_{i=0}^{n-1} L_{t_i}^{(c)} \mathbf{1}_{[t_i, t_{i+1})}(t)$. Then, for $t \in [t_i, t_{i+1})$:

$$|L_t^{(c)} - S_t^{(c)}| = |\gamma(t - t_i) + \sigma(B_t - B_{t_i})| \leq |\gamma||t - t_i| + |\sigma||B_t - B_{t_i}| \leq \gamma\delta + \sigma \sup_{t\in[t_i, t_{i+1})} |B_t - B_{t_i}|$$

Therefore:

$$\sup_{t\in[0,T]} |L_t^{(c)} - S_t^{(c)}| \leq \gamma\delta + \sigma \sup_{t\in[0,T]} |B_t - B_{t_{\lfloor t/\delta \rfloor}}|.$$

Using the maximal inequality for Brownian motion[18], we have that

$$\mathbb{E}\left[\sup_{t\in[0,T]} |B_t - B_{t_{\lfloor t/\delta \rfloor}}|^2\right] \leq C_1 \delta$$

for some constant C_1 depending on T. This gives

$$\mathbb{E}\left[\sup_{t\in[0,T]} |L_t^{(c)} - S_t^{(c)}|^2\right] \leq 2\gamma^2\delta^2 + 2\sigma^2 C_1 \delta.$$

Let us now bound the jump part:

Define $S_t^{(j)} = \sum_{i=0}^{n-1} L_{t_i}^{(j)} \mathbf{1}_{[t_i, t_{i+1})}(t)$. Then for $t \in [t_i, t_{i+1})$ we have:

$$\mathbb{E}[|L_t^{(j)} - L_{t_i}^{(j)}|^2] \leq C_2(t - t_i) \leq C_2 \delta$$

for some constant C_2 which depends on ν and T. Using the fact that for a positive random variable, X, we have that $\mathbb{E}[\sup_t X_t^2] \leq \sup_t \mathbb{E}[X_t^2]$, we have that

$$\mathbb{E}\left[\sup_{t\in[0,T]} |L_t^{(j)} - S_t^{(j)}|^2\right] \leq C_2\delta$$

Combining our bounds:

$$\mathbb{E}\left[\sup_{t\in[0,T]} |L_t - S_t|^2\right] \leq 4\gamma^2\delta^2 + (4\sigma^2 C_1 + 2C_2)\delta$$

By choosing δ sufficiently small (i.e., taking n to be sufficiently large), we can ensure that the error is less than ϵ. □

In the next section, we'll use these results to show neural network approximations exist that preserve the essential properties of Lévy processes, while being significantly simpler computationally.

3 Neural Network Approximation of Lévy Processes

3.1 Brief Introduction to Universal Approximation with Neural Networks

Neural networks are a class of function approximators composed of layers of interconnected nodes (or "neurons"). At their core, neural networks perform a repeated sequence of linear transformations followed by element-wise nonlinearities. Each layer performs the operation

$$a(Wx + b),$$

where x represents the input to the layer (which might be the input to the network, or the output from a previous layer), W is a matrix of weights, b is a vector of biases, and a is an element-wise nonlinear activation function. Common activation functions include the sigmoid, ReLU (Rectified Linear Unit), tanh, and softplus functions.

These transformations are repeated across multiple layers, with the output of each layer becoming the input for the subsequent layer. The final layer produces the output of the neural network. The weights W and biases b (across all layers) are the trainable parameters of the network, and are typically learned by minimizing a loss function using some form of stochastic gradient descent.

The Universal Approximation Theorem states that, under suitable conditions, a neural network can approximate a wide range of functions to arbitrary precision, given sufficient network complexity and appropriate training. Further details on neural network architectures and training can be found in [20].

3.2 Approximating Lévy Processes

The approximation of Lévy processes presents unique challenges due to their hybrid nature - combining continuous paths with discontinuous jumps. Traditional numerical methods often struggle to capture both components efficiently. Neural networks, with their universal approximation properties, offer a promising framework for modeling such processes. In this section, we develop a principled approach to approximating Lévy processes using Neural Jump Stochastic Differential Equations (Neural JSDEs). Neural networks are known for their universal approximation property: roughly, given a sufficiently large (and appropriately structured) neural network, one can approximate a wide class of functions arbitrarily well. This includes continuous functions on compact sets and, under additional conditions, functions arising in stochastic dynamics.

3.3 Neural Jump Stochastic Differential Equations

Following the work of Jia and Benson[12], we can represent a Lévy process using a Neural JSDE framework. This approach is particularly powerful because it naturally separates the continuous and jump components while maintaining the mathematical properties essential to Lévy processes.

Definition 3.1 (Neural Jump SDE)**.** A Neural Jump SDE represents a stochastic process through a latent state $\mathbf{z}(t) \in \mathbb{R}^n$ evolving according to:

$$d\mathbf{z}(t) = \mathbf{f}(\mathbf{z}(t), t; \boldsymbol{\theta})dt + \mathbf{w}(\mathbf{z}(t), \mathbf{k}(t), t; \boldsymbol{\theta})dN(t)$$

where $\mathbf{f}$ and $\mathbf{w}$ are neural networks, $N(t)$ is a counting process for jumps, and $\boldsymbol{\theta}$ represents the network parameters.

The key insight here is that we can decompose the evolution of the process into two distinct components:

1. A continuous drift term $\mathbf{f}$ that captures the smooth evolution of the process
2. A jump term $\mathbf{w}$ that handles discontinuous changes

This decomposition mirrors the Lévy-Itô decomposition we established in Section 2, providing a natural bridge between classical theory and neural approximation.

3.4 Neural Architecture Design

While standard neural networks can indeed model any function $f : \mathbb{R} \to \mathbb{R}$ to arbitrary precision by the Universal Approximation Theorem, the search space for parameters θ is often too large to be computationally tractable. Therefore, architecture of the neural networks must be carefully designed to preserve the essential properties of Lévy processes (also known as an inductive bias). Following Jia and Benson, we partition the latent state into two components [12]:

$$\mathbf{z}(t) = [\mathbf{c}(t), \mathbf{h}(t)]$$

The internal state $\mathbf{c}(t) \in \mathbb{R}^{n_1}$ handles the continuous dynamics, while the jump memory $\mathbf{h}(t) \in \mathbb{R}^{n_2}$ tracks the discontinuous components. This separation is important for two reasons:

1. It allows the network to learn different patterns for continuous and jump behavior
2. It maintains the independence of increments characteristic of Lévy processes

The continuous flow function $\mathbf{f}$ is decomposed as:

$$\mathbf{f}(\mathbf{c}(t)) = \mathbf{f}_1(\mathbf{c}(t)) - \langle \mathbf{f}_1(\mathbf{c}(t)), \mathbf{c}(t) \rangle \mathbf{c}(t)$$

where $\mathbf{f}_1$ is parameterized by a neural network. The orthogonality constraint (second term) ensures stability by constraining the dynamics to a sphere, inherently bounding the latent state. Additionally, we can see that $\frac{d}{dt}||c(t)||^2 = \frac{d}{dt}(c(t) \cdot c(t)) = 2c(t) \cdot \frac{dc(t)}{dt} = 0$. Since the derivative is zero, the norm of $c(t)$ remains constant over time. This preservation of the norm further contributes to the stability of the solution.

3.5 Jump Handling and Intensity Modeling

The jump component requires particular attention. The jump function $\mathbf{w}$ is defined as:

$$\mathbf{w}(\mathbf{c}(t), \mathbf{k}(t)) = [0, \Delta\mathbf{h}(t)]$$

where $\Delta\mathbf{h}(t)$ is parameterized by another neural network and $\mathbf{k}(t)$ represents the jump characteristics. The probability of a jump occurring in a small interval $[t, t+dt)$ is given by:

$$\mathbb{P}\{\text{jump in } [t, t+dt)|\mathcal{F}_t\} = \lambda(\mathbf{z}(t))dt$$

This formulation allows us to model both he timing of jumps through the intensity function λ, and the size of jumps through the jump function $\mathbf{w}$

3.6 Training and Convergence Properties

The training objective combines maximum likelihood estimation with path-wise optimization:

$$\mathcal{L} = -\sum_j \log \lambda(\mathbf{z}(\tau_j)) - \sum_j \log p(\mathbf{k}_j|\mathbf{z}(\tau_j)) + \int_{t_0}^{t_N} \lambda(\mathbf{z}(t))dt$$

This loss function has several important properties:

1. It penalizes both missed jumps and false predictions
2. It accounts for the continuous evolution between jumps
3. It maintains the temporal ordering of events

Indeed, we can rigorously prove that the neural networks $\mathbf{f}$ and $\mathbf{w}$ converge to any target Lévy process. To prove this, we must first understand the terms *network capacity* and *Universal approximation.*

The term "network capacity" refers to the ability of a neural network to approximate a wide class of functions, which is determined by factors such as the number of layers, the number of neurons per layer, and the complexity of the activation functions. Intuitively, a network with higher capacity has more parameters and can therefore learn more complex mappings.

The universal approximation theorem for neural networks state that, given any continuous function on a compact set and any desired accuracy $\delta > 0$, one can construct a neural network with sufficient capacity that can approximate that function to within an error δ. In the case of Neural Jump SDEs, this implies that by increasing the network capacity (number of layers in the neural networks that parameterize $\mathbf{f}$ and $\mathbf{w}$), we can make the model output $\hat{L}_t$ arbitrarily close to any simple process S_t. Proof for the universal approximation theorem can be found in [19] (more accessible variants in [20].)

With this background, we will now prove the convergence guarantee theorem.

Theorem 7 (Convergence Guarantee)**.** Under appropriate regularity conditions on the neural networks $\mathbf{f}$ and $\mathbf{w}$, the trained Neural Jump SDE converges to the target Lévy process in the following sense:

$$\sup_{t\in[0,T]} \mathbb{E}[|L_t - \hat{L}_t|^2] \to 0$$

as the network capacity increases.

Proof. Our goal is to demonstrate that as the "network capacity" of the Neural Jump SDE (denoted $\hat{L}_t$) increases, it converges in the mean-square sense, uniformly on the interval $[0, T]$, to the target Lévy process L_t. That is,

$$\sup_{t\in[0,T]} \mathbb{E}[|L_t - \hat{L}_t|^2] \to 0$$

Fix $\epsilon > 0$. By the Density of Simple Processes Theorem (Theorem 5), there exists a simple process S_t of the form

$$S_t = \sum_{i=0}^{n-1} X_i \mathbf{1}_{[t_i,t_{i+1})}(t),$$

with $0 = t_0 < t_1 < \cdots < t_n = T$, such that

$$\mathbb{E}\left[\sup_{t\in[0,T]} |L_t - S_t|^2\right] < \frac{\epsilon}{4}.$$

Since simple processes can be approximated arbitrarily closely by neural networks (universal approximation arguments ensure that neural ODE components can capture piecewise-constant segments, and neural jump mechanisms can approximate the jumps in distribution and timing), we can find a Neural Jump SDE $\hat{L}_t$ for which

$$\mathbb{E}\left[\sup_{t\in[0,T]} |S_t - \hat{L}_t|^2\right] < \frac{\epsilon}{4}.$$

To relate $\hat{L}_t$ to L_t, observe that:

$$L_t - \hat{L}_t = (L_t - S_t) + (S_t - \hat{L}_t).$$

Applying the inequality $|X+Y|^2 \le 2|X|^2 + 2|Y|^2$ (a consequence of the standard triangle inequality and the elementary inequality $2|XY| \le |X|^2 + |Y|^2$), we get:

$$|L_t - \hat{L}_t|^2 \le 2|L_t - S_t|^2 + 2|S_t - \hat{L}_t|^2.$$

Taking the supremum in t and then the expectation, we have:

$$\mathbb{E}\left[\sup_{t\in[0,T]} |L_t - \hat{L}_t|^2\right] \le 2\,\mathbb{E}\left[\sup_{t\in[0,T]} |L_t - S_t|^2\right] + 2\,\mathbb{E}\left[\sup_{t\in[0,T]} |S_t - \hat{L}_t|^2\right].$$

Substituting the known bounds:

$$\mathbb{E}\left[\sup_{t\in[0,T]} |L_t - S_t|^2\right] < \frac{\epsilon}{4} \quad \text{and} \quad \mathbb{E}\left[\sup_{t\in[0,T]} |S_t - \hat{L}_t|^2\right] < \frac{\epsilon}{4},$$

we obtain:

$$\mathbb{E}\left[\sup_{t\in[0,T]} |L_t - \hat{L}_t|^2\right] < 2\cdot\frac{\epsilon}{4} + 2\cdot\frac{\epsilon}{4} = \epsilon.$$

Since $\epsilon > 0$ was arbitrary, this shows that by increasing the network capacity (and thus improving the approximation quality), the error can be made arbitrarily small. This establishes that

$$\sup_{t\in[0,T]} \mathbb{E}[|L_t - \hat{L}_t|^2] \to 0.$$

Under suitable regularity conditions (which include appropriate choice of loss function) ensuring that both the Lévy process and the Neural Jump SDE are well-defined and that universal approximation theorems apply, the convergence claim holds.

$\square$

3.7 Generator Approximation via Neural Networks

Neural JSDEs can also be understood through their approximation of the infinitesimal generator of Lévy processes. This perspective provides both theoretical justification for our architecture and practical guidance for implementation.

3.7.1 Decomposition of the Generator

Recall that the generator $\mathcal{A}$ of a Lévy process acts on $f \in C_0^2(\mathbb{R})$ as:

$$\mathcal{A}f(x) = \underbrace{\gamma f'(x) + \frac{\sigma^2}{2} f''(x)}_{\text{continuous part}} + \underbrace{\int_{\mathbb{R}\setminus\{0\}} [f(x+y) - f(x) - yf'(x)\mathbf{1}_{\{|y|<1\}}]\nu(dy)}_{\text{jump part}}$$

The Neural JSDE architecture mirrors this decomposition through: 1. The continuous network $\mathbf{f}$ approximating the differential operator 2. The jump network $\mathbf{w}$ approximating the integral operator

3.7.2 Neural Approximation of the Generator

Let $\hat{\mathcal{A}}$ denote our neural approximation to the generator. We construct it as:

$$\hat{\mathcal{A}}f(x) = \hat{\mathcal{A}}_{\text{cont}} f(x) + \hat{\mathcal{A}}_{\text{jump}} f(x)$$

where:

$$\hat{\mathcal{A}}_{\text{cont}} f(x) = \langle \nabla f(x), \mathbf{f}(x, \theta) \rangle$$
$$\hat{\mathcal{A}}_{\text{jump}} f(x) = \lambda(x) \mathbb{E}_{\mathbf{k} \sim p(\cdot|x)} [f(x + \mathbf{w}(x, \mathbf{k})) - f(x)]$$

3.7.3 Architectural Implications

This generator perspective motivates several architectural choices:

1. **Orthogonal Flows:** The constraint

$$\mathbf{f}(\mathbf{c}(t)) = \mathbf{f}_1(\mathbf{c}(t)) - \langle \mathbf{f}_1(\mathbf{c}(t)), \mathbf{c}(t) \rangle \mathbf{c}(t)$$

ensures the continuous part preserves the geometry of the generator's differential operator.

2. **Jump Structure:** The decomposition

$$\mathbf{w}(\mathbf{z}(t), \mathbf{k}(t)) = [0, \Delta\mathbf{h}(t)]$$

mirrors the integral structure in the generator.

3. **Intensity Network:** The choice of

$$\lambda(\mathbf{z}(t)) = \text{softplus}(W\mathbf{z}(t) + b)$$

ensures positive jump intensities, matching the measure-theoretic requirements of the generator.

3.7.4 Training Implications

The generator perspective also informs training:

1. **Loss Decomposition:** We can split the loss to reflect generator components:

$$\mathcal{L} = \mathcal{L}_{\text{cont}} + \mathcal{L}_{\text{jump}}$$

2. **Regularization:** We can add generator-based regularization:

$$\mathcal{R}(\theta) = \|\mathcal{A}f_{\text{test}} - \hat{\mathcal{A}}f_{\text{test}}\|^2$$

for test functions f_{test}.

This generator-based perspective provides a deeper understanding of why Neural JSDEs are particularly well-suited for approximating Lévy processes and guides both architectural choices and training procedures.

4 Implementation and Experimental Validation

We implemented the Neural Jump SDE framework in `PyTorch`, leveraging the `torchdiffeq` library for ODE integration. The model, with just $1,797$ trainable parameters, is composed of four primary neural components parameterizing the stochastic dynamics:

1. **ODE Function (`ODEF`):** A multi-layer perceptron that defines the continuous-time

drift and diffusion of the latent state $z(t)$. This ODE function is integrated using `torchdiffeq`'s `odeint` solver to advance the latent state between events.

2. **Jump Function (`JumpFcn`):** Another MLP that determines the jump magnitude applied to the latent state at event times. Given the current state $z(t^-)$ and a sampled jump size k, the jump function outputs the increment to be added to $z(t)$ upon an event's occurrence.

3. **Intensity Function (`IntensityFcn`):** An MLP mapping $z(t)$ to a nonnegative intensity $\lambda(z(t))$. We use an exponential transformation to ensure positivity. This intensity governs the waiting times between events, with event times sampled from a (thinned) Poisson process whose rate depends on $\lambda(z(t))$.

4. **Probability Function (`ProbabilityFcn`):** An MLP that parameterizes the conditional distribution of jump sizes given the current latent state. It outputs a mean and a variance, and from this normal distribution we sample the jump size k at event times.

All of the above multi-layer perceptrons (neural networks) had two layers, each with the ReLU (Rectified Linear Unit) nonlinearity. The *ReLU* activation function, defined as $\text{ReLU}(x) = \max(0, x)$, is a standard choice for introducing non-linearity into neural networks.

The entire Neural Jump SDE model integrates these components. We simulate trajectories by:

1. Starting from an initial state $z(0)$.

2. For each event, sampling the next event time from the intensity $\lambda(z(t))$.

3. Using `torchdiffeq.odeint` to integrate the ODE from the current time to the sampled event time.

4. At the event time, sampling a jump size k from the learned normal distribution and updating $z(t)$ via the jump function.

5. Repeating until a fixed horizon or a maximum number of events is reached.

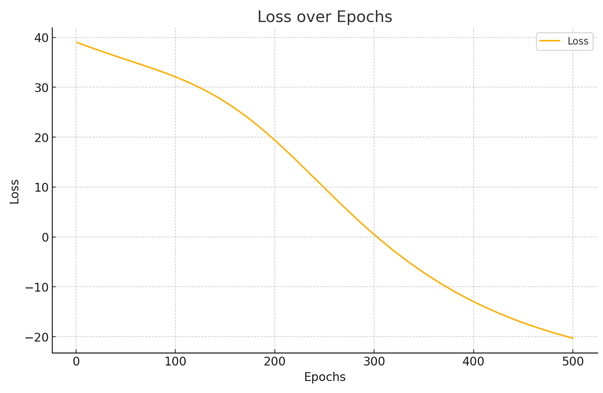

Figure 1: Loss Curve

4.1 Synthetic Data from Merton Processes

To train and test our model, we generated data from a Merton jump-diffusion model, a Lévy-type process commonly used in financial mathematics. The Merton process is given by:

$$dX_t = \gamma\,dt + \sigma\,dW_t + J\,dN_t,$$

where W_t is a Wiener process and N_t a Poisson process with intensity λ. The jump sizes J are drawn from a lognormal or normal distribution. In our implementation, we choose parameters (e.g., $\gamma = 0$, $\sigma = 1$, and a chosen λ, μ, and σ for the jump distribution) that yield a moderate number of events (between 0 and 12) over the chosen time interval $[0, T]$.

We simulate multiple trajectories from this Merton process and record event times and jump sizes. To simplify the matching of events with latent dynamics, we discretize time and snap event occurrences to the nearest time step. Although this is a simplification, it ensures that the model and the data share a common temporal grid, allowing straightforward computation of the likelihood terms without fractional time steps (see Section 6 for future work using diffusion or flow-matching models).

4.2 Training Procedure

We construct a dataset of synthetic Merton trajectories and their associated event sequences. The training objective is the negative log-likelihood of the observed events under the Neural Jump SDE model (that is, we are maximizing the likelihood of observed events given our model). Specifically, we minimize:

$$\mathcal{L} = \sum_j -\log(\lambda(z(\tau_j))) - \log p(k_j|z(\tau_j)) \ + \int_0^T \lambda(z(t))\,dt,$$

where τ_j, k_j are event times and jump sizes from the simulated data. The first two sums encourage the model to place high intensity and appropriate jump distributions at event times, while the integral term penalizes unnecessary intensity away from events.

We train the model for 500 epochs using the Adam (Adaptive Moment Estimation) optimizer - a faster variant of Stochastic Gradient Descent [15]. At each epoch, we sample mini-batches of trajectories from the dataset, compute the loss, and perform gradient-based updates on all model parameters (in the ODE function, jump function, intensity function, and probability function).

4.3 Empirical Results and Visualization

As training progresses, the model adjusts its parameters to better align the latent intensity and jump distributions with the observed events from the Merton processes. Initially, the loss is high because the model places intensity and jump distributions incorrectly. Over epochs, we observe a steady decrease in the integrated intensity and event log-likelihood terms, indicating that the model learns to increase intensity near observed event times and match the jump size distributions more closely. The model's loss at each epoch is visualized in Figure 1. We stop training when the loss starts to flatten out at around 500 epochs.

To illustrate the model's learned behavior, we:

1. Compare the intensity function $\lambda(z(t))$ with actual event occurrences on a training trajectory. We typically see the intensity spike or remain elevated around observed jump times.

2. Simulate a new "test" Merton trajectory (with a different random seed) not seen during training and compare the model's predicted events and intensity pattern to the true events. While perfect alignment is not expected or seen, the model often captures the general structure of event timing and magnitude distributions, as seen in Figure 2.

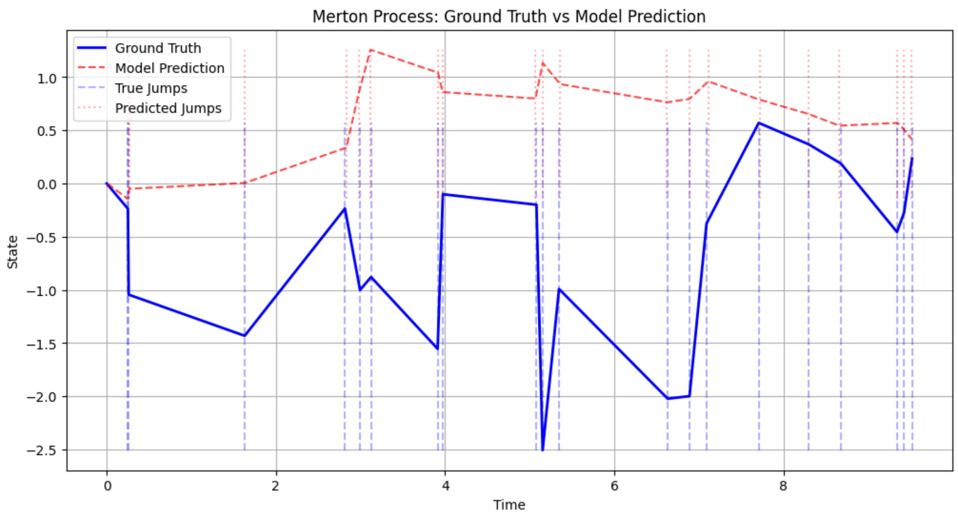

Figure 2: Model Prediction on Unseen Test Trajectory

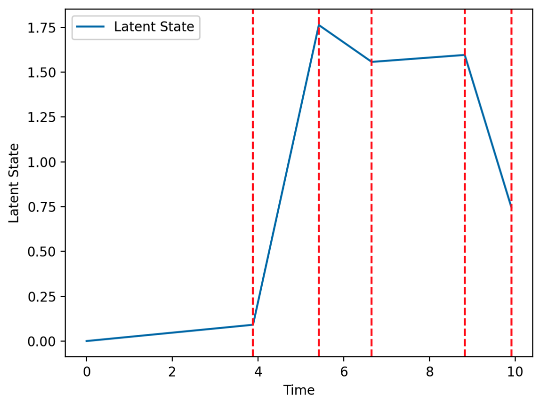

Figure 3: Latent State Over an Unseen Test Trajectory (Ground Truth Events In Red)

3. Visualize the latent state evolution (Figure 3), showing how the model inserts jumps in $h(t)$ at predicted event times and integrates the ODE between these times. This reveals that the model encodes timing and magnitude information in the latent space.

The final results show that the Neural Jump SDE is capable of learning a reasonable approximation to the underlying Merton jump structure from data alone, without prior knowledge of the parametric form of the process. The experiments validate the model's ability to represent both continuous dynamics and discrete events in a unified latent framework.

As Figures 2 and 3 show, the model's internal latent state reflects the jump event timings very accurately, while not being able to model the continuous dynamics as well. In a separate experiment, even a 100,000 parameter large multi-layer perceptron model exhibited a very high loss and testing revealed that it failed to learn the jump dynamics of such Merton processes.

The remarkable aspect of such a model is that while performing a traditional parameter search to accurately fit a Lévy process to observed data would require extensive computational resources and intricate manual calibration, the neural network-based approach can learn to replicate the timing and distribution of jumps efficiently, thereby streamlining the modeling process and offering greater flexibility.

5 Applications

Lévy processes have found significant applications in financial modeling. This section surveys key applications and empirical findings from the literature, focusing on high-frequency trading and market microstructure.

5.1 High-Frequency Market Dynamics

Recent studies have demonstrated that market microstructure at high frequencies exhibits behavior better captured by Lévy processes than traditional Brownian motion models. Cont et al (2004) [5] provide extensive evidence that:

1. Price jumps occur more frequently than Gaussian models suggest

2. Returns exhibit heavy tails even at very short time scales
3. Volatility clustering persists at the microstructure level

Their model demonstrates superior fit to high-frequency data compared to traditional stochastic volatility models or simpler multi-layer perceptron neural networks.

5.2 Order Book Dynamics

The limit order book provides a natural application for Lévy process models with neural network approximations. Several studies have shown promising results:

1. **Price Formation:** Cont et al. (2004)[5] demonstrate that order flow can be modeled as a marked point process, naturally fitting within the Neural JSDE framework.
2. **Market Impact:** Large trades create price impacts that exhibit jump-diffusion characteristics, as documented by Almgren and Chriss (2001)[13] and later modeled using neural jump processes.
3. **Liquidity Dynamics:** Work by Cartea et al. (2009)[14] shows that liquidity provision follows patterns well-captured by Lévy processes with state-dependent intensities.

6 Future Work

While Neural Jump Stochastic Differential Equations provide a flexible and principled framework for learning hybrid systems that evolve through both continuous flows and discrete jumps, there remains a rich landscape for further exploration. One particularly promising direction is the integration of ideas from modern diffusion-based generative models [16], which have seen remarkable success in continuous data domains such as images, audio, and text embeddings.

Classical jump-diffusion processes have long been studied in fields like finance and engineering, and recent advances in machine learning suggest we could extend diffusion modeling to hybrid scenarios. For instance, one could imagine defining a forward "noising" process that gradually perturbs both the continuous latent trajectory and the set of discrete events, transforming an observed hybrid system into a simpler, more tractable reference distribution. A suitably trained diffusion model could then learn to reverse this noising process, jointly denoising and “de-jumping” the system’s evolution. Such a construction would enable generative modeling, simulation, and interpolation of complex hybrid dynamics without relying on strict parametric assumptions, since such a system will be modeling stepwise maps between distributions.

However, the integration of diffusion models in this setting poses significant conceptual and technical challenges. Unlike standard Euclidean data spaces, hybrid processes span both continuous and measure-valued elements, making it nontrivial to design forward diffusions and corresponding score functions. Diffusion-based methods that incorporate Lévy processes or represent event sequences as sparse measures in time may offer a starting point. Another potential avenue involves combining flow matching techniques which were originally developed for continuous distributions with measure-theoretic constructions that handle jumps[17].

7 Closing Thoughts

The integration of stochastic processes and machine learning offers a transformative approach to financial risk assessment, equipping institutions with tools to model uncertainty in near real-time, adapt to market complexities, and optimize decision-making. These methodologies are not merely academic pursuits; they address the urgent need for resilient and transparent risk management systems in an era of rapid economic and technological transformation.

By synthesizing mathematical theory, computational advances, and practical applications, we provide robust evidence that neural network approximations of a class of stochastic processes can be practical and immensely useful. As the field evolves, continued innovation and interdisciplinary collaboration will be essential to navigating the challenges and opportunities of an increasingly uncertain financial landscape.

References

[1] Karatzas, I., & Shreve, S. E. (1991). *Brownian Motion and Stochastic Calculus.* Springer-Verlag.

[2] Bachelier, L. (1900). Théorie de la spéculation. *Annales Scientifiques de l'École Normale Supérieure*, *17*, 21–86.

[3] Black, F., & Scholes, M. (1973). The Pricing of Options and Corporate Liabilities. *Journal of Political Economy*, *81*(3), 637–654.

[4] Hull, J. C. (2018). *Options, Futures, and Other Derivatives* (10th ed.). Pearson.

[5] Cont, R., & Tankov, P. (2004). *Financial Modelling with Jump Processes.* Chapman and Hall/CRC.

[6] Gu, S., Kelly, B., & Xiu, D. (2020). Empirical Asset Pricing via Machine Learning. *The Review of Financial Studies*, *33*(5), 2223–2273.

[7] Sirignano, J., & Cont, R. (2019). Universal features of price formation in financial markets: Perspectives from Deep Learning. *Quantitative Finance*, *19*(9), 1449–1459.

[8] Glasserman, P. (2003). *Monte Carlo Methods in Financial Engineering.* Springer.

[9] Buehler, H., Gonon, L., Teichmann, J., & Wood, B. (2019). Deep Hedging. *Quantitative Finance*, *19*(8), 1271–1291.

[10] Athey, S., & Imbens, G. W. (2019). Machine Learning Methods Economists Should Know About. *Annual Review of Economics*, *11*(1), 685–725.

[11] A.V. Skorokhod, "Limit theorems for stochastic processes" Probab. Appl. , 1–3 (1956) pp. 261–290

[12] Jia, Junteng, and Austin R. Benson. "Neural jump stochastic differential equations." Advances in Neural Information Processing Systems 32 (2019).

[13] Almgren, Robert, and Neil Chriss. "Optimal execution of portfolio transactions." Journal of Risk 3 (2001): 5-40.

[14] Cartea, Alvaro, and Sam Howison. "Option pricing with Lévy-stable processes generated by Lévy-stable integrated variance." Quantitative Finance 9.4 (2009): 397-409.

[15] Kingma, Diederik P. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

[16] Rombach, Robin, et al. "High-resolution image synthesis with latent diffusion models." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022.

[17] Lipman, Yaron, et al. "Flow matching for generative modeling." arXiv preprint arXiv:2210.02747 (2022).

[18] Applebaum, D. "Levy Processes and Stochastic Calculus." Cambridge Studies in Advanced Mathematics 116 (2009).

[19] Cybenko, George. "Approximation by superpositions of a sigmoidal function." Mathematics of control, signals and systems 2.4 (1989): 303-314.

[20] Bengio, Yoshua, Ian Goodfellow, and Aaron Courville. Deep learning. Vol. 1. Cambridge, MA, USA: MIT press, 2017.

[21] Rudin, Walter. Principles of mathematical analysis. Vol. 3. New York: McGraw-hill, 1964.

8 Remarks

8.1 Remark 1: Arzela-Ascoli theorem

The Arzela-Ascoli theorem is a fundamental result in real analysis, providing conditions under which a family of continuous functions is relatively compact. We state the theorem and provide a brief explanation of its relevance:

Theorem 8 (Arzela-Ascoli Theorem)**.** Let X be a compact metric space, and let $\mathcal{F}$ be a subset of the space of continuous real-valued functions on X, denoted $C(X)$. Then $\mathcal{F}$ is relatively compact in $C(X)$ (with respect to the uniform norm) if and only if the following two conditions hold:

1. **Equiboundedness:** There exists a constant $M > 0$ such that for all $f \in \mathcal{F}$ and all $x \in X$, we have $|f(x)| \leq M$.

2. **Equicontinuity:** For every $\epsilon > 0$, there exists a $\delta > 0$ such that for all $f \in \mathcal{F}$ and all $x, y \in X$, if $d(x, y) < \delta$, then $|f(x) - f(y)| < \epsilon$. Here, d is the metric on X.

Definitions:

- **Compact Metric Space:** A metric space X is compact if every sequence in X has a convergent subsequence.

- **Space of Continuous Functions $C(X)$:** The space of all continuous functions $f : X \to \mathbb{R}$.

- **Relative Compactness:** A subset $\mathcal{F}$ of a metric space is relatively compact if every sequence in $\mathcal{F}$ has a subsequence that converges to a limit in the metric space.

- **Uniform Norm on $C(X)$:** For $f \in C(X)$, the uniform norm is defined as $||f||_\infty = \sup_{x \in X} |f(x)|$.

- **Equiboundedness:** A family of functions $\mathcal{F}$ is equibounded if there exists a uniform bound for the absolute value of all functions in the family at all points in X.

- **Equicontinuity:** A family of functions $\mathcal{F}$ is equicontinuous if all functions in the family are continuous in a uniform manner, as quantified by the definition.

The theorem's proof is typically found in a standard real analysis textbook. Informally, the proof of necessity involves showing that if the space is not equibounded or equicontinuous, then we can construct a sequence without a uniformly converging subsequence. The proof of sufficiency often involves constructing a uniformly converging subsequence, using the Bolzano-Weierstrass theorem to obtain a pointwise convergent subsequence, and then using the equicontinuity to show that convergence is indeed uniform. A complete proof can be found in Walter Rudin's *Principles of Mathematical Analysis* [21] (Theorem 7.25).

8.2 Remark 2

In the proof of the Infinitesimal Generator theorem, we utilized key properties of the inverse Fourier transform to relate multiplication by $-iu$ and $-u^2$ in the Fourier domain to derivatives in the spatial domain. Recall that the Fourier transform of a function $f(x)$ is defined as:

$$\hat{f}(u) = \int_{\mathbb{R}} e^{iux} f(x) dx$$

and the inverse Fourier transform is given by

$$f(x) = \frac{1}{2\pi} \int_{\mathbb{R}} e^{-iux} \hat{f}(u) du.$$

Then, we have the following well-known properties:

1. **First Derivative:** If f is a sufficiently well behaved function (i.e. it is smooth and sufficiently fast decaying), then taking the derivative with respect to position, we see that

$$f'(x) = \frac{1}{2\pi} \int_{\mathbb{R}} (-iu) e^{-iux} \hat{f}(u) du.$$

   This demonstrates that multiplication by $-iu$ in the Fourier domain corresponds to taking the first derivative in the spatial domain, and this argument can be made precise via integration by parts.

2. **Second Derivative:** Similarly, taking the second derivative with respect to position yields:

$$f''(x) = \frac{1}{2\pi} \int_{\mathbb{R}} (-u^2) e^{-iux} \hat{f}(u) du.$$

   Thus, multiplication by $-u^2$ in the Fourier domain corresponds to taking the second derivative in the spatial domain.

These properties are fundamental in Fourier analysis, and allow us to connect the behavior of the infinitesimal generator in the Fourier domain to its spatial form.